Thesis

GUIDING MACHINE LEARNING DESIGN WITH INSIGHTS FROM SIMPLE SANDBOXES

Bingbin Liu

August 2024
CMU-ML-24-110

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This research was funded by: Air Force Research Laboratory award FA87501720152; National Science Foundation awards IIS1909816 and CCF2238523; Office of Naval Research awards N000141812861 and N000142312368; and a grant from PricewaterhouseCoopers.

Keywords: machine learning, theory, self-supervised learning, unsupervised learning, noise contrastive estimation, contrastive learning, masked prediction, parameter identifiability, tensor decomposition, Transformer, self-attention, formal language, circuit complexity, algebraic automata theory, hallucinations, generalization, out-of-distribution, interpretability, knowledge distillation, feature learning, curriculum

For Xianming, Mi, and Tim

Abstract

Machine learning research often follows two seemingly distinct approaches: the empirical approach, which excels at developing practical algorithms, and the theoretical approach, which offers formal guarantees and resource-efficient solutions. While the empirical approach often relies on heuristics and demands costly large-scale experiments, the theoretical approach often hinges on unrealistic assumptions, limiting its applicability to real-world scenarios.

This thesis aims to bridge these approaches by studying "sandbox" setups, which are conceptual abstractions of complex systems. A well-designed sandbox is both *minimal*, enabling clean theoretical analyses and rapid, accessible empirical investigations, and *representative*, ensuring that findings within the sandbox are generalizable to broader contexts.

This thesis details the use of the sandbox approach to understand the task design, the model class, and the learning process. Chapter 2 examines design choices in machine learning tasks, focusing on how self-supervised methods—namely, contrastive learning and masked prediction—extract information from sequential data. Chapter 3 analyzes the capabilities and limitations of a specific model class, with an emphasis on Transformers for sequential reasoning. This chapter characterizes the feasible solutions, discusses generalization challenges, and proposes improvements with implications on interpretability. Finally, Chapter 4 examines factors that impact the learning process. It identifies and addresses an algorithmic challenge in contrastive learning, and explores how knowledge distillation can improve sample complexity.

Acknowledgment

I feel incredibly lucky to have crossed paths with many amazing mentors and friends.

First and foremost, I would like to thank my advisors Pradeep Ravikumar and Andrej Risteski. I am deeply grateful that both of them, as theorists, were willing to take me as their advisee and had faith in me despite my lack of a track record in theoretical research at the time. I remember asking Pradeep during the open house whether my lack of theoretical background was a concern. He kindly reassured me that it wasn't, emphasizing that perseverance would be the key to success. His encouragement gave me the confidence to switch my research topic from computer vision in my master's to theoretical machine learning as I began my PhD. Andrej welcomed me as his first batch of students and guided me with invaluable lessons that helped me build a strong foundation. I still remember him introducing to me what an ϵ-net was on the board, assuring me that I would soon develop my own toolbox—a toolbox I continue to expand, thanks to his teaching and the wide array of resources (sometimes wonderfully niche) he's shared. Over the years I have learned so much from both of them, from problem formulation to proof techniques, from career/life advice to restaurant recommendations. They have also generously given me plenty of fun discussions and free food opportunities. Above all, I want to thank them for teaching me to be selective of ideas, to value the clarity of thoughts, and to seek clean formulations that lead to elegant solutions.

I've had the fortune to receive guidance from many incredible mentors (many of whom double as friends): this include my committee members—Cyril Zhang, Daniel Hsu, and Sham Kakade, and mentors at CMU and my internships—Arun Suggala, Leqi Liu, Aaditya Ramdas, Geoffrey Gordon, Nihar Shah, Sivaraman Balakrishnan, Zachary Lipton, Surbhi Goel, Akshay Krishnamurthy, Jordan Ash, Yi Zhang, Allie Del Giorno, Anh Nguyen, Janardhan Kulkarni, Ronen Eldan, and Sebastien Bubeck. I am impressed by them for being extremely knowledgeable and smart, and at the same time kind and approachable. Arun was the first (non-advisor) mentor I had at CMU, who was very technically strong yet was able to explain things in simple terms that I would understand. Leqi is two years senior in the program and acted as a wise elder sister (even though she's of my age) who gave me advice on classes, research, and career along the way. The faculty at MLD, especially Aadi, Geoff, Nihar, Siva, and Zack, have been so generous in sharing their knowledge, wisdom, and time that I've benefited from tremendously. During the pandemic, I had the opportunity to work on a project with Daniel, who is both incredibly sharp and extremely patient and supportive. I enjoyed reading and learned so much from his work and Sham's, both on latent-variable models at that time and more recently on language models. These are just two of the many areas in which they are experts, and I'm constantly amazed by the breadth and depth of their knowledge. Exactly in the mid-point of my PhD, I was extremely fortunate to be matched with the dream team of Akshay, Cyril, Jordan, and Surbhi during my first internship at MSR, who taught me so much both in research (from Jupyter notebook to group theory) and in being kind, supportive persons that make the community better. I also want to give a special shout-out to Cyril for making my day when he traveled to my thesis defense in person. The internship was not only fun and rewarding, but also a pivotal point in my research direction that led to a bulk of the work in this thesis, a second internship mentored by Yi (who taught me to learn to identify "bottleneck" problems in research and was tremendously helpful both during and after my internship) and the team, and a postdoc position with Sham which I'm very excited to start in a few months.

I also owe a great deal to my mentors who paved the way for me to pursue a PhD: my undergraduate professors Francis Lau, Kenneth Wong, Ben Hardekopf, who taught me the very first ideas of research and were empathetic and encouraging towards a complete novice like me; the undergraduate graduate student mentors Chaofeng Chen, Guanying Chen, Kai Han, Wei Liu, and Zhenfang Chen, who welcomed me in their office and were so patient in teaching me (and debugging my CUDA installation); and my master's mentors De-An Huang, Serena Yeung, Juan Carlos Niebles, Ehsan Adeli-mosabbeb, Francesca Rinaldo, Arnold Milstein, and Fei-Fei Li, who gave me the opportunity, support, and encouragement to work on my first research projects and continue to a PhD. Their valuable advice continues to help me as I progress through various stages of my life.

Many thanks to the amazingly capable and thoughtful MLD staff, Christy Melucci, Dorothy Holland-Minkley, Laura Winter, Mary Stech, Sharon Cavlovich, and especially Diane Stidle. They have helped (and saved) me in numerous ways, including but not limited to coffee machine instructions, arranging events, bringing their puppies, helping with my struggles, checkups, pep talks, hallways chats, sharing fun events, and just looking out for me (and frankly our entire department) overall.

I feel so fortunate to be surrounded by amazing friends (please forgive me if I've unintentionally overlooked anyone!). Thanks to my labmates Elan Rosenfeld, Tanya Marwah, Yuchen Li, Che-Ping Tsai, Chih-Kuan Yeh, Daniel Jeong, Goutham Rajendran, Juyong Kim, Kartik Gupta, Rattana Pukdee, Runtian Zhai, and collaborators Abhishek Panigrahi, Kaiyue Wen, Omar Chehab, Sadhika Malladi, Zixin Wen, for the numerous discussions, friendly critiques, chats inside and outside academic settings, and technical/mental support. Thanks to my officemates Arundhati Banerjee, Haitian Sun, Niki Hasrati, Tomas Gomez, Yuda Song for snacks, random conversations, and our glorious office mission. Thanks to my (uncategorized) friends Asher Trockman, Aakash Lahoti, Amrith Setlur, Benny Jiang, Chelsea Yang, Charvi Rastogi, Christina Baek, Isaac Grosof, Jeff Xu, Jennifer Hsia, Jinjin Tian, Junhong Shen, Lingjing Kong, Sarah Wang, Shuqi Dai, Siqi Liu, Stefani Karp, Valerie Chen, Yixuan Xu, for the chats, walks, foods, and bouldering sessions that have made my time at CMU and Pittsburgh fun and memorable. Thanks to Nari Johnson for organizing the MLD women lunch and thanks to all the girls in the MLD for the roses, buds, and thorns in and beyond our lunch chats. Thanks to Chulin Xie, Cindy Niu, Huihan Liu, Hejie Cui, Jinghang Hu, for giving me a lovely summer at Bellevue and for sharing life updates every now and then that have made many of my days.

A special shout-out to the amazing girls in my cohort: Arundhati Banerjee, Ashwini Pokle, Stephanie Milani, Tanya Marwah, and Yusha Liu. I was extremely lucky to have such a family to share each others' laughs and tears throughout my PhD. Arundhati and I share the same office and birthday. Ashwini introduced proper Indian cooking and BTS to me. Stephanie has top-notch advice and fashion sense. Tanya is so tech-savvy and led me to Notion, Roam, and Hydra. Yusha could be a stand-up comedian any time and make a fortune. These are just a few fun facts that I want to share about these girls, who are brilliant, witty, kind, hard-working, so similar yet so unique. They know best how I can't express in words how much they mean to me. I sincerely wish them all the very best and brightest in their journeys forward.

Many thanks to my friends who have known and supported me from way back when: Shiyuan Zhu (my dear partner in crime since two years old), Lingshu Kong (the proud mom of my godson), Haocheng Li, Runze Ren, Guanlin Qiao, Kaiwen Zhang, Kanghui Zhao, Xiaoye Guo, Yang Liu, Yun Lu, Congxiao Zhou, Mengxuan You, Sijia Liu, Yuncong Ding, Fangyuan Xu, Jiayi Li. Although we can't meet in person as much as I would like, it's always a pleasure to hear from them, and it gives me an energy boost whenever we get together.

Thanks also to my furry friends Bobo, Floe, Raven, and Schrodinger (well, also our stuffed animals Songsong and Panghu). Their trust towards me (as a food provider and as a friend) is pure and endearing—and of course, thanks to my friends for trusting their beloved pets on me. Thanks to the sunny moments and clear skies (however brief) in Pittsburgh, the snow in January 2022 which gave me a fun break from a stuck proof, and the magnificent tree on the Fifth Avenue who greets me with its beautiful fall colors every year.

Finally, I am deeply grateful for my family. Thanks to my parents Xianming and Mi for their unwavering love and support, for believing in and being proud of me, and for being my role models in many ways. Thanks to my cousin Hanhan for being my remote life tutor and research buddy, and to her family for their encouragement and lovely visits. Finally, many thanks to my husband Jun-Ting (Tim) Hsieh for figuring out PhD and life together with me. Tim is my dear friend, my math tutor, and my mental stabilizer. I couldn't have done this without him.

Contents

Chapter 1

Introduction

There are two primary methodologies for advancing machine learning: the empirical approach which derives insights largely through trial and error, and the theoretical approach which builds on formal guarantees. Each approach comes with its strengths and limitations. The empirical approach excels at developing practical algorithms and real-world applications, with performance that often improves with scale and complexity [Sutton, 2019]. However, this reliance on large-scale experiments also makes advancing state-of-the-art methods increasingly resource-intensive, limiting accessibility to research and making it prohibitively expensive to gain a scientific understanding of the mechanisms driving these methods. In contrast, the theoretical approach requires minimal computational resources and provides provable guarantees. Yet, these guarantees are often derived under strict assumptions that may not hold in practical settings, limiting their applicability to real-world problems.

Fortunately, these two approaches are not mutually exclusive but complementary. Empirical findings frequently uncover phenomena that challenge existing theories. For example, the generalization of overparameterized neural networks has exposed the limitations of uniform convergence [Nagarajan, 2021], while the edge-of-stability phenomenon [Cohen et al., 2021] has inspired a reevaluation of optimization principles. Conversely, theoretical insights deepen our understanding of empirical methods and offer guidance that reduces the cost of trial-and-error experiments and informs the design of new algorithms. Classic examples include VC-theory-based SVM Cortes and Vapnik [1995], boosting Schapire [1990], and optimizers like Adam [Kingma and Ba, 2014], which have been widely adopted in the field. More recently, techniques like the tensor program [Yang and Hu, 2020, Yang et al., 2022] and dynamical mean field theory [Bordelon and Pehlevan, 2022, 2024] have helped eliminate the need for expensive hyperparameter searches, further bridging the gap between theory and practice.

This thesis aims to take the advantages of both the theoretical and the empirical approaches by studying simple *sandboxes*, [1] which serve as conceptual abstractions of complex systems. A useful sandbox should be *minimal*, meaning it is the simplest setting that allows for the study of the question of interest. This simplicity facilitates theoretical analyses and provides a lightweight experimental setup that supports rapid iterations and broad accessibility. At the same time, a sandbox should be sufficiently *representative* to ensure that its findings generalize beyond the sandbox itself and can offer insight to

[1] If you like this approach, please also read Edelman [2024].

more complex systems. The sandbox should ideally also be *controllable* so that it can be extended to incorporate varying levels of complexity as needed.

Sandboxes offer many benefits to building the theory and science of machine learning. From a practical point of view, sandboxes help with *algorithm designs*, as they allow for isolating the main bottleneck of a problem for both theoretical analyses and empirical verification; Section 4.1 will provide an example where the simple Gaussian mean estimation is used as a sandbox to highlight an algorithmic challenge in noise contrastive estimation [Gutmann and Hyvärinen, 2010a]. Sandboxes are also well-suited for *diagnostics and stress testing* such as in addressing the long tails of real-world distributions (see Section 3.5 for an example), especially with fully synthetic sandboxes where there is an added benefit of access to a practically infinite amount of data. Besides these practical advantages, the most significant value of sandboxes is that they provide conceptual abstractions that promote *clarity*. A classic example is the abstraction of railway traffic flow in Harris and Ross [1955], whose simplified formulation was instrumental in establishing the connection between max flow and min cut [Ford and Fulkerson, 1956].

As noted in Harris and Ross [1955] and more broadly, identifying suitable sandboxes is as much an art as a science, and the appropriate level of abstraction provided by a sandbox is often highly context-dependent. Encouragingly though, sandboxes have proven effective in many areas of machine learning. For instance, studies of representability often draw on sandboxes inspired by formal languages [Bhattamishra et al., 2020b, Hahn, 2020, Yao et al., 2021a, Liu et al., 2022a] and communication complexity [Sanford et al., 2024]. Similarly, optimization analyses frequently adopt simplified models trained on Gaussian or structured data [Abbe et al., 2022, 2023b, Li et al., 2023, Nichani et al., 2024]. Moreover, sandboxes are highly versatile. Boolean data, for instance, can be used to study a range of topics, including representation capacity [Hahn, 2020, Chiang and Cholak, 2022], optimization [Abbe et al., 2023b, Glasgow, 2023], inductive biases Bhattamishra et al. [2022], Morwani et al. [2024], (length) generalization behaviors [Liu et al., 2023a, Anil et al., 2022b], and key design choices of the modern machine learning pipeline, such as the impact of data curriculum [Abbe et al., 2023a, 2024].

This thesis applies the sandbox methodology to study the following aspects of machine learning, with the goal of principled and efficient progress: 1) design choices of the *task*, which determines the ideal scenario without practical constraints; 2) characteristics and limitations associated with the *model class*; and 3) the *learning process* governed by various factors.

Task design in self-supervised learning. Predominantly, a machine learning task aims to achieve a goal by minimizing an objective function. The task and the objective are designed to reflect our belief or prior knowledge, but they are only surrogates of the ultimate goal and can be misleading. A classic example is reward hacking in reinforcement learning: the goal is to encourage certain desired behaviors, the learning task is to maximize a reward function, but an inappropriately designed reward can lead to unwanted behaviors and surprising failures. Therefore, when designing a machine learning task, it is important to understand what the task entails.

Such consideration is especially relevant in the context of self-supervised learning, where the model is trained to leverage inherent structures in the data rather than manually collected labels. Since there are no labels, it is clear that the self-supervised task is intended as a surrogate of the final goal that we care about. Nevertheless, it is widely believed that learning from properly identified structures can be useful. For example, if our goal is to classify between images of dogs and cars, one self-supervised task we can use is to provide models with images at different rotation angles and train the model to identify the angle of each image [Gidaris et al., 2018]. Intuitively, a model that can correctly identify the rotation angle should have learned some features that are useful to distinguish between dogs and cars. The question though is how to quantify the precise effect of the design parameters.

Our contribution. In Chapter 2, we will study the effect of design parameters in a discriminative and a generative self-supervised learning task.

The discriminative task considered in Section 2.1 is contrastive learning of strong-mixing continuous-time stochastic processes. Contrastive learning is one of the leading learning paradigms of self-supervised learning, where a model is trained to solve a classification task constructed from unlabeled data. Despite its empirical popularity in mutliple domains, theoretical understanding of many aspects of the task, both statistical and algrithmic, remained fairly elusive. Our results study time-series data, a natural structure for natural languages [Devlin et al., 2018b], finance data [Ait-Sahalia et al., 2008a], and brain imaging data [Hyvarinen and Morioka, 2016]. The sandbox we choose is for the data to come from a strong-mixing continuous-time stochastic process, where the mixing speed relates to the strength of the sequential dependency in data. We show that a properly constructed contrastive learning task can be used to estimate the transition kernel for small-to-mid-range intervals in the diffusion case. Our results provide sample complexity bounds for solving this task and quantitatively characterize what the value of the contrastive loss implies for distributional closeness of the learned kernel. Moreover, we illuminate the appropriate settings for the contrastive distribution, such as the intervals at which training samples are drawn.

In Section 2.2, we turn to the masked prediction task and study how the choice of the masking ratio affects the identifiability of the data generative model. A masked prediction task trains the model by predicting missing tokens in the input sequence. It is a popular method for both natural languages [Devlin et al., 2018b] and visual data He et al. [2021]. The sandbox for data is a family of parametric probabilistic models. Given an optimal predictor with a suitably chosen parametric form, we ask whether we can read off the ground truth parameters of the probabilistic model from the optimal predictor. While incarnations of this approach have already been successfully used for simpler probabilistic models (e.g. learning fully-observed undirected graphical models [Ravikumar et al., 2010]), we focus instead on latent-variable models capturing sequential structures—namely Hidden Markov Models with both discrete and conditionally Gaussian observations. Our results show that there is a rich landscape of possibilities, out of which some prediction tasks yield identifiability, while others do not. As for proof techniques, we uncover close connections with uniqueness of tensor rank decomposition, a widely used tool in studying identifiability through the lens of the method of moments.

Model class: capabilities and limitations of Transformers Given a learning task, another important choice is the model class adopted to learn the task, which crucially affects the solution found in practice. One consideration is the representational capacity, namely, whether the model class is sufficiently rich to represent an optimal solution to the learning task. A classic example often seen in an introductory machine learning class is that the XOR function can be solved by a linear classifier but can be solved by a 2-layer neural network with nonlinear activation. Often times, we also care about representational efficiency. For instance, the focus of Chapter 3, Transformers, is more parameter efficient than other architectures such as graph neural networks or recurrent models in solving some tasks [Sanford et al., 2024] but not some others [Bhattamishra et al., 2024]. The separation in representability also extends to learnability [Wang et al., 2024] and affects the generalization behaviors [Anil et al., 2022a, Liu et al., 2023b].

The particular architecture we are interested in is the Transformer [Vaswani et al., 2017], which has been the dominant sequence model in the recent years. We will center the discussion around (deductive) reasoning modeled as synthetic combinatorial tasks. Specifically, the sandbox we focus on is finite-state automata, where a state transition represents one reasoning step. Despite being at the simplest level of the automata hierarchy, finite-state automata can capture various practical applications and are important building blocks of more complex tasks. In this thesis, we consider both the representational capacity and the generalization behaviors.

Our contribution. We start in Section 3.4 with positive representability result showing that Transformers, while lacking recurrence, are able to perform long chains of sequential reasoning using far fewer layers than the number of reasoning steps. We show that a low-depth Transformer can represent the computations of *any* finite-state automaton (thus, any bounded-memory algorithm), by hierarchically reparameterizing its recurrent dynamics. Our theoretical results characterize *shortcut solutions*, whereby a Transformer with $o(T)$ layers can exactly replicate the computation of an automaton on an input sequence of length T. We find that polynomial-sized $O(\log T)$-depth solutions always exist; furthermore, $O(1)$-depth simulators are surprisingly common, and can be understood using tools from Krohn-Rhodes theory [Krohn and Rhodes, 1965] and circuit complexity.

However, solutions found in practice may not recover these computational shortcuts. Empirically, while Transformers are able to reach perfect accuracy on in-distribution validation samples across a wide variety of automata, they suffer a significant performance drop when tested out-of-distribution (OOD). This is in stark contrast to recurrent neural networks (RNNs), which are able to generalize perfectly (on the finite test samples covered in the experiments) with far less data and compute. Towards making sense of this fundamentally unsolved generalization problem in Transformer, our work in Section 3.5 identifies and analyzes the phenomenon of *attention glitches*, in which the Transformer architecture's inductive biases intermittently fail to capture robust reasoning. We hypothesize that attention glitches account for (some of) the closed-domain hallucinations [Ji et al., 2023] in natural LLMs. To isolate the issue, we introduce *flip-flop language modeling* (FFLM), a parametric family of synthetic benchmarks designed to probe the extrapolative behavior of neural language models. FFLM is a variant of the flip-flop automaton, a primitive identified by Krohn-Rhodes decomposition explained in the previous section. As a sandbox for the copy or recall abilities in general reasoning tasks, FFLM requires a model to copy binary symbols over long-range dependencies while ignoring the tokens in between. We find that Transformer FFLMs suffer from a long tail of sporadic reasoning errors, some of which we can eliminate using various regularization techniques. Our preliminary mechanistic analyses show why the remaining errors may be very difficult to diagnose and resolve.

In addition to diagnosing failure modes of Transformers, we also study how theoretical insights can be used to reflect on current practices and improve OOD performance. In Section 3.6, we use Dyck languages as a sandbox to illustrate the limitations of certain interpretability methods. Interpretability methods aim to understand the algorithm implemented by a trained model (e.g., a Transofmer) by examining various aspects of the model, such as the weight matrices or the attention patterns. Our work takes a critical view of methods that exclusively focus on individual parts of the model, rather than consider the network as a whole. The sandbox we consider is the (bounded) Dyck language. Theoretically, we show that the set of models that (exactly or approximately) solve this task satisfy a structural characterization derived from ideas in formal languages, specifically the pumping lemma. We use this characterization to show that the set of optima is qualitatively rich; in particular, the attention pattern of a single layer can be "nearly randomized", while preserving the functionality of the network. We also show via extensive experiments that these constructions are not merely a theoretical artifact: even after severely constraining the architecture of the model, vastly different solutions can be reached via standard training. Thus, interpretability claims based on inspecting individual heads or weight matrices in the Transformer can be misleading. Moreover, adding a regularization term akin to a Transformer's implementation of the pumping lemma improves the OOD performance of the model.

Learning process. The aforementioned gap between the theoretical constructions and the practical solutions results from the fact that most learning problems in modern machine learning are non-convex, which means that the optimization process and hence the final solutions are sensitive to various factors in the learning process. This prompts the need to study the learning process itself, which is the focus of Chapter 4. We will see two examples where we accelerate training by modifying the objective function, the update rule, or the source of the supervision signal.

Our first example is to improve the loss landscape of noise-contrastive estimation (NCE), a statistically consistent method for learning unnormalized probabilistic models. NCE is computationally cheaper than maximum likelihood estimation (MLE) and is well known to be consistent. However, it has been empirically observed that the choice of the noise distribution is crucial for NCE's performance, although such observations have never been made formal or quantitative. In fact, it is not even clear whether the difficulties arising from a poorly chosen noise distribution are statistical or algorithmic in nature. In Section 4.1, we formally pinpoint reasons for NCE's poor performance when an inappropriate noise distribution is used, with a focus on the algorithmic aspect of these challenges. Using a simple sandbox of 1-dimensional Gaussian mean estimation, we prove that they are due to an ill-behaved loss landscape that is extremely flat near the optimum. We then introduce a 2-part mitigation for the landscape issues: a variant of NCE called eNCE (exponential NCE) which changes NCE's log loss to an exponential loss, and a modified gradient update step with a simple change from gradient descent to normalized gradient descent. These changes lead to an exponential improvement in the convergence speed when the target and noise distributions are in a given exponential family, and also work well empirically for more complex distributions.

In our second example, we consider alternative supervision signals for accelerating the training of small models. Our focus is knowledge distillation [Hinton et al., 2015], where a student model learns from a teacher model. Knowledge distillation has seen successful adoption in various scenarios [Jia et al., 2021, Touvron et al., 2021, Sanh et al., 2019, Gunasekar et al., 2023]. Perhaps counterintiutively, it has been repeated confirmed that a better-performing teacher does not always lead to a better-performing student. Prior work attributes this to a large teacher-student performance gap and aim to mitigate by providing intermediate supervision [Mirzadeh et al., 2019, Jin et al., 2019]. In Section 4.2, we analyze an empirically successful approach called *progressive distillation*, where several intermediate checkpoints of the teacher are used successively to supervise the student as it learns [Anil et al., 2018, Harutyunyan et al., 2023]. We will consider two sandboxes. The first is sparse parity, a simple combinatorial task whose difficulty of learning is well understood. The second is probabilistic context-free grammar (PCFG), which is a natural extension of the Dyck languages and captures the hierarchical structure in natural languages. We show that progressive distillation accelerates the training of the students via an *implicit curriculum* that is present only in the intermediate checkpoints (but not the final checkpoint after convergence). The improvement in training efficiency is demonstrated both theoretically by a reduced sample complexity, and empirically on both the two sandboxes and real-world natural language datasets.

Chapter 2

Illuminating task designs of self-supervised learning

In this chapter, we study the design choices involved in learning tasks, with a particular emphasis on self-supervised methods and the recovery of the underlying data distribution. Self-supervised learning presents an alternative to supervised learning by reducing the reliance on costly human-annotated data. One of the major advantages of self-supervised methods is their ability to leverage large volumes of "in the wild" data, such as images and text from the internet, making them more scalable than supervised approaches. A problem though is that the effectiveness of self-supervised methods can vary widely depending on specific design choices of the task, whose search space is often too costly to be explored thoroughly through empirical investigations alone. In contrast, sandboxes can help illuminate the search space by leveraging insights from theoretical analyses.

This chapter includes two examples on learning sequential data. In Section 2.1 (based on Liu et al. [2021]), we show how the noise distribution in contrastive methods [Chen et al., 2020, Tian et al., 2020c, Gutmann and Hyvärinen, 2010a, Rhodes et al., 2020] influences the sample complexity of learning the underlying distribution, using continuous-time stochastic processes as the sandbox. We study the choice of the observation interval and derive the first finite-sample bound for contrastive learning on such processes. In Section 2.2 (based on [Liu et al., 2022b]), we analyze masked prediction methods, for which the masking ratio is a key design parameter. We study the parameter identifiability of a generative model for sequential data, using the hidden Markov model and its continuous variant as the sandboxes. Our findings indicate that this ratio controls the task complexity and plays a crucial role in determining parameter identifiability.

2.1 Contrastive learning of strong-mixing continuous-time stochastic processes

One of the paradigms of learning from unlabeled data that has seen a lot of recent work in various application domains is "self-supervised learning". These methods supervise the training process with information inherent to the data without requiring human annotations, and have been applied across computer vision, natural language processing, reinforcement learning and scientific domains.

Despite the popularity, they are still not very well understood—both on the theoretical and empirical front—often requiring extensive trial and error to find the right pairing of architecture and learning method. In particular, it is often hard to pin down what exactly these methods are trying to learn, and it is even harder to determine what is their statistical and algorithmic complexity.

The specific family of self-supervised approaches we focus on in this work is *contrastive learning*, which constructs different types of tuples by utilizing certain structures in the data and trains the model to identify the types. For an example in vision, Chen et al. [2020] apply two random augmentations (e.g. crops and discolorations) on each training image, and form pairs that are labeled as either positive or negative depending on whether two augmentations are from the same image or not. In NLP, one of the tasks in Devlin et al. [2018a], Tosh et al. [2020b] trains the model to predict whether two half-sentences are from the same original sentence.

In this paper, we focus on understanding a natural type of contrastive learning tasks for time series data—a natural structure in NLP [Devlin et al., 2018a, Tosh et al., 2020b], finance [Ait-Sahalia et al., 2008b], and brain imagining research [Hyvarinen and Morioka, 2016] More precisely, we focus on data coming from a discretization of a *diffusion* process—a common modeling assumption in many of these domains—and show that a natural distinguishing task we set up on pairs of samples from the time series approximately learns the transition kernel of the stochastic process.

Note, a diffusion process is a continuous-time stochastic process and we are interested in learning transition kernels for "mid-range" time intervals, that is, intervals that are potentially too large for the Euler scheme to be accurate. These transition kernels are not easy to learn in general through standard maximum likelihood methods, as closed-form solutions are complicated [Ait-Sahalia et al., 2008b] and often do not exist, and empirical estimations can also be challenging [Milstein et al., 2004]. To our knowledge, our work is the first one to use contrastive learning to learn such transition kernels. Moreover, we provide a statistical complexity analysis—that is, analyzing the number of samples required to learn a good approximation of the transition kernel. This helps quantify certain aspects of contrastive learning —how should we choose the contrast distribution, and how a small loss on the contrastive task transfers to closeness of the transition kernel estimate.

Related Work There is a large body of recent empirical work on self-supervised learning in general, which we won't make an effort to survey in full, as it does not directly relate to our results.

There have been some recent works on trying to understand theoretically why and when self-supervised learning works. The closest ones in spirit to our work are Tosh et al. [2020d] and Hyvarinen and Morioka [2016], but there are significant differences with both. Tosh et al. [2020d] focus on a data distribution coming from LDA (topic modeling), and characterize the kinds of downstream classification tasks the learned predictor is useful for. Hyvarinen and Morioka [2016] focus on a time series setting as well but with several differences as highlighted below.

First, they work with a latent-variable model, and show that their method recovers some function of the latent. One example parametrization is an exponential family, and the function of the recovered latent variable depends on the choice of the exponential family.

Second, they assume the data in the time series can be subdivided into "blocks", such that the distribution remains the same in each block and is sufficiently different from the others. In practice, it is not clear how to choose these "blocks" or how to even verify the assumptions needed on them. We do not need this "blocking", but our data needs to come from the stationary distribution of the process.

Third, they do not provide an analysis on statistical complexity. In particular, important aspects of how various hyperparameters are chosen and affect the quality of the learned predictor—the size of the blocks, the amount of "difference" between the blocks—are not clear.

For temporally dependent and stationary data, another related work is by Hyvarinen and Morioka [2017b]. The setup is however different: Hyvarinen and Morioka [2017b] focus on discrete-time data with autocorrelations, whereas we analyze a continuous-time diffusion, leading to different setups and goals for the contrastive task. Moreover, in contrast to our finite sample analysis, their results describe only the asymptotic behaviors, which can hide certain statistical aspects of the algorithm as discussed earlier.

In the simpler iid setting, a classical precursor paper to this is by Gutmann and Hyvärinen [2010b], who apply the contrastive learning approach to learning a distribution from iid samples—by setting up a classification task to distinguish between samples from the target distribution and a simple "contrast" distribution. Their analysis is again asymptotic and only provide sample efficiency bounds in the asymptotic limit (i.e. as the number of samples goes to infinity). More recently, such classical approaches have been combined with more modern generative models based approaches to generate better contrast distributions [Gao et al., 2020], and augmented with intermediate tasks to better handle dissimilar target and contrast distributions [Rhodes et al., 2020].

Finally, other papers on empirical and theoretical properties of contrastive learning that are worth mentioning include Purushwalkam and Gupta [2020], Tian et al. [2020b] and Saunshi et al. [2019], Wang and Isola [2020b]—these are not directly comparable to what we are doing here, as the data models are quite different, as is the flavor of guarantees they show. In particular, these papers work with iid data and focus on learning good representations that can perform well on certain supervised tasks, whereas we use contrastive learning to perform distribution learning, that is, learning the transition kernels.

2.1.1 Results overview

We now formally state our results. We will start with specifying the distributional model for the data and the contrastive learning task, and build intuitions on what the task aims to achieve before stating the formal guarantees.

2.1.1.1 Setup

We will assume our data comes from continuous time series: namely $\{\boldsymbol{x}_t\}_{t\geq 0} \subset \mathbb{R}^d$, drawn according to a stochastic process called the *Langevin diffusion*[1], defined by the stochastic differential equation

$$d\boldsymbol{x}_t = -\nabla f(\boldsymbol{x}_t)dt + \sqrt{2}dW_t, \ \forall t \geq 0, \tag{2.1}$$

for $f : \mathbb{R}^d \to \mathbb{R}$ a convex function, and $\{W_t\}_{t\geq 0} \subset \mathbb{R}^d$ a Wiener process, i.e. $W_s - W_t \sim \mathcal{N}(0, (s-t)\boldsymbol{I}_d), \forall s \geq t \geq 0$. For the reader unfamiliar with diffusions, we can think of a diffusion process as the limit of a discrete sequence of noisy gradient updates with a fresh Gaussian noise: as $\eta \to 0$, the discrete sequence defined by $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta\nabla f(\boldsymbol{x}_t) + \sqrt{2\eta}\xi_t$ where $\xi_t \sim \mathcal{N}(0, \boldsymbol{I})$ converges to the continuous time diffusion [Bhattacharya et al., 1978]. The simplest instantiation of this, when f is quadratic (i.e. ∇f is linear), gives rise to the Ornstein–Uhlenbeck process, which has broad applications in science and finance modeling.

It is well-known [Bhattacharya et al., 1978] that the stationary distribution of the above process is the distribution $\pi(x) \propto e^{-f(x)}$, under relatively mild regularity conditions on f. We will assume that x_0 (and hence all subsequent x_t) marginally follow π—i.e. the process is *stationary*.

We will also need several common assumptions on the f in the generative process.

Assumption 1 (Strong convexity)**.** *f is ρ-strongly convex.*

Assumption 2 (Smoothness of f)**.** *f is infinitely differentiable, L_0-smooth, and ∇f is L_1-smooth.*[2]

Assumption 3 (Linear growth)**.** *There exists a positive constant $K < \infty$, such that $\forall \boldsymbol{z} \in \mathcal{Z}$, $\|\nabla f(\boldsymbol{z})\| \leq K(1 + \|\boldsymbol{z}\|)$.*

Assumption 1 ensures the least singular value of the Hessian is lower bounded by ρ—which ensures that $\int_x e^{-f(x)}$ is finite. Assumption 3 ensures the existence of a solution to equation 2.1, and Assumption 2 ensures the solution of equation 2.1 is unique.[3] We refer the readers to Ait-Sahalia et al. [2008b] for formal justifications of these assumptions. We denote with $\boldsymbol{z}_*$ the minimizer of f, and assume $\boldsymbol{z}_* = \text{vec0}$ for convenience.

Finally, denote $B := \mathbb{E}_\pi\|\boldsymbol{x}\|$. Note that our assumptions on f guarantee a bounded B: let $Z_\pi := \int_{\boldsymbol{z}} \exp(-f(\boldsymbol{z}))\, d\boldsymbol{z}$ denote the partition function of the stationary distribution π, and with $\boldsymbol{x}_* = \text{vec0}$, we have

$$\begin{aligned} B = \frac{1}{Z_\pi}\int_{\boldsymbol{z}} \|\boldsymbol{z}\| \exp(-f(\boldsymbol{z}))d\boldsymbol{z} \leq \frac{1}{Z_\pi}\int_{\boldsymbol{z}} \|\boldsymbol{z}\| \exp\left(-f(\boldsymbol{z}_*) - \frac{\|\boldsymbol{z}\|^2}{2(1/\rho)}\right) d\boldsymbol{z} \\ = \pi(\boldsymbol{z}_*)\mathbb{E}_{\mathcal{N}(0,\frac{1}{\rho}\boldsymbol{I}_d)}\|\boldsymbol{z}\| \leq \pi(\boldsymbol{z}_*)\sqrt{\mathbb{E}_{\mathcal{N}(0,\frac{1}{\rho}\boldsymbol{I}_d)}\|\boldsymbol{x}\|^2} = \pi(\boldsymbol{z}_*)\sqrt{\frac{d}{\rho}}. \end{aligned} \tag{2.2}$$

[1]The results we state can more generally be stated about an Íto diffusion, namely a stochastic differential equation of the type $d\boldsymbol{x}_t = -g(\boldsymbol{x}_t)dt + \sigma(\boldsymbol{x}_t)dW_t, \ \forall t \geq 0$ satisfying similar regularity conditions to ours. We chose the simplest setting for clarity of exposition.

[2]Recall a function f is L smooth if for any $\boldsymbol{x}, \boldsymbol{y}$ in the support, $f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \langle\nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$.

[3]Milder conditions on f ensure uniqueness/existence of solutions and not essential for our proofs—we assume this for simplicity of exposition.

2.1.1.2 Contrastive learning task

We choose the contrastive task to be binary classification on observations from the diffusion defined in equation 2.1. For $\eta = O_{L_1,L_2,\rho}(1)$—i.e. any η sufficiently small as a function of the regularity parameters of f—we will consider the observations at integer multiples of η, namely $L_{\tilde{X}} := \{\tilde{\boldsymbol{x}}_{i\eta}\} \subset \mathbb{R}^n$, and let $T > 0$ be length of the (continuous-time) sequence covered by these observations. Suppose the number of observations in $L_{\tilde{X}}$ is $2m$ where $2m = \lfloor T/\eta \rfloor$.

The binary classification task is defined on a sequence of pairs of points denoted as $S_X := \{(\boldsymbol{x}_{2i\eta}, \boldsymbol{x}'_{2i\eta})\}_{i=0}^{m-1}$, where $\boldsymbol{x}_{2i\eta} = \tilde{\boldsymbol{x}}_{2i\eta}$, and $\boldsymbol{x}'_{2i\eta}$ is chosen in one of the two ways:

- With probability $1/2$, we let $\boldsymbol{x}'_{2i\eta} = \tilde{\boldsymbol{x}}_{(2i+1)\eta}$ and output $(\boldsymbol{x}_{2i\eta}, \boldsymbol{x}'_{2i\eta})$ with label 1. We call these *positive* pairs.
- With probability $1/2$, we sample $\boldsymbol{x}'_{2i\eta} \sim q$ for some *contrast* proposal distribution q and output $(\boldsymbol{x}_{2i\eta}, \boldsymbol{x}'_{2i\eta})$ with label 0. (We will specify the restrictions on q momentarily.) We call these *negative* pairs.

Intuitively, the task asks the model to distinguish the noise distribution q from the η-time transition kernel of the process $p_*^\eta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, which is defined as

$$p_*^\eta(\boldsymbol{z}, \boldsymbol{z}') := \Pr(\boldsymbol{z}_{(t+1)\eta} = \boldsymbol{z}' | \boldsymbol{z}_{t\eta} = \boldsymbol{z}). \tag{2.3}$$

What we need to assume on the contrast distribution q is that it is sufficiently close to p_*^η (algorithmically, it also needs to have a pdf that is efficient to evaluate). Specifically, define a constant $c_q \geq 1$, such that the ratio between p_*^η and the proposal distribution q is bounded as

$$\frac{1}{c_q} \leq \frac{p_*^\eta(\boldsymbol{z}, \boldsymbol{z}')}{q(\boldsymbol{z}')} \leq c_q, \ \forall \boldsymbol{z}, \boldsymbol{z}'. \tag{2.4}$$

We will show later that a smaller c_q is more preferable, which amounts to choosing a proposal distribution q that closely tracks the data distribution. This is consistent with observations in previous works on noise contrastive learning (NCE) that a closer q makes the contrastive task harder and hence tends to work better in practice [Gutmann and Hyvärinen, 2010b, Gao et al., 2020]. Formally, a larger c_q will give a looser bound on the KL divergence between p^η and p_*^η, as we will see in Theorem 2 and its proof.

The model we use for the supervised task is denoted as $h : \mathbb{R}^{2n} \to \mathbb{R}$, which takes in a $(\boldsymbol{x}, \boldsymbol{x}')$ pair and predicts the probability of the pair being positive. We assume the output of h to be bounded in $[0, 1]$.[4] We denote the function class h belongs to as $\mathcal{H}$, and train h with the ℓ_2 loss:

$$\ell\left(h, \{(\boldsymbol{x}, \boldsymbol{x}'), y\}\right) = \left(h(\boldsymbol{x}, \boldsymbol{x}') - y\right)^2 \tag{2.5}$$

Let the *empirical risk* $\hat{R}$ of a model h and loss ℓ associated with a training set S_X be defined as usual, and taking the expectation over S_X gives the *population risk* R:

$$\begin{aligned} \hat{R}_{S_X}(\ell \circ h) &:= \frac{1}{|S_X|} \sum_{i=1}^{|S_X|} \ell(h, \{(\boldsymbol{x}_{2i\eta}, \boldsymbol{x}'_{2i\eta}), y\}), \\ R(\ell \circ h) &:= \mathbb{E}_{S_X} \hat{R}_{S_X}(\ell \circ h). \end{aligned} \tag{2.6}$$

[4]This can easily be enforced, for example, by having a sigmoid layer at the end of a neural network.

The *generalization gap* is defined as the maximum difference between the above two in the class of classifiers that we consider:

$$\Phi(S_X) := \sup_{h\in\mathcal{H}} \left[\left|\hat{R}_{S_X}(\ell\circ h) - R(\ell\circ h)\right|\right] \tag{2.7}$$

By way of remarks: the l_2 loss is chosen since it is bounded, Lipschitz and strongly-convex, which makes the generalization bound calculations more manageable. It would be interesting to also provide bounds for cross-entropy or other losses.

We also need concepts capturing the complexity of the function class: the *empirical Rademacher complexity* $\hat{\mathfrak{R}}$ of a function class $\mathcal{H}$ is defined with a given dataset S of size m as

$$\hat{\mathfrak{R}}_S(\mathcal{H}) := \frac{1}{m}\mathbb{E}_\varepsilon\left[\sup_{h\in\mathcal{H}}\left|\sum_{i\in[m]}\varepsilon_i h(x_i)\right| |S=(x_1,...,x_m)\right]. \tag{2.8}$$

The *Rademacher complexity* is defined by taking the expectation over the dataset S as

$$\mathfrak{R}_m(\mathcal{H}) := \mathbb{E}_{S:|S|=m}\hat{\mathfrak{R}}_S(\mathcal{H}). \tag{2.9}$$

Assumption 4 (Rademacher Complexity)**.** *We will assume the Rademacher complexity of $\mathcal{H}$ satisfies $\mathfrak{R}_m(\mathcal{H}) = O(K_\eta(\mathcal{H})\sqrt{\log m/m})$, where $K_\eta(\mathcal{H})$ depends on both $\mathcal{H}$ and the task setup η.*

The expression for $\mathfrak{R}_m(\mathcal{H})$ is common in standard generalization bounds. For instance, such dependency is achieved when the square root of the VC dimension of $\mathcal{H}$ is bounded by $K_\eta(\mathcal{H})$ [Mohri et al., 2012].

2.1.1.3 Warmup: characterizing the optimum given infinite data

To gain intuition on what the contrastive task does, we first characterize the optimum of the contrastive learning objective in the limit of infinite data. We note that similar analyses have appeared in other works on variants of contrastive learning, e.g. Hyvarinen and Morioka [2016], Tosh et al. [2020b,d]. We show:

Lemma 1 (Population optimum)**.** *The optimum of the contrastive learning objective*

$$\arg\min_h \mathbb{E}_{((x,x'),y)}\ell\left(h,\{(\boldsymbol{x},\boldsymbol{x}'),y\}\right)$$

satisfies

$$h^*(\boldsymbol{x},\boldsymbol{x}') = \frac{p_*^\eta(\boldsymbol{z},\boldsymbol{z}')}{q(\boldsymbol{z}')+p_*^\eta(\boldsymbol{z},\boldsymbol{z}')}.$$

Proof. The proof proceeds by expanding the expectation in question, and a variance-bias like calculation. Namely, for a fixed (x,x'), taking the expectation over y gives:

$$\mathbb{E}_y\ell\left(h,\{(\boldsymbol{x},\boldsymbol{x}'),y\}\right) = \left(h(\boldsymbol{x},\boldsymbol{x}')-\Pr(y=1|\boldsymbol{x},\boldsymbol{x}')\right)^2 + \Pr(y=1|\boldsymbol{x},\boldsymbol{x}')\left(1-\Pr(y=1|\boldsymbol{x},\boldsymbol{x}')\right). \tag{2.10}$$

The last term of equation 2.10 does not depend on h, so the minimum is achieved when $h^*(\boldsymbol{x}, \boldsymbol{x}') = \Pr(y = 1|\boldsymbol{x}, \boldsymbol{x}')$. Expanding $\Pr(y = 1|\boldsymbol{x}, \boldsymbol{x}')$ by the Bayes rule, we get

$$\begin{aligned} &h^*(\boldsymbol{x}, \boldsymbol{x}') = \Pr(y = 1|\boldsymbol{x}, \boldsymbol{x}') \\ = &\frac{\Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 1)\Pr(y = 1)}{\Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 0)\Pr(y = 0) + \Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 1)\Pr(y = 1)} \\ = &\frac{\Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 1)}{\Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 0) + \Pr(\boldsymbol{x}, \boldsymbol{x}'|y = 1)} \\ = &\frac{\pi(\boldsymbol{z}) p_*^\eta(\boldsymbol{z}, \boldsymbol{z}')}{\pi(\boldsymbol{z}) q(\boldsymbol{z}') + \pi(\boldsymbol{z}) p_*^\eta(\boldsymbol{z}, \boldsymbol{z}')} = \frac{p_*^\eta(\boldsymbol{z}, \boldsymbol{z}')}{q(\boldsymbol{z}') + p_*^\eta(\boldsymbol{z}, \boldsymbol{z}')}. \end{aligned} \tag{2.11}$$

□

Note that the above proof uses essentially nothing about q other than that it is known: this is why population level analyses of contrastive objectives (e.g. like Hyvarinen and Morioka [2016]) may fail to capture many non-asymptotic aspects of the contrastive task.

2.1.1.4 Statement of main results

We are now ready to state the main results of this section. We claim that a low loss on the contrastive task implies closeness in a learned η-time transition kernel and the ground truth one. We will state the main results here and defer the proofs to Section 2.1.3.

2.1.1.4.1 Sample complexity bounds

We first present the sample complexity for controlling the generalization gap defined in equation 2.7:

Theorem 1. *If $T = \Omega\left(\frac{B^2 K_\eta(\mathcal{H})^3}{\delta^2 \Delta_{gen}^3}\left(\log \frac{1}{\delta}\right)^{\frac{3}{2}}\right)$, then with probability $1 - \delta$, the generalization gap is bounded by Δ_{gen}.*

Note that the dependency of T on η comes only through the complexity measure $K_\eta(\mathcal{H})$. The reason there isn't additional dependence on η (e.g. the reader might imagine the number of "samples" effectively depends on T/η) is that though decreasing η gives more samples, the samples will be more dependent and hence less useful for generalization. The proof in Section 2.1.3.1 will formally justify this intuition.

2.1.1.4.2 Distribution estimator from classifier for contrastive task

Second, we show how to prove guarantees on an estimator for the transition kernel, given a classifier with a small contrastive task loss.

In light of Lemma 1, given a classifier h, define the transition kernel implied by h as

$$p^\eta(\boldsymbol{z}, \boldsymbol{z}') := \frac{h(\boldsymbol{z}, \boldsymbol{z}') q(\boldsymbol{z}')}{1 - h(\boldsymbol{z}, \boldsymbol{z}')}. \tag{2.12}$$

We wish to show that if h achieves a small loss, the p^η defined above is in fact close to p^* in some distributional sense.

We will show two types of guarantees, one under the assumption that p^η is somewhat close to p^*, and one for arbitrary p^η. In the first case, we will in fact show that a small loss implies that the learned p^η is close to p^* in a KL divergence sense (more precisely, $\mathbb{E}_x KL(p_*^\eta(\cdot|x)\|p^\eta(\cdot|x))$ is small); in the second case, we will show that $\mathbb{E}_{x,x'}|p_*^\eta(x,x') - p^\eta(x,x')|$ is small.

The reason we can extract a stronger result in the first case is that we can leverage the strong convexity of the contrastive loss near the global optimum in an appropriate sense. Intuitively, in a strongly convex loss, a small loss implies closeness of the parameter to the global optimum. Such a property will not hold globally, as the loss may be arbitrarily non-convex as a function of p^η. Still, we will be able to extract a weaker guarantee (and with a less standard notion of distance).

Case 1: local guarantees for p^η close to p_*^η. Let constants $\Delta_{\min}, \Delta_{\max}$ be defined such that

$$\frac{p^\eta(\boldsymbol{z},\boldsymbol{z}')}{p_*^\eta(\boldsymbol{z},\boldsymbol{z}')} \in [\Delta_{\min}, \Delta_{\max}],\ \forall \boldsymbol{z},\boldsymbol{z}'. \tag{2.13}$$

and note that $0 < \Delta_{\min} \le 1 \le \Delta_{\max}$.

$\Delta_{\min}, \Delta_{\max}$ can be considered as a notion of closeness between p_*^η and p^η. When $\Delta_{\min}, \Delta_{\max}$ are close to 1, that is, when p^η lies in a small neighborhood of p_*^η, we can show the contrastive loss is locally strongly convex with respect to the KL divergence. This allows us to relate the loss to the KL divergence between p_*^η and p^η. Formally, we state the following result:

Theorem 2. *Suppose assumption 1-3 are satisfied and that $\Delta_{\max} \le \frac{7}{6}$. Suppose the training error of h is $\epsilon_{tr} + \epsilon_\star$, where $\epsilon_\star := \mathbb{E}_{\{(\boldsymbol{x},\boldsymbol{x}'),y\}}\left(\frac{p_*^\eta(\boldsymbol{x},\boldsymbol{x}')}{p_*^\eta(\boldsymbol{x},\boldsymbol{x}')+q(\boldsymbol{x}')} - y\right)^2$ is the optimal error achieved by p_*^η. If the generalization gap is bounded by ϵ_{tr}, then the average KL divergence between the ground truth and learned transition kernel is bounded by the contrastive loss as*

$$\mathbb{E}_{\boldsymbol{z}\sim\pi} KL\left(p_*^\eta(\cdot|\boldsymbol{z})\|p^\eta(\cdot|\boldsymbol{z})\right) \le \frac{2(1+c_q)^5\epsilon_{tr}}{\Delta_{\min}^2}. \tag{2.14}$$

Recall that c_q (defined in Equation (2.4)) represents how close the contrast distribution q is to p_*^η. Theorem 2 hence explains why a closer q is more preferable, as has been suggested by empirical evidence [Gao et al., 2020]. In addition, as mentioned earlier, the bound $\Delta_{\max} \le \frac{7}{6}$ is required to reason about the convexity of the contrastive loss in the neighborhood of p_*^η. Globally, the loss need not be convex, so it is entirely possible for a p^η faraway from p_*^η in the KL sense to have a small contrastive loss. Nevertheless, we can prove something weaker in this case.

Case 2: Global guarantees for arbitrary p^η. In the case of an arbitrary p^η, we prove the following bound on the closeness to p_*^η:

Theorem 3. *Under assumptions 1-3, and let ϵ_*, T, p^η be the same as in Theorem 2. Let $\eta = O_{\rho,L_0,L_1}(1)$, then p^η satisfies*

$$\mathop{\mathbb{E}}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\cdot|\boldsymbol{z})}\left|p^\eta(\boldsymbol{z},\boldsymbol{z}') - p_*^\eta(\boldsymbol{z},\boldsymbol{z}')\right| \le \sqrt{2\epsilon_{tr}}\,O\left(\sqrt{\pi(\boldsymbol{z}_*)}\left(\frac{1}{\rho\eta^2}\right)^{d/4}\right). \tag{2.15}$$

We make two remarks about the Theorem 3. First, the value of η cannot be too large for the RHS of Equation (2.15) to obtain (i.e. $\eta = O_{\rho,L_0,L_1}(1)$). Analyzing merely the optimum of the contrastive objective would not reveal this.

Moreover, though the exponential dependency in η may appear pessimistic, it is in fact the right one. A closer inspection of the left hand side of Equation (2.15) shows that its scaling in η is also $(1/\eta)^{d/2}$ (by Lemma 6)—so the only "extra" exponential factors are the η-independent exponential terms. It is not clear if this can be removed or is essential—or if possibly other losses can remove this kind of dependence.

2.1.2 Generalization Machinery for Non-iid Data

At the core of our analysis is a set of tools for non-iid data, which we first build up before discussing the proof. We will use generalization results for data coming from *strong mixing* stochastic processes: namely, the samples are not independent; but, intuitively, after a short amount of time, the samples are "almost independent". Precisely, we use the notion of β-mixing:

Definition 1 (β-mixing)**.** *For a stationary Markov process, the β-mixing coefficient is defined as the average TV distance between the distribution after running the process for t time with a given starting point, and the stationary distribution π:*

$$\beta(t) = \mathbb{E}_{\boldsymbol{x}}\, TV\left(P^t(\cdot|x_0 = \boldsymbol{x}), \pi\right). \tag{2.16}$$

A process is said to be β-mixing if $\lim_{t\to\infty} \beta(t) = 0$.

The β-mixing coefficient of a discrete-time sequence is defined similarly, with the conditional distribution defined between points in the sequence.

We note that β-mixing can be defined more generally on processes that may are not necessarily stationary or Markov. The above definition is the cleanest version that suffices for our setting.

The reason this will be useful for us is that when our data is β-mixing, we will be able to use generalization bounds similar to those we have for iid data. More precisely, we will leverage the following result by Mohri and Rostamizadeh [2009], which when applied to our setting becomes:

Lemma 2 (Rademacher complexity bound, Mohri and Rostamizadeh [2009], Theorem 1)**.** *Let $\mathcal{S}_X$ form a β-mixing sequence with stationary distribution π. Then, for some $\delta \in (0,1)$, for every μ such that $\delta > 2(\mu-1)\beta(T/2\mu)$, with probability at least $1-\delta$, the generalization gap $\Phi(\mathcal{S}_X)$ is bounded by*

$$\Phi(S_X) := \sup_{h\in\mathcal{H}} \left[R(\ell\circ h) - \hat{R}_{S_X}(\ell\circ h)\right] \le \left\{\mathfrak{R}_\mu(\mathcal{H}) + \sqrt{\frac{\log\left(2/\left(\delta - \Delta^\mu_{appr}\right)\right)}{2\mu}}\right\}, \tag{2.17}$$

where $\Delta^\mu_{appr} := 2(\mu-1)\beta_{S_X}(T/2\mu)$, and $\mathfrak{R}_\mu(\mathcal{H})$ is the Rademacher complexity of $\mathcal{H}$ over samples of size μ drawn iid from π.

The result is proved using a technique called *blocking* from Yu [1994]. The idea is to divide a dependent sequence of samples into 2μ blocks of consecutive points, such that when the block size $\frac{T}{2\mu}$ is sufficiently large, every other block would be approximately independent because of the fast mixing.

The generalization analysis can hence be divided into two steps, one for applying standard generalization bound on i.i.d. data (i.e. the blocks), and the other for bounding the approximation error of treating dependent blocks as independent ones. The term Δ^{μ}_{appr} is a consequence of the derivation in Yu [1994] and accounts for errors of approximating non-iid data with iid ones.

We will proceed by first showing fast mixing, then applying the generalization bounds above.

2.1.2.1 Proving β-mixing

We will first show β-mixing of the sequence S_X of pairs $(\boldsymbol{x}, \boldsymbol{x}')$ as constructed in Section 2.1.1.1; that is, by choosing $\boldsymbol{x}$ from the diffusion process, and then choosing $\boldsymbol{x}'$ to be η-time after in the process or from a proposal distribution with equal probability. Intuitively, this would suggest that once two points are sufficiently apart, they will be approximately independent, on which standard generalization bounds apply. Formally, we have the following result:

Lemma 3. *The β-mixing coefficients for the sequence S_X defined in Section 2.1.1.1 is $\beta_{S_X}(t) = O\left(\frac{B}{\sqrt{t}}\right)$.*

Proof. We will prove this by showing the sequence of pairs S_X shares the same β coefficients as the sequence of points $L_{\tilde{X}}$ (Lemma 4). Then, since $L_{\tilde{X}}$ is itself β-mixing (Lemma 5), the claim follows. □

Having the same β coefficients between S_X an $L_{\tilde{X}}$ makes intuitive sense, since the sequence of pairs can be considered as a mixture of a dependent sequence and an independent sequence, and adding the independent one should not worsen the mixing coefficient.

Lemma 4. *$\beta_{S_X}(t) = \beta_{L_{\tilde{X}}}(t)$.*

Proof. First note that S_X is Markov and stationary, since the temporal dependency only comes from the first elements in the pairs, which are points in $L_{\tilde{X}}$ that is itself Markov and stationary:

$$\begin{aligned}
&\Pr\left((\boldsymbol{x}_{2(i+1)\eta}, \boldsymbol{x}'_{2(i+1)\eta}) | (\boldsymbol{x}_0, \boldsymbol{x}'_0), ..., (\boldsymbol{x}_{2i\eta}, \boldsymbol{x}'_{2i\eta})\right) \\
&= \Pr\left(\boldsymbol{x}'_{2(i+1)\eta} | \boldsymbol{x}_{2(i+1)\eta}\right) \Pr\left(\boldsymbol{x}_{2(i+1)\eta} | \boldsymbol{x}_0, ..., \boldsymbol{x}_{2i\eta}\right) \\
&= \Pr(\boldsymbol{x}'_{2(i+1)\eta} | \boldsymbol{x}_{2(i+1)\eta}) \Pr(\tilde{\boldsymbol{x}}_{2(i+1)\eta} | \tilde{\boldsymbol{x}}_{2i\eta}) \\
&= \Pr(\boldsymbol{x}'_{2(i+1)\eta} | \boldsymbol{x}_{2(i+1)\eta}) \Pr(\boldsymbol{x}_{2(i+1)\eta} | \boldsymbol{x}_{2i\eta}).
\end{aligned} \tag{2.18}$$

The mixing coefficient of S_X can then be calculated explicitly, leading to $\beta_{S_X}(2i\eta) = \beta_{L_{\tilde{X}}}(2i\eta)$:

$$\begin{aligned}
&\beta_{S_X}(2i\eta) = \frac{1}{2}\int \Big|\pi(z_0, z_0')\pi(z_{2i\eta}, z_{2i\eta}') \\
&\quad - \pi(z_0, z_0')p(z_{2i\eta}, z_{2i\eta}'|z_0, z_0')\Big| \\
&= \frac{1}{2}\int \pi(z_0, z_0') \cdot \Big|\pi(z_{2i\eta}, z_{2i\eta}') - p(z_{2i\eta}, z_{2i\eta}'|z_0, z_0')\Big| \\
&= \frac{1}{2}\int \left(\frac{1}{2}\pi(z_0)\left(p(z_0'|z_0) + \pi(z_0')\right)\right) \\
&\quad \cdot \left(\frac{1}{2}\big|\pi(z_{2i\eta}) - p(z_{2i\eta}|z_0)\big|\left(p(z_{2i\eta}'|z_{2i\eta}) + \pi(z_{2i\eta}')\right)\right) \\
&= \frac{1}{8}\int_{z_0, z_{2i\eta}} \pi(z_0)\big|\pi(z_{2i\eta}) - p(z_{2i\eta}|z_0)\big| \\
&\quad \cdot \int_{z_0'} \left(p(z_0'|z_0) + \pi(z_0')\right) \cdot \int_{z_{2i\eta}'} \left(p(z_{2i\eta}'|z_{2i\eta}) + \pi(z_{2i\eta}')\right) \\
&= \frac{1}{2}\int_{z_0, z_{2i\eta}} \pi(z_0)\big|\pi(z_{2i\eta}) - p(z_{2i\eta}|z_0)\big| \\
&= \frac{1}{2}\int_{\tilde{z}_0, \tilde{z}_{2i\eta}} \pi(\tilde{z}_0)\big|\pi(\tilde{z}_{2i\eta}) - p(\tilde{z}_{2i\eta}|z_0)\big| = \beta_{L_{\tilde{X}}}(2i\eta).
\end{aligned} \tag{2.19}$$

□

Next, we bound the TV distance, as a function of t, between the stationary distribution π and the distribution after running the diffusion for time t given any starting point:

Lemma 5 (Bubeck et al. [2018], Proposition 4). *Let $B := \mathbb{E}_\pi\|x\|$. For any $t > 0$, $\forall z \in \mathcal{X}$,*

$$TV(\mathbb{P}(z_t|z_0 = z), \pi) \leq \frac{B}{\sqrt{2\pi t}},$$

where $\mathbb{P}(z_t|z_0 = z)$ denotes the distribution after running the diffusion for time t conditioned on being at z at time 0.

With the definition of β-mixing, Lemma 5 shows that $L_{\tilde{X}}$ itself is β-mixing, as long as $B < \infty$, $\beta_{L_{\tilde{X}}}(t\eta) = \text{TV}(\mathbb{P}(z_{t\eta}|z_0 = z), \pi) = O(\frac{1}{\sqrt{t\eta}}) \to 0$ as $t \to \infty$.

2.1.3 Proofs

We are now ready to prove the main results in Section 2.1.1.4. We will start with the finite sample generalization bound, and map the loss on the contrastive task to the KL divergence between the learned and true transition kernels, assuming the former lies in a neighborhood of the latter. We will finish with the proof for Theorem 3 where the closeness assumption is lifted.

2.1.3.1 Proof of the generalization bound

Let's first prove the sample complexity bound for generalization, where we use results in Mohri and Rostamizadeh [2009] to choose the optimal μ to bound the generalization gap.

Proof of Theorem 1. Following notations in Lemma 2, let μ denote the number of "effective" training samples. Substituting in the choice of $T = \Omega\left(\frac{B^2 K_\eta(\mathcal{H})^3}{\delta^2 \Delta_{gen}^3}\left(\log\frac{1}{\delta}\right)^{\frac{3}{2}}\right)$, we have $\Delta_{appr}^{\mu} = O\left(\frac{B}{\sqrt{T}}\mu^{\frac{3}{2}}\right) \le \delta$.

By Lemma 2, and recall the empirical Rademacher complexity is $\mathfrak{R}_\mu = O\left(K_\eta(\mathcal{H})\sqrt{\log\mu/\mu}\right)$, we need to choose T, μ such that

$$C\sqrt{\frac{1}{\mu}}\left(K_\eta(\mathcal{H})\sqrt{\log\mu} + \sqrt{-\log\left(\delta - \Delta_{appr}\right)}\right) \le \Delta_{gen}, \tag{2.20}$$

where $\Delta_{appr}^{\mu} := O\left(\frac{B}{\sqrt{T}}\mu^{\frac{3}{2}}\right)$ by Lemma 3.

We would like to control $\Delta_{appr}^{\mu} = O(\delta)$. Substituting in the choice of $T = \Omega\left(\frac{B^2 K_\eta(\mathcal{H})^3}{\delta^2\Delta_{gen}^3}\left(\log\frac{1}{\delta}\right)^{\frac{3}{2}}\right)$, we have

$$\Delta_{appr}^{\mu} = O\left(\frac{B\delta\Delta_{gen}^{3/2}}{BK_\eta(\mathcal{H})^{3/2}}\left(\log\frac{1}{\delta}\right)^{-3/2}\cdot\mu^{3/2}\right) = O(\delta), \tag{2.21}$$

which is satisfied by setting $\mu = \Theta\left(\frac{K_\eta(\mathcal{H})\sqrt{\log(1/(\delta-\Delta_{appr}))}}{\Delta_{gen}}\right)$.

□

2.1.3.2 Proof of Theorem 2: local guarantees

Theorem 2 states that when p^η is close to p_*^η and the population contrastive loss is not much worse than the optimal value, the KL divergence between p_*^η and p^η is also small. At a high level, with p^η close to p_*^η, we can do a "multiplicative" Taylor expansion of p^η around p_*^η. Then, it can be shown that the second derivative is strictly positive with a proper choice of $\Delta_{\max}$. This is similar in spirit to the notion of strong convexity with respect to KL, from the difference in losses.

Proof of Theorem 2. Recall that in Section 2.1.1.4 we defined constants $\Delta_{\min}, \Delta_{\max}$ such that $\forall \boldsymbol{x}, \boldsymbol{x}'$, $\frac{p^\eta(\boldsymbol{z},\boldsymbol{z}')}{p_*^\eta(\boldsymbol{z},\boldsymbol{z}')} \in [\Delta_{\min}, \Delta_{\max}]$. We can equivalently write this relation as $p^\eta = p_*^\eta(1+\delta)$ with $\delta \in [\Delta_{\min} - 1, \Delta_{\max} - 1]$. By the mean value theorem, $\exists\, \xi \in [\Delta_{\min} - 1, \Delta_{\max} - 1]$, such that

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta}\mathrm{KL}(p_*^\eta \| p^\eta) = \mathbb{E}_\pi\int p_*^\eta \log\frac{p_*^\eta}{p_*^\eta(1+\delta)} \\
&= -\mathbb{E}_\pi\int p_*^\eta \log(1+\delta) = -\mathbb{E}_\pi\int p_*^\eta\left(\delta - \int_{s=0}^{\delta}\frac{1}{2(1+s)^2}s\,ds\right) \\
&\overset{(i)}{=} -\mathbb{E}_\pi\int p_*^\eta(\delta - \frac{1}{2}\frac{1}{(1+\xi)^2}\delta^2) \\
&\overset{(ii)}{\le} \frac{1}{2\Delta_{\min}^2}\mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta}\delta^2.
\end{aligned} \tag{2.22}$$

where (i) applies the mean value theorem to the second order Taylor expansion around 0, and (ii) uses $\int_x p_*^\eta\delta = 0$ since p^η, p_*^η both integrates to 1.

Rewriting the gap between the population loss between $p^\eta(\boldsymbol{z},\boldsymbol{z}')$ and $p_*^\eta(\boldsymbol{z},\boldsymbol{z}')$ as a function of

$\delta(z, z')$, we get:

$$r(\delta) := \left(\frac{p_*^\eta q \delta}{(p_*^\eta(1+\delta) + q)(p_*^\eta + q)} \right)^2, \tag{2.23}$$

where the dependence on z, z' is omitted for clarity.

We would like to lower bound $r''(\delta)$. By the mean value theorem, for any z, z', $\exists\ \xi' \in [\Delta_{\min} - 1, \Delta_{\max} - 1]$,

$$\begin{aligned} r''(\delta) &= r''(0) + r'''(0)\delta + \frac{1}{2} r''''(\xi')\delta^2 \\ &\geq \frac{2(p_*^\eta)^2 q^2}{(p_*^\eta + q)^5} \left((7 - 6\Delta_{\max}) p_*^\eta + q \right) + \frac{12 (p_*^\eta)^5 q^2}{p_*^\eta + q} \cdot \frac{5p_*^\eta + 3q - 2p_*^\eta \Delta_{\max}}{(\Delta_{\max} p_*^\eta + q)^6}. \end{aligned} \tag{2.24}$$

The Taylor series converges when $|\delta| \leq \frac{p_*^\eta + q}{p_*^\eta} = 1 + \frac{q}{p_*^\eta}$, which always holds under our assumption on δ. Moreover, we require $\Delta_{\max} \leq \frac{7}{6}$ to ensure $(7 - 6\Delta_{\max})p_*^\eta + q > 0$. Then,

$$\frac{5p_*^\eta + 3q - 2p_*^\eta \Delta_{\max}}{(\Delta_{\max} p_*^\eta + q)^6} \geq \frac{1}{3} \frac{8p_*^\eta + 9q}{(\frac{1}{6})^6 (7p + 6q)^6} \geq \frac{6^6}{3} \frac{8p_*^\eta + 8q}{(7p_*^\eta + 7q)^6} = \frac{8 \cdot 6^6}{3 \cdot 7^6} \frac{1}{(p_*^\eta + q)^5} \geq \frac{1}{(p_*^\eta + q)^5}. \tag{2.25}$$

Substituting this back to Equation (2.24) gives

$$r''(\delta) \geq \frac{2(p_*^\eta)^2 q^3}{(p_*^\eta + q)^5} + \frac{24 (p_*^\eta)^5 q^2}{(p_*^\eta + q)^6} \geq 2 \cdot \frac{1}{(1 + \frac{q}{p_*^\eta})^2} \cdot \frac{1}{(1 + \frac{p_*^\eta}{q})^3} \geq \frac{2}{(1 + c_q)^5}. \tag{2.26}$$

We can then derive an upper bound on $\mathbb{E}\delta^2$. Recall that in Theorem 2 the gap between the population loss of the learned h and that of h_* is set to be $2\epsilon_{tr}$:

$$\begin{aligned} 2\epsilon_{tr} &= \mathbb{E}_{z \sim \pi, z' \sim \frac{p_*^\eta + q}{2}} \left[r(\delta(z, z')) - r(0) \right] \\ &= \mathbb{E}_{z \sim \pi, z' \sim \frac{p_*^\eta + q}{2}} \int_0^\delta (\delta - t) r''(t) dt \geq \frac{r_{\min}}{2} \mathbb{E}_{z \sim \pi, z' \sim \frac{p_*^\eta + q}{2}} \delta^2 \geq \frac{r_{\min}}{4} \mathbb{E}_{z \sim \pi, z' \sim p_*^\eta} \delta^2, \end{aligned} \tag{2.27}$$

where we denote $r_{\min} := \frac{2}{(1 + c_q)^5}$.

This means $\mathbb{E}_{z \sim \pi, z' \sim p_*^\eta} \delta^2 \leq \frac{8\epsilon_{tr}}{r_{\min}}$. Together with Equation (2.22), we can bound the average KL as

$$\mathbb{E}_{z \sim \pi} \mathrm{KL}\left(p_*^\eta(\cdot|z) \| p^\eta(\cdot|z) \right) \leq \frac{1}{2\Delta_{\min}^2} \mathbb{E}_{\pi, p_*^\eta} \delta^2 \leq \frac{4\epsilon_{tr}}{r_{\min} \Delta_{\min}^2} = \frac{2(1 + c_q)^5 \epsilon_{tr}}{\Delta_{\min}^2}. \tag{2.28}$$

□

2.1.3.3 Proof of Theorem 3: global guarantees

Up to this point, we have reasoned about the generalization gap and the relation between the loss and distributional closeness when p^η is in the proximity of p_*^η. There is one last piece missing: we need to characterize what the value of the loss implies for p^η when its relation to p_*^η is unknown.

This is not an obvious task because the loss guarantees that the squared difference in Equation (2.10) is small *on average* over x, x' according to our data distribution. This does not necessarily imply the squared difference in its numerator, [5] i.e. $\left(p^\eta(\boldsymbol{z},\boldsymbol{z}') - p_*^\eta(\boldsymbol{z},\boldsymbol{z}')\right)^2$, is small. For example, if $q(x') \ll \min\{p^\eta(\boldsymbol{z},\boldsymbol{z}'), p_*^\eta(\boldsymbol{z},\boldsymbol{z}')\}$, then the above difference would be small regardless of the values of $p^\eta(\boldsymbol{z},\boldsymbol{z}'), p_*^\eta(\boldsymbol{z},\boldsymbol{z}')$.

We will leverage the following estimates on the transition kernel of the Langevin diffusion:

Lemma 6 (Gobet [2002], Proposition 1.2). *Under Assumptions 1-3, $\exists\, c, C > 1$, such that*

$$\begin{aligned} p_*^\eta(\boldsymbol{z},\boldsymbol{z}') &\geq \frac{1}{c}\frac{1}{\eta^{d/2}} e^{-C\frac{\|\boldsymbol{z}-\boldsymbol{z}'\|^2}{\eta}} e^{-C\eta\|\boldsymbol{z}\|^2}, \\ p_*^\eta(\boldsymbol{z},\boldsymbol{z}') &\leq c\frac{1}{\eta^{d/2}} e^{-\frac{1}{C}\frac{\|\boldsymbol{z}-\boldsymbol{z}'\|^2}{\eta}} e^{C\eta\|\boldsymbol{z}\|^2}. \end{aligned} \tag{2.29}$$

The theorem in Gobet [2002] holds actually in a substantially more general setting than ours: it only requires that the drift (in our setting ∇f) and diffusion coefficient are in $C^{1+\gamma}, \gamma > 0$.

With this result in mind, as well as the previous lemmas, we are ready to prove Theorem 3:

Proof of Theorem 3. Recall that the optimal solution of the contrastive task satisfies $h^* = \frac{p_*^\eta(\boldsymbol{z},\boldsymbol{z}')}{q(\boldsymbol{z}')+p_*^\eta(\boldsymbol{z},\boldsymbol{z}')}$ by Lemma 1, and that the population contrastive loss is no more than $2\epsilon_{tr}$ over the optimal ϵ_* achieved by h^*. This gives:

$$\begin{aligned} 2\epsilon_{tr} \geq & \mathbb{E}_{\boldsymbol{z}\sim\pi}\mathbb{E}_{\boldsymbol{z}'\sim\frac{1}{2}(p_*^\eta(\boldsymbol{x},\cdot)+q)}\ell(h) \geq \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\pi}\mathbb{E}_{\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}\ell(\hat{h}) \\ &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim\pi}\mathbb{E}_{\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}\left(\frac{q(p^\eta - p_*^\eta)}{(q+p^\eta)(q+p_*^\eta)}\right)^2. \end{aligned} \tag{2.30}$$

We now use the above loss bound to upper bound $\mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}|p^\eta(\boldsymbol{x},\boldsymbol{x}') - p_*^\eta(\boldsymbol{x},\boldsymbol{x}')|$. For notational convenience, we will drop $\boldsymbol{x},\boldsymbol{x}'$ when it is clear from the context.

Define a function r_q with $r_q(p) = \frac{p}{p+q}$. The population risk can now be written as $\mathbb{E}_{\boldsymbol{z},\boldsymbol{z}'}\left(r_q(p^\eta) - r_q(p_*^\eta)\right)^2$. Note that $r_q'(p) = \frac{q}{(p+q)^2}$ and r_q is concave in p, hence

$$r_q'\left(\max\{p^\eta, p_*^\eta\}\right)\cdot\left(p^\eta - p_*^\eta\right) \leq r_q(p^\eta) - r_q(p_*^\eta). \tag{2.31}$$

Using Equation (2.31) and Cauchy-Schwarz, we have

$$\begin{aligned} \mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}|p^\eta - p_*^\eta| &= \mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}\left[\frac{|p^\eta - p_*^\eta| r_q'\left(\max\{p^\eta, p_*^\eta\}\right)}{r_q'\left(\max\{p^\eta, p_*^\eta\}\right)}\right] \\ \leq \mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}&\left[\left(r_q(p^\eta) - r_q(p_*^\eta)\right)\cdot\frac{1}{r_q'\left(\max\{p^\eta, p_*^\eta\}\right)}\right] \\ \leq \sqrt{\mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}\left(r_q(p^\eta) - r_q(p_*^\eta)\right)^2}&\cdot\sqrt{\mathbb{E}_{\boldsymbol{z}\sim\pi,\boldsymbol{z}'\sim p_*^\eta(\boldsymbol{x},\cdot)}\left(\max\{p^\eta, p_*^\eta\} + q\right)^4/q^2}, \end{aligned} \tag{2.32}$$

where the first term is the population risk on the contrastive task, which is bounded by $2\epsilon_{tr}$.

[5] Recall that the squared term in Equation (2.10) can be expanded as $(h(\boldsymbol{x},\boldsymbol{x}') - \Pr(y=1|\boldsymbol{x},\boldsymbol{x}'))^2 = \left(\frac{p}{p+q} - \frac{p_*}{p_*+q}\right)^2 = \frac{(p-p_*)^2q^2}{(p+q)^2(p_*+q)^2}$.

We will proceed to bound the second term. Since p^η, p_*^η are both assumed to satisfy Assumption 1-3, Lemma 6 allows us to bound the quantity of interest in Equation (2.32): let c_*, C_* and $\hat{c}, \hat{C}$ be the constants in Lemma 6 for p_*^η and p^η respectively. Denote $C_u = \max\{C_*, \hat{C}\}$, $C_l = \min\{C_*, \hat{C}\}$. Recall that ρ is the strong convexity constant of f. It can be shown that if η is sufficiently small as a function of these constants (e.g. $\eta = \frac{\rho}{10C_u}$), let $\sigma^2 = \frac{C_l\eta}{2}$, then

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'} \frac{\left(\max\{p^\eta, p_*^\eta\} + q\right)^4}{q^2} \leq O\left(\pi(\boldsymbol{z}_*)\left(\frac{1}{\rho\eta^2}\right)^{d/2}\right). \tag{2.33}$$

The proof applies Lemma 6 and the strong convexity of f to simplify the expression with a Gaussian-integral like calculation. Specifically:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'} \frac{\left(\max\{p^\eta, p_*^\eta\} + q\right)^4}{q^2} \\
\leq &\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'} Z_\sigma^2 \exp\left(\frac{1}{\sigma^2}\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right) \cdot \left(\frac{c}{\eta^{d/2}}\exp(C_u\eta\|\boldsymbol{x}\|^2) + \frac{1}{Z_\sigma}\right)^4 \exp\left(-\frac{4}{C_l\eta}\|\boldsymbol{z}-\boldsymbol{x}'\|^2\right) \\
\leq &\mathbb{E}_{\boldsymbol{x}} Z_\sigma^2 \left(\frac{c}{\eta^{d/2}}\exp(C_u\eta\|\boldsymbol{x}\|^2) + \frac{1}{Z_\sigma}\right)^4 \mathbb{E}_{\boldsymbol{x}'} \exp\left(-\frac{2}{C_l\eta}\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right) \\
\leq &\mathbb{E}_{\boldsymbol{x}} \frac{cZ_\sigma^2}{\eta^{d/2}} \left(\frac{c}{\eta^{d/2}}\exp(C_1\eta\|\boldsymbol{x}\|^2) + (\pi C_l\eta)^{-\frac{d}{2}}\right)^4 \exp\left(C_u\eta\|\boldsymbol{x}\|^2\right) \int_{\boldsymbol{x}'} \exp\left(-\frac{3}{C_l\eta}\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right) \\
\leq &\frac{cZ_\sigma^2}{\eta^{5d/2}} \left(\frac{2\pi C_l\eta}{3}\right)^{\frac{d}{2}} \mathbb{E}_{\boldsymbol{x}} \left(c\exp(C_u\eta\|\boldsymbol{x}\|^2) + (\pi C_l)^{-\frac{d}{2}}\right)^4 \exp\left(C_u\eta\|\boldsymbol{z}\|^2\right) \\
\leq &c\left(\frac{2\pi^3 C_2^3}{\eta^2}\right)^{d/2} \exp(-f(\boldsymbol{x}_*)) \int_{\boldsymbol{x}} \left(c\exp(C_u\eta\|\boldsymbol{x}\|^2) + (\pi C_l)^{-\frac{d}{2}}\right)^4 \exp\left(-\left(\frac{\rho}{2} - C_u\eta\right)\|\boldsymbol{x}\|^2\right) \\
\leq &16c\pi(\boldsymbol{z}_*)\left(\frac{2\pi^3 C_l^3}{\eta^2}\right)^{d/2} \left[c^4\left(\frac{\rho}{2} - 5C_1\eta\right)^{-\frac{d}{2}} + (\pi C_l)^{-2d}\left(\frac{\rho}{2} - C_u\eta\right)^{-\frac{d}{2}}\right] \\
\leq &O\left(\pi(\boldsymbol{z}_*)\left(\frac{1}{\rho\eta^2}\right)^{d/2}\right).
\end{aligned} \tag{2.34}$$

where C_1, C_2 are constants introduced to simplify the notations.

Plugging this inequality back in equation 2.32 gives the statement of the theorem. □

Remark 1. *The proof of theorem Theorem 3 can be adapted straightforwardly to accommodate the boundedness assumptions in theorem Theorem 2, namely, when $\frac{p^\eta}{p_*^\eta}$ and $\frac{p_*^\eta}{q}$ are bounded by $(\Delta_{\min}, \Delta_{\max})$ and $[\frac{1}{c_q}, c_q]$ respectively. In this case, the right hand side of Equation (2.33) will be updated to $(2c_q+1)^4\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}(p_*^\eta)^2$. The exponential dependency in η is however still present, a consequence of Lemma 6.*

2.2 A parameter identifiability view for masked prediction

Self-supervised learning (SSL) is a relatively new approach to unsupervised learning, where the learning algorithm learns to predict auxiliary labels generated automatically from the data without human annotators. The hope is that with a properly designed prediction task, a successfully learned predictor would capture some knowledge about the underlying data. While SSL has been enjoying a rapid growth on the empirical front, theoretical understanding of why and when SSL works is still nascent. In no small part, this is because formalizing the desired guarantees seems challenging. For instance, the focus of SSL has largely been on learning *good features*, which in practice has been quantified by downstream performance on various benchmark datasets [Wang et al., 2018, 2019, Deng et al., 2009, Zhai et al., 2019, Tamkin et al., 2021]. To provide theoretical underpinning to this, one needs to make extra assumptions on the relationship between the self-supervised prediction task and the downstream tasks [Arora et al., 2019, Saunshi et al., 2020, HaoChen et al., 2021, Lee et al., 2021, Wang et al., 2021, Wei et al., 2021, Wen and Li, 2021].

While associating SSL with downstream supervised tasks is a useful perspective and has led to several very interesting theoretical results, we take a step back and revisit a more general goal of SSL, which is to learn some informative functionals of the data distribution. Naturally, the key question here is what functionals should be considered *informative*. While downstream performance is a notable valid choice, in this work, we choose an alternative criterion that is meaningful even without referencing any downstream tasks.

The alternative lens we are interested in is whether the functionals of the data distribution extracted by the SSL predictors can be simply stitched together to obtain the data distribution itself, given additional side-information about the family from which the data distribution is drawn. While this might seem like a tall order, masked prediction based SSL algorithms (which is essentially what pseudo-likelihood corresponds to) have classically been used for learning parametric graphical models such as Ising models [Ravikumar et al., 2010, Bresler, 2015, Vuffray et al., 2016]. But can this be done for broader classes of parametric models?

In this section, we take a preliminary step towards this and ask the question of *parameter identifiability*: assuming the data comes from a ground truth parametric probabilistic model, can common self-supervised tasks uniquely identify the parameters of the ground truth model? More precisely, are the parameters of the model uniquely determined by the optimal predictor for the SSL task (Definition 2)? An appeal of this identifiability perspective is that when a SSL task is sufficient for parameter identifiability, the model parameters can then be recovered straightforwardly from the parameters from the optimal SSL predictor. Parameter identification also has the desirable property of being independent of any downstream task.

A priori, it is unclear whether we can achieve such model parameter identifiability via self-supervised tasks, since it requires recovering the full (parametric) generative model which is arguably more difficult than learning generic latent representations. This work provides a positive answer for broad classes of HMMs: we show that the commonly-used *masked prediction task* [Pathak et al., 2016, Devlin et al., 2018a, He et al., 2021, Lee et al., 2021], wherein a model is trained to predict a masked-out part of a sample given the rest of the sample, can identify the parameters of a HMM. As noted earlier, while such masked prediction for parameter learning has been applied in classical settings such as Ising models [Ravikumar et al., 2010, Bresler, 2015, Vuffray et al., 2016], the HMM setup in this work is more challenging due to the presence of latent variables. HMMs are also more suitable for modeling practical sequential data, and have been commonly adopted in theoretical analyses as a clean proxy for languages [Wei et al., 2021, Xie et al., 2021].

Concretely, the two HMM models we consider in this work are 1) the classic HMM with discrete latent and discrete observables, and 2) a HMM variant with discrete latents and continuous observables that are conditionally Gaussian given the latent, which we abbreviate as G-HMMs. We show that:

- Parameter identifiability is governed by the difficulty of the masked prediction task. The task difficulty is related to the amount of information provided by the combination of the model and the prediction task—where the difficulty can be increased by using a more complicated model, or by predicting more tokens. For instance, predicting the conditional mean of one token given another does not yield identifiability for a discrete HMM (Theorem 4), but does so when data comes from a G-HMM (Theorem 5). Moreover, the identifiability in the latter case quite strongly leverages structural properties of the posterior of the latent variables (Section 2.2.2.1).
- Tools for characterizing the uniqueness of tensor decompositions (e.g., Kruskal's Theorem [Kruskal, 1977, Allman et al., 2009]) can be leveraged to prove identifiability: For both HMM (Theorem 7) and G-HMM (Theorem 8), if we have predictors of the tensor product of tokens (e.g., $\mathbb{E}[x_2 \otimes x_3 | x_1]$), we can use the predictor output to construct a 3-tensor whose rank-1 components are uniquely determined and reveal the parameters of the model.

The rest of this subsection is structured as follows. Section 2.2.1 provides relevant definitions, preliminaries and assumptions. Section 2.2.2 states the main results of this work. Proofs for results on HMMs, including the identifiability proof via tensor decomposition, are provided in Section 2.2.3. Proofs for G-HMM are provided Section 2.2.4.

Related works There have been theoretical analyses on both masked predictions [Lee et al., 2021, Zhang and Hashimoto, 2021] and contrastive methods [Arora et al., 2019, Tosh et al., 2020a,c, Wang and Isola, 2020a, HaoChen et al., 2021, Wen and Li, 2021], with a focus on characterizing the quality of the learned features for downstream tasks [Saunshi et al., 2020, Wei et al., 2021]. These approaches usually rely on quite strong assumptions to tie the self-supervised learning objective to the downstream tasks of interest. In contrast, our work takes the view of parameter identifiability, for which there is no need for downstream assumptions but instead the specific parametric form is key. Note also that while the parameter recovery lens is a new contribution of our work, Wen and Li [2021] argue (as a side-product of their analysis) that some aspects of a generative model are recovered in their setup. Their data model, however, is substantially different from ours and has very different identifiability properties (owing to its basis in sparse coding).

The data generative models that will be described in Section 2.2.1 are special cases of *latent variable models*, which have been widely studied in the literature. One important area of research is independent component analysis (ICA), where the data is assumed to be given as a transformation (mixing) of unknown independent sources which ICA aims to identify. In nonlinear ICA data models, both the sources and the mixing function are generally not identifiable. However, identifiability of the sources can be shown under some additional assumptions (e.g. on the dependency structure of different time steps) [Hyvarinen and Morioka, 2016, 2017a, Hälvä and Hyvarinen, 2020]. Similar ideas have also been applied in the self-supervised setting, where the latent variables can be identified under suitable assumptions on the conditional distribution of the latent [Zimmermann et al., 2021] or on data augmentations [Von Kügelgen et al., 2021]. Unlike our setup though, the mixing function in these models is deterministic and not the object of recovery.

More related to this work is the line of work on learning latent variable models with tensor methods. Specific to learning HMMs, Mossel and Roch [2005] and Anandkumar et al. [2012, 2014] provide algorithms based on third-order moments. A major difference between these prior works on tensor methods and ours is that previous results operate on joint moments, while the results in this work are based on conditional moments given by the optimal predictors for the masked tokens.

2.2.1 Setup: HMM, G-HMM, and the masked prediction task

This work focuses on two classes of latent-variable sequence models. The first are fully discrete hidden Markov models (HMMs), and the second are HMMs whose observables marginally follow a mixtures of Gaussians with identity covariance. We denote the observations and hidden states respectively by $\{x_t\}_{t\geq 1}$ and $\{h_t\}_{t\geq 1}$ for both classes. The hidden states $h_1 \to h_2 \to \cdots$ form a Markov chain, and conditional on h_t, the observable x_t is independent of all other variables. Throughout, we refer to $\{x_t\}_{t\geq 1}$ as tokens, following the nomenclature from language models.

2.2.1.1 Data generative models

Discrete Hidden Markov Model We first describe the parameterization of the standard HMMs with discrete latents and observations. Let $\mathcal{X} := \{1, \ldots, d\} = [d]$ denote the observation space, and let $\mathcal{H} := [k]$ be the state space.[6] The parameters of interest are the *transition matrix* $\boldsymbol{T} \in \mathbb{R}^{k\times k}$ and the *emission matrix* $\boldsymbol{O} \in \mathbb{R}^{d\times k}$, defined in the standard way as

$$P(h_{t+1} = i \mid h_t = j) = \boldsymbol{T}_{ij}, \qquad P(x_t = i \mid h_t = j) = \boldsymbol{O}_{ij}.$$

Conditionally-Gaussian HMM (G-HMM) We next describe the parameterization of *conditionally-Gaussian HMMs (G-HMMs)*. The state space $\mathcal{H} := [k]$ is the same as in the previous case, while the observation space is now continuous with $\mathcal{X} := \mathbb{R}^d$. The parameters of interest are $\boldsymbol{T} \in \mathbb{R}^{k\times k}$, the transition matrix, and $\{\mu_i\}_{i\in[k]} \subset \mathbb{R}^d$, the means of the k identity-covariance Gaussians. Precisely,

$$P(h_{t+1} = i \mid h_t = j) = \boldsymbol{T}_{ij}, \qquad P(x_t = x \mid h_t = i) = (2\pi)^{-\frac{d}{2}} \exp\big(-\|x - \mu_i\|^2/2\big).$$

We use $\boldsymbol{M} := [\mu_1, \ldots, \mu_k] \in \mathbb{R}^{d\times k}$ to denote the matrix whose columns are the Gaussian means.

2.2.1.2 Masked prediction tasks

We are interested in the (regression) task of predicting one or more "masked out" tokens as a function of another observed token, with the goal of minimizing expected squared loss under a distribution given by an HMM or G-HMM (equation 2.35). In the case of the discrete HMMs, we will specifically be predicting the *one-hot encoding vectors* of the observations. Thus, both for HMM and G-HMM, predicting a single token will correspond to predicting a vector. For notational convenience, we will simply associate the discrete states or observations via their one-hot vectors $\{e_1, e_2, \ldots\}$ in the appropriate space and interchangeably write $h = i$ or $h = e_i$, and similarly for x. For the task of predicting the tensor product of (one-hot encoding vectors of) tokens $\otimes_{\tau\in\mathcal{T}} \boldsymbol{x}_\tau$ from another token $\boldsymbol{x}_t$ (where $\mathcal{T}$ is some index set and $t \notin \mathcal{T}$), the optimal predictor with respect to the squared loss

[6]Our results will assume $d \geq k$; see Section 2.2.1.3.

calculates the conditional expectation:

$$f(\boldsymbol{x}_t) = \arg\min_{\tilde{f}} \mathbb{E}_{\{\boldsymbol{x}_\tau\}_{\tau\in\mathcal{T}}} \|\text{vec}(\otimes_{\tau\in\mathcal{T}}\, \boldsymbol{x}_\tau) - \text{vec}(\tilde{f}(\boldsymbol{x}_t))\|_2^2 = \mathbb{E}[\otimes_{\tau\in\mathcal{T}}\, \boldsymbol{x}_\tau \mid \boldsymbol{x}_t] \in (\mathbb{R}^d)^{\otimes|\mathcal{T}|}, \tag{2.35}$$

where "vec" returns the vectorized form of a tensor.

We use the shorthand "$\otimes_{\tau\in\mathcal{T}}\boldsymbol{x}_\tau|x_t$" to refer to this prediction task. For instance, consider the case of predicting x_2 given x_1 under the HMM with parameters $(\boldsymbol{O},\boldsymbol{T})$. The optimal predictor, denoted by $f^{2|1}$, can be written in terms of $(\boldsymbol{O},\boldsymbol{T})$ as [7]

$$\begin{aligned} f^{2|1}(x) &= \mathbb{E}[x_2 \mid x_1 = x] = \sum_{i\in[k]} \mathbb{E}[x_2 \mid h_2 = i] P(h_2 = i \mid x_1 = x) \\ &= \sum_{i\in[k]}\sum_{j\in[k]} \mathbb{E}[x_2 \mid h_2 = i] P(h_2 = i \mid h_1 = j) \underbrace{P(h_1 = j \mid x)}_{:=[\phi(x)]_j} = \sum_{i\in[k]}\sum_{j\in[k]} \boldsymbol{O}_i \boldsymbol{T}_{ij} \underbrace{\frac{\boldsymbol{O}_{x,j}}{\sum_{l\in[k]} \boldsymbol{O}_{x,l}}}_{:=[\phi(x)]_j}. \end{aligned}$$

Here $\phi : \mathbb{R}^d \to \mathbb{R}^k$ denotes the posterior distribution of a hidden state h_t given the corresponding observation x_t, i.e., $\phi(x_t) = \mathbb{E}[h_t \mid x_t]$. [8]

Our goal is to study the parameter identifiability from the prediction tasks, when the predictors have the correct parametric form. Formally, we define identifiability from a prediction task as follows:

Definition 2 (Identifiability from a prediction task, HMM)**.** *A prediction task suffices for identifiability if, for any two HMMs with parameters $(\boldsymbol{O},\boldsymbol{T})$ and $(\tilde{\boldsymbol{O}},\tilde{\boldsymbol{T}})$, equality of their optimal predictors for this task implies that there is a permutation matrix Π such that $\boldsymbol{O} = \tilde{\boldsymbol{O}}\Pi$ and $\boldsymbol{T} = \Pi^\top \tilde{\boldsymbol{T}}\Pi$.*

In other words, the mapping from (the natural equivalence classes of) HMM distributions to optimal predictors for a task is injective, up to a permutation of the hidden state labels. By identifiability from a collection of prediction tasks, we refer to the injectiveness of the mapping from HMM distributions to the collections of optimal predictors for the tasks. Identifiability for G-HMMs is defined analogously with $\boldsymbol{O}, \tilde{\boldsymbol{O}}$ changed to $\boldsymbol{M}, \tilde{\boldsymbol{M}}$.

2.2.1.3 Assumptions

We now state the assumptions used in our results. The first assumption is that the transition matrices of the HMMs are doubly stochastic.

Assumption 5 (Doubly stochastic transitions)**.** *The transition matrix $\boldsymbol{T}$ is doubly stochastic, and the marginal distribution of the initial hidden state h_1 is stationary with respect to $\boldsymbol{T}$.*

This assumption guarantees that the stationary distribution of the latent distribution is uniform for any t, and the transition matrix for the reversed chain is simply $\boldsymbol{T}^\top$. Moreover, this assumption reduces the parameter space and hence will make the non-identifiability results stronger.

We require the following conditions on the parameters for the discrete HMM:

[7] The computation here relies on Assumption 5, given in Section 2.2.1.3.

[8] For discrete HMMs, $\phi(x_t) = \frac{\boldsymbol{O}^\top x_t}{\|\boldsymbol{O}^\top x_t\|_1}$. For GHMMs, $[\phi(x_t)]_i = \frac{\exp\left(-\frac{\|x_t-\mu_i\|_2^2}{2}\right)}{\sum_{j\in[k]}\exp\left(-\frac{\|x_t-\mu_j\|_2^2}{2}\right)}$, $\forall i \in [k]$. ϕ does not need to be indexed by t due to the stationarity assumption in Section 2.2.1.3.

Assumption 6 (Non-redundancy, discrete HMM)**.** *Every row of $\boldsymbol{O}$ is non-zero.*

Assumption 6 can be interpreted as requiring each token to have a non-zero probability of being observed, which is a mild assumption. We also require the following non-degeneracy condition:

Assumption 7 (Non-degeneracy, discrete HMM)**.** $\text{rank}(\boldsymbol{T}) = \text{rank}(\boldsymbol{O}) = k \leq d$.

Note that Assumption 7 only requires the parameters to be non-degenerate, rather than have singular values bounded away from 0. The reason is that this work will focus on population level quantities and make no claims on finite sample behaviors or robustness.

For G-HMM, we similarly require the parameters to be non-degenerate:

Assumption 8 (Non-degeneracy, G-HMM)**.** $\text{rank}(\boldsymbol{T}) = \text{rank}(\boldsymbol{M}) = k \leq d$.

Moreover, we assume that the norms of the means are known and equal:

Assumption 9 (Equal norms of the means)**.** *For each $i \in [k]$, μ_i is a unit vector.*[9]

Assumptions 5-8 are fairly standard [see, e.g., Anandkumar et al., 2012]; in particular, Assumption 7, 8 are required to enable efficient learning, since learning degenerate HMMs can be computationally hard [Mossel and Roch, 2005]. Assumption 9 may be an artifact of our proofs, and it would be interesting to relax in future work.

Our notion of identifiability from a prediction task (or a collection of prediction tasks) will restrict attention to HMMs satisfying Assumptions 5, 6, 7 and G-HMMs satisfying Assumptions 5, 8, 9.

2.2.1.4 Preliminary on the uniqueness of tensor rank decompositions

Some of our identifiability results rely on the uniqueness of *tensor rank-1 decompositions* [Hitchcock, 1927]. An *order-t tensor* (or *t-tensor*) is an t-way multidimensional array; a matrix is a 2-tensor. The *tensor rank* of a tensor $\boldsymbol{W}$ is the minimum number R such that $\boldsymbol{W}$ can be written as a sum of R rank-1 tensors. That is, if a t-tensor $\boldsymbol{W}$ has rank-R, it means that $\boldsymbol{W} = \sum_{i\in[R]} \otimes_{j\in[t]} \boldsymbol{U}_i^{(j)}$ for some matrices $\boldsymbol{U}^{(j)} \in \mathbb{R}^{n_j \times R}$, where $U_i^{(j)}$ denotes the i^{th} column of matrix $\boldsymbol{U}^{(j)}$.

In this work, we only need to work with 3-tensors of the form $\boldsymbol{W} = \sum_{i\in[R]} A_i \otimes B_i \otimes C_i$ for some matrices $\boldsymbol{A} \in \mathbb{R}^{n_1\times R}$, $\boldsymbol{B} \in \mathbb{R}^{n_2\times R}$, $\boldsymbol{C} \in \mathbb{R}^{n_3\times R}$, as 3-tensors will suffice for identifiability in all of our settings of interest.[10] A classic work by Kruskal [1977] gives a sufficient condition under which $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ can be recovered up to column-wise permutation and scaling. The condition is stated in terms of the *Kruskal rank*, which is the maximum number r such that every r columns of the matrix are linearly independent. Let k_A denote the Kruskal rank of matrix $\boldsymbol{A}$, then:

Proposition 1 (Kruskal's theorem, Kruskal [1977])**.** *The components $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ of a 3-tensor $\boldsymbol{W} := \sum_{i\in[R]} \boldsymbol{A}_i \otimes \boldsymbol{B}_i \otimes \boldsymbol{C}_i$ are identifiable up to a shared column-wise permutation and column-wise scaling if $k_A + k_B + k_C \geq 2R + 2$.*

We note that this work focuses on identifiability results rather than providing an algorithm or sample complexity bounds, though the proofs can be adapted into algorithms [see, e.g., Harshman, 1970] under slightly more restrictive conditions (which will be satisfied by all of our identifiability results).

[9] Assumption 9 can be changed to $\|\mu_i\|_2 = c$ for all $i \in [k]$, for any other fixed number $c > 0$.

[10] To apply our results on higher order tensors, one can consider an order-3 slice of the higher order tensor.

2.2.2 Main results on parameter identifiability

We now present the main (non-)identifiability results, and show that the combination of the data generative models and the prediction task directly impacts the sufficiency of identifiability.

2.2.2.1 Pairwise prediction

We begin with the simplest prediction task: namely predicting one token from another, which we refer to as *pairwise prediction tasks*. For HMMs, this task fails to provide parameter identifiability:

Theorem 4 (Nonidentifiability of HMM from predicting $x_t|x_1$)**.** *For any $t \in \mathbb{Z}, t \geq 2$, there exists a pair of HMM distributions with parameters $(\boldsymbol{O}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}})$, each satisfying Assumptions 5, 6 and 7, such that the optimal predictors for the task $x_t|x_1$ are the same under each distribution, but there is no permutation matrix $\Pi \in \mathbb{R}^{k\times k}$ such that $\tilde{\boldsymbol{O}} = \boldsymbol{O}\Pi$ and $\tilde{\boldsymbol{T}} = \Pi^\top \boldsymbol{T}\Pi$ are both satisfied.*

Theorem 4 follows from the fact that the optimal predictor has the form of a product of (stochastic) matrices, and generally, one cannot uniquely recover matrices from their product sans additional conditions [Donoho and Elad, 2003, Candes et al., 2006, Spielman et al., 2012, Arora et al., 2014, Georgiev et al., 2005, Aharon et al., 2006, Cohen and Gillis, 2019]. Specifically, by equation 2.35, the optimal predictor is $f(x_1) = \mathbb{E}[x_t|x_1] = \boldsymbol{O}\boldsymbol{T}^{t-1}\phi(x_1)$ (where $\phi(x_1) := \mathbb{E}[h_1|x_t]$ is the posterior). When $t = 2$, we can find a non-permutation matrix $\boldsymbol{R}$ such that $\tilde{\boldsymbol{O}} = \boldsymbol{O}\boldsymbol{R}$, $\tilde{\boldsymbol{T}} = \boldsymbol{R}^\top \boldsymbol{T}\boldsymbol{R}$ give the same predictor as $\boldsymbol{O}, \boldsymbol{T}$. For $t > 2$, even if $\tilde{\boldsymbol{O}} = \boldsymbol{O}$, we show that the matrix power $\boldsymbol{T}^{t-1}$ is not identifiable:

Claim 1 (Nonidentifiability of matrix powers)**.** *For any $t \in \mathbb{Z}, t \geq 2$, there exist stochastic matrices $\boldsymbol{T}, \tilde{\boldsymbol{T}}$ satisfying Assumption 5, 7, such that $\boldsymbol{T} \neq \tilde{\boldsymbol{T}}$ and $\boldsymbol{T}^t = \tilde{\boldsymbol{T}}^t$.*

On the other hand, pairwise prediction actually *does* suffice for identifiability for G-HMM:

Theorem 5 (Identifiability of G-HMM from predicting $x_2|x_1$)**.** *Under Assumption 5, 8, and 9, if the optimal predictors for the task $x_2|x_1$ under the G-HMM distributions with parameters $(\boldsymbol{M}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}})$ are the same, then $(\boldsymbol{M}, \boldsymbol{T}) = (\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}})$ up to a permutation of the hidden state labels.*

Comparing Theorem 4 and 5 shows that the *specific parametric form* of the generative model matters. Note that HMM and G-HMM have a similar form when conditioning on the latent variable; that is, with $t = 2$, the predictor conditioned on the hidden variable h_2 is $P(x_2|h_2 = i) = \boldsymbol{O}\boldsymbol{T}_i$ for HMM, and $P(x_2|h_2 = i) = \boldsymbol{M}\boldsymbol{T}_i$ for G-HMM. The salient difference between these two setups lies in the posterior function: while the posterior function for HMM is linear in the observable, the posterior function for G-HMM is more complicated and "reveals" more information about the parameter.

To formalize the above intuition, first recall that the GHMM posterior has entries $[\phi(x_t)]_i = \frac{\exp\left(-\frac{\|x_t-\mu_i\|_2^2}{2}\right)}{\sum_{j\in[k]}\exp\left(-\frac{\|x_t-\mu_j\|_2^2}{2}\right)}$, $\forall i \in [k]$. We will show that for G-HMM, even matching the posterior function *nearly* suffices to identify $\boldsymbol{M}$: if $\boldsymbol{M}, \tilde{\boldsymbol{M}}$ parameterize two posterior functions $\phi, \tilde{\phi}$ where $\phi = \tilde{\phi}$, then up to a permutation, $\tilde{\boldsymbol{M}}$ must be equal to either $\boldsymbol{M}$ or a unique (and somewhat special) transformation of $\boldsymbol{M}$. The next step is to further exclude the (special) transformation, which is achieved using the constraint that $T, \tilde{T}$ are stochastic matrices. The first step of the proof sketch is captured by following lemma:

Lemma 7. *For $d \geq k \geq 2$, under Assumption 8, 9, $\phi = \tilde{\phi}$ implies $\tilde{\boldsymbol{M}} = \boldsymbol{M}$ or $\tilde{\boldsymbol{M}} = \boldsymbol{H}\boldsymbol{M}$, where $\boldsymbol{H}$ is a Householder transformation of the form $\boldsymbol{H} := \boldsymbol{I}_d - 2\hat{v}\hat{v}^\top \in \mathbb{R}^{d\times d}$, with $\hat{v} := \frac{(\boldsymbol{M}^\dagger)^\top \boldsymbol{1}}{\sqrt{\boldsymbol{1}^\top \boldsymbol{M}^\dagger (\boldsymbol{M}^\dagger)^\top \boldsymbol{1}}}$.*

To provide some geometric intuition about how $\boldsymbol{H}$ acts on $\boldsymbol{M}$, note that $\hat{v}$ is a unit vector in the column space of $\boldsymbol{M}$ and perpendicular to the affine hull of $\mathcal{A} := \{\mu_i : i \in [k]\}$, which means $\hat{v}^\top \mu_i$ is the same for all $i \in [k]$. As a result, $\tilde{\boldsymbol{M}} = [\tilde{\mu}_1, ..., \tilde{\mu}_k] = [\boldsymbol{H}\mu_1, ..., \boldsymbol{H}\mu_k] = \boldsymbol{M} - 2(\hat{v}^\top \mu_1)[\hat{v}, ..., \hat{v}]$ is a translation of $\boldsymbol{M}$ along the direction of $\hat{v}$, such that the translated points $\{\tilde{\mu}_i\}_{i\in[k]}$ lie on the opposite side of the origin. We can show that $\boldsymbol{H}\boldsymbol{M}$ is the only solution (other than $\boldsymbol{M}$ itself) that preserves ϕ, which we defer to Lemma 8 in Section 2.2.4. It is, however, easy to see that $\boldsymbol{H}\boldsymbol{M}$ indeed results in a matching posterior, whose sufficient conditions are 1) $\tilde{\boldsymbol{M}}$ is a translation of $\boldsymbol{M}$, and 2) $\|\tilde{\mu}_i\|^2 - \|\mu_i\|^2$ is the same for all $i \in [k]$. $\tilde{\boldsymbol{M}} := \boldsymbol{H}\boldsymbol{M}$ indeed satisfies both conditions.

Remark 2. *Another way to think of the difference between the two setups is that for HMM, $P(x_2|x_1)$ is a mixture of categorical distributions, which itself is also a categorical distribution. This also implies that the nonidentifiability from pairwise prediction in the HMM case cannot be resolved by changing the squared loss to another proper loss function. On the other hand, for G-HMM, the conditional distribution $P(x_2|x_1)$ is a mixture of Gaussians, which is well known to be identifiable. In fact, if we were given access to the* entire *conditional distribution $P(x_2|x_1)$ (instead of just the conditional mean), it is even easier to prove identifiability for G-HMM. Though this is already implied from identifiability from the conditional means, we provided a (much simpler) proof in Section 2.2.4.2 assuming access to the full conditional distribution.*

2.2.2.2 Beyond pairwise prediction

The conclusion from Theorem 4 is that a single pairwise prediction task does not suffice for identifiability on HMMs. The next question is then: can we modify the task to obtain identifiability? A natural idea is to force the model to "predict more", and one straightforward way to do so is to combine multiple pairwise prediction tasks. It turns out that this does not resolve the nonidentifiability issue, as we can show that the parameters are not identifiable even when considering *all* possible pairwise tasks involved 3 (adjacent) tokens:

Theorem 6 (Nonidentifiability of HMM from all pairwise predictions on 3 tokens)**.** *There exists a pair of HMM distributions with parameters $(\boldsymbol{O}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}})$, each satisfying Assumptions 5, 6 and 7, and also $\tilde{\boldsymbol{O}} \neq \boldsymbol{O}$, such that, for each of the tasks $x_2|x_1$, $x_1|x_2$, $x_3|x_1$, and $x_1|x_3$, the optimal predictors are the same under each distribution.*[11]

We briefly remark that the reason for only considering adjacent time steps is that when the tokens are at least two time steps apart, matching predictors only matches powers of the transition matrices, which in general does not ensure the transition matrices themselves are matched as shown in Claim 1.

For the intuition of the nonidentifiability result in Theorem 6, recall that the limitation of pairwise predictions on HMMs comes from non-uniqueness of matrix factorization. While adding additional pairwise prediction tasks introduces more equations on the product of matrices, these equations are highly dependent, and the proof works by providing counterexamples that can simultaneously satisfy all these equations.

[11]These 4 pairwise tasks cover all possible pairwise tasks on 3 adjacent tokens. In particular, there is no need to consider $x_2|x_3$ or $x_3|x_2$, since they are the same as $x_1|x_2$ and $x_2|x_1$.

The above intuition leads to another way of forcing the model to "predict more", that is, to increase the number of predicted tokens. The hope is that doing so results in equations on tensors—as opposed to matrices— for which there is a lot of classical machinery delineating tensors for which the rank-1 decomposition is unique, as discussed in Section 2.2.1.4. This intuition proves to be true and we show that increasing the number from 1 to 2 already suffices for identifiability:

Theorem 7 (Identifiability from masked prediction on three tokens, HMM)**.** *Let (t_1, t_2, t_3) be any permutation of $(1, 2, 3)$, and consider the prediction task $x_{t_2} \otimes x_{t_3} | x_{t_1}$. Under Assumption 5, 6, 7, if the optimal predictors under the HMM distributions with parameters $(\boldsymbol{O}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}})$ are the same, then $(\boldsymbol{O}, \boldsymbol{T}) = (\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}})$ up to a permutation of the hidden state labels.*

Compared to prior results on identifiability from third order moments [Allman et al., 2009, Anandkumar et al., 2012, 2014], the difficulty in our setup is that we only have access to the conditional 2-tensors (i.e. matrices) given by the predictors. The proof idea is to construct a third-order tensor by linearly combining the conditional 2-tensors for each possible value of the token being conditioned on, such that Kruskal's theorem applies and gives identifiability. Note, importantly, that the weights for the linear combination cannot depend on the marginal probabilities of the token being conditioned on, since we do not have access to these marginals, and it is unclear whether we could extract unique marginals given the conditional probabilities we are predicting. Thus, the above theorem cannot be simply derived from results showing parameter identifiability from the 3rd order moments.

Note that this tensor decomposition argument can also be applied to G-HMM, with the help of Lemma 7. We leave the details to Theorem 8 in Section 2.2.4.1.

2.2.3 Proofs for HMMs: connection to tensor decomposition

We now discuss proofs for some of the main results. Section 2.2.3.1 proves the identifiability of HMM parameters from the task of predicting two tokens (Theorem 7) using ideas from tensor decomposition, and Section 2.2.4 shows the identifiability proof of pairwise prediction on G-HMM. The rest of the proofs are deferred to the appendix.

2.2.3.1 Proof of Theorem 7: identifiability of predicting two tokens for HMM

There are three cases for the two-token prediction task, i.e. 1) $x_2 \otimes x_3 | x_1$, 2) $x_1 \otimes x_3 | x_2$, and 3) $x_1 \otimes x_2 | x_3$. We will prove for the first two cases, as the third case is proved the same way as the first case by symmetry. In all cases, the idea is to use the predictor to construct a 3-tensor whose components are each of rank-k, so that applying Kruskal's theorem gives identifiability.

Case 1, $x_2 \otimes x_3 | x_1$: $\boldsymbol{O}, \boldsymbol{T}$ and $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ producing the same predictor means $f^{2\otimes 3|1}(x_1) := \mathbb{E}[x_2 \otimes x_3 | x_1] = \tilde{\mathbb{E}}[x_2 \otimes x_3 | x_1] := \tilde{f}^{2\otimes 3|1}(x_1)$, where $\mathbb{E}, \tilde{\mathbb{E}}$ are parameterized by the corresponding parameters. Let

$\mathcal{X} := \{e_i : i \in [d]\}$, and consider the following 3-tensor:

$$\begin{aligned}\boldsymbol{W} &:= \sum_{x_1\in\mathcal{X}} x_1 \otimes \mathbb{E}[x_2 \otimes x_3|x_1] = \sum_{x_1\in\mathcal{X}} x_1 \otimes \mathbb{E}_{h_2|x_1}[\mathbb{E}[x_2\otimes x_3|x_1]|h_2] \\ &= \sum_{i\in[k]}\sum_{x_1\in\mathcal{X}} P(h_2=i|x_1)x_1 \otimes \mathbb{E}[x_2|h_2=i]\otimes\mathbb{E}[x_3|h_2=i] \\ &= \sum_{i\in[k]} \underbrace{\Big(\sum_{x_1\in\mathcal{X}} (\boldsymbol{T}\phi(x_1))^\top e_i^{(k)} x_1\Big)}_{:=a_i} \otimes \boldsymbol{O}_i \otimes (\boldsymbol{OT})_i,\end{aligned} \tag{2.36}$$

where $\boldsymbol{O}_i$ denotes the i^{th} column of $\boldsymbol{O}$, and similarly for $(\boldsymbol{OT})_i$. Note that $\boldsymbol{W}$ can also be written as

$$\boldsymbol{W} = \sum_{x_1\in\mathcal{X}} x_1 \otimes \tilde{\mathbb{E}}[x_2\otimes x_3|x_1] = \sum_{i\in[k]}\Big(\sum_{x_1\in\mathcal{X}} (\tilde{\boldsymbol{T}}\tilde{\phi}(x_1))^\top e_i^{(k)} x_1\Big)\otimes \tilde{\boldsymbol{O}}_i \otimes (\tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}})_i. \tag{2.37}$$

We want to apply Kruskal's theorem for identifiability. In particular, we will show that each component in equation 2.36 forms a matrix of Kruskal rank k. The second and third components clearly satisfy this condition by Assumption 7. For the first component, recall that $\phi(x) = \frac{\boldsymbol{O}^\top x}{\|\boldsymbol{O}^\top x\|_1}$ and write a_i as

$$a_i = \sum_{j\in[d]} \big(\boldsymbol{T}\phi(e_j^{(d)})\big)^\top e_i^{(k)} \cdot e_j^{(d)} = \operatorname{diag}\left([\frac{1}{\|(e_j^{(d)})^\top \boldsymbol{O}\|_1}]_{j\in[d]}\right)\boldsymbol{OT}^\top e_i^{(k)}. \tag{2.38}$$

Putting a_i into a matrix form, we get $\boldsymbol{A} := [a_1, ..., a_k] = \operatorname{diag}\big([1/\|(e_j^{(d)})^\top\boldsymbol{O}\|_1]_{j\in[d]}\big)\boldsymbol{OT}^\top$, [12] which is of rank k by Assumption 7. Hence components $\boldsymbol{W}$ are all of Kruskal rank k, and columns of $\boldsymbol{OT}, \boldsymbol{O}$ are identified up to column-wise permutation and scaling by Kruskal's theorem. The indeterminacy in scaling is further removed noting that columns of $\boldsymbol{O}, \boldsymbol{T}$ need to sum up to 1. Lastly, $\boldsymbol{T}$ is recovered as $\boldsymbol{T} = \boldsymbol{O}^\dagger\boldsymbol{OT}$.

Case 2, $x_1\otimes x_3|x_2$: The optimal predictor for the task of predicting x_1, x_3 given x_2 takes the form

$$\mathbb{E}[x_1\otimes x_3|x_2] = (\boldsymbol{OT}^\top)\operatorname{diag}(\phi(x_2))(\boldsymbol{OT})^\top. \tag{2.39}$$

Similarly as the previous case, we would like to construct a 3-tensor whose components can be uniquely determined by Kruskal's theorem. Let $\mathcal{X}$ be the same as before, and consider the 3-tensor

$$\begin{aligned}\boldsymbol{W} &:= \sum_{x_2\in\mathcal{X}} x_2\otimes\mathbb{E}[x_1\otimes x_3|x_2] = \sum_{x_2\in\mathcal{X}} x_2\otimes\mathbb{E}_{h_2|x_2}(\mathbb{E}[x_1|h_2]\otimes\mathbb{E}[x_3|h_2]) \\ &= \sum_{i\in[k]}\underbrace{\sum_{x_2\in\mathcal{X}}(\phi(x_2))^\top e_i^{(k)} x_2}_{:=a_i}\otimes\mathbb{E}[x_1|h_2]\otimes\mathbb{E}[x_3|h_2] = \sum_{i\in[k]} a_i\otimes(\boldsymbol{OT}^\top)_i\otimes(\boldsymbol{OT})_i,\end{aligned} \tag{2.40}$$

where the first component can be simplified to

$$a_i = \sum_{j\in[d]}\frac{(e_j^{(d)})^\top\boldsymbol{O}}{\|(e_j^{(d)})^\top\boldsymbol{O}\|_1}e_i^{(k)}\cdot e_j^{(d)} = \Big(\operatorname{diag}\big([\|\boldsymbol{O}_j^\top\|_1]_{j\in[d]}\big)\Big)^{-1}\boldsymbol{O}e_i^{(k)} := \boldsymbol{D}^{-1}\boldsymbol{O}e_i^{(k)}. \tag{2.41}$$

The matrix $\boldsymbol{A} := [a_1, ..., a_k] = \boldsymbol{D}^{-1}\boldsymbol{O}$ is of rank k, hence we can identify (up to permutation) columns of each component of $\boldsymbol{W}$ by Kruskal's theorem. This means if $\boldsymbol{O}, \boldsymbol{T}$ and $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ produce the same

[12] We use $[\alpha_i]_{i\in[d]}$ to denote a d-dimensional vector whose i^{th} entry is α_i.

predictor, then we have $\boldsymbol{O}\boldsymbol{T} = \tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}$, $\boldsymbol{O}\boldsymbol{T}^\top = \tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}^\top$, and that $\boldsymbol{O}, \tilde{\boldsymbol{O}}$ are matched up to a scaling of rows (i.e. $\boldsymbol{D}^{-1}$). Next, to determine $\boldsymbol{D}$, note that $\boldsymbol{T}, \tilde{\boldsymbol{T}}$ are doubly stochastic by Assumption 5, which means the all-one vector $\mathbf{1} \in \mathbb{R}^k$ satisfies $\boldsymbol{T}\mathbf{1} = \tilde{\boldsymbol{T}}\mathbf{1} = \mathbf{1}$. Hence $\tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}\mathbf{1} = \boldsymbol{O}\boldsymbol{T}\mathbf{1} = \boldsymbol{O}\mathbf{1} = [\|\boldsymbol{O}_j^\top\|_1]_{j\in[d]}$. We can then compute $\boldsymbol{D}$ as $\boldsymbol{D} = \text{diag}(\boldsymbol{O}\boldsymbol{T}\mathbf{1})$, and recover $\boldsymbol{O}$ as $\boldsymbol{O} = \boldsymbol{D}\boldsymbol{A}$. Finally, $\boldsymbol{T}$ is also recovered since $\tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}} = \boldsymbol{O}\tilde{\boldsymbol{T}} = \boldsymbol{O}\boldsymbol{T} \Rightarrow \tilde{\boldsymbol{T}}\boldsymbol{T}^{-1} = \boldsymbol{I}_k \Rightarrow \tilde{\boldsymbol{T}} = \boldsymbol{T}$.

2.2.3.2 Proof of Theorem 6: non-identifiability of HMM from multiple pairwise predictions

Theorem (Theorem 6 restated)**.** *There exists a pair of HMM distributions with parameters $(\boldsymbol{O}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}})$, each satisfying Assumptions 5, 6 and 7, and also $\tilde{\boldsymbol{O}} \neq \boldsymbol{O}$, such that, for each of the tasks $x_2|x_1$, $x_1|x_2$, $x_3|x_1$, and $x_1|x_3$, the optimal predictors are the same under each distribution.*

Proof. We provide an example to show the nonidentifiability result in Theorem 6. The goal is to find $\tilde{\boldsymbol{O}} \neq \boldsymbol{O}, \tilde{\boldsymbol{T}} \neq \boldsymbol{T}$ that produce the same predictors for predicting both $x_2|x_1$ and $x_3|x_1$. We will choose $\boldsymbol{T}, \tilde{\boldsymbol{T}}$ to be symmetric, so that $\boldsymbol{O}, \boldsymbol{T}$ and $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ also form the same predictors for the reversed direction, i.e., for predicting x_1 given x_2 and x_1 given x_3, since the reverse chain has transition matrix $\boldsymbol{T}^\top = \boldsymbol{T}$.

Let's consider the case where the all row sums of $\boldsymbol{O}$ and $\tilde{\boldsymbol{O}}$ are k/d. Consequently, the posterior function is simply $\phi(x) = \frac{\boldsymbol{O}^\top x}{\|\boldsymbol{O}^\top x\|_1} = \frac{d}{k}\boldsymbol{O}^\top x$, and similarly we have $\tilde{\phi}(x) = \frac{d}{k}\tilde{\boldsymbol{O}}^\top x$. The predictors are of the form:

$$f^{2|1}(x) = \boldsymbol{O}\boldsymbol{T}\phi(x) = \frac{d}{k}\boldsymbol{O}\boldsymbol{T}\boldsymbol{O}^\top x,\ f^{3|1}(x) = \boldsymbol{O}\boldsymbol{T}^2\phi(x) = \frac{d}{k}\boldsymbol{O}\boldsymbol{T}^2\boldsymbol{O}^\top x. \tag{2.42}$$

Matching $f^{2|1}(x) = \tilde{f}^{2|1}(x)$ on all $x \in \mathcal{X} := \{e_i\}_{i\in[d]}$ means

$$\boldsymbol{O}\boldsymbol{T}\boldsymbol{O}^\top \boldsymbol{I}_d = \boldsymbol{O}\boldsymbol{T}\boldsymbol{O}^\top = \tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}\tilde{\boldsymbol{O}}^\top \Rightarrow \tilde{\boldsymbol{T}} = \tilde{\boldsymbol{O}}^\dagger\boldsymbol{O}\cdot\boldsymbol{T}\cdot(\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O})^\top. \tag{2.43}$$

Similarly, matching $f^{3|1} = \tilde{f}^{3|1}$ gives $\boldsymbol{O}\boldsymbol{T}^2\boldsymbol{O}^\top = \tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}^2\tilde{\boldsymbol{O}}^\top$, hence

$$\begin{aligned}\tilde{\boldsymbol{O}}\tilde{\boldsymbol{T}}^2\tilde{\boldsymbol{O}}^\top &= \tilde{\boldsymbol{O}}\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O}\boldsymbol{T}(\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O})^\top\cdot\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O}\boldsymbol{T}(\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O})^\top\tilde{\boldsymbol{O}}^\top \\ &\overset{(i)}{=} \boldsymbol{O}\boldsymbol{T}\cdot(\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O})^\top\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O}\cdot\boldsymbol{T}\boldsymbol{O}^\top = \boldsymbol{O}\boldsymbol{T}\cdot\boldsymbol{T}\boldsymbol{O} \Rightarrow (\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O})^\top\cdot\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O} = \boldsymbol{I}_k,\end{aligned} \tag{2.44}$$

where step (i) uses $\tilde{\boldsymbol{O}}\tilde{\boldsymbol{O}}^\dagger\boldsymbol{O} = \boldsymbol{O}$, since $\tilde{\boldsymbol{O}}, \boldsymbol{O}$ share the same column space.

Denote $\boldsymbol{R} := \tilde{\boldsymbol{O}}^\dagger\boldsymbol{O}$; $\boldsymbol{R}$ is orthogonal by the last equality in equation 2.44. To construct the desired example, consider $k = 3$, and let $\boldsymbol{R}$ represent a rotation with axis of rotation $\frac{1}{3}(e_1 + e_2 + e_3)$. This axis is the direction pointing from the origin to the projection of the origin on the hyperplane $\mathcal{P}_c := \{v \in \mathbb{R}^d : \sum_{i\in[d]} v_i = c\}$ for any positive constant c (i.e. $\mathcal{P}_c$ is parallel to the hyperplane in which probability vectors lie). This means such rotation guarantees $\boldsymbol{R}v \in \mathcal{P}_c, \forall v \in \mathcal{P}_c$, and has the following property:

Claim 2. *Each row and each column of $\boldsymbol{R}$ sums up to 1.*

Proof. We would like to show that each row and each column of $\boldsymbol{R}$ sums up to 1. Denote the d-dimensional simplex by Δ_d, i.e. $\Delta_d := \{x \in \mathbb{R}^d : \sum_{i\in[d]} x_i = 1\}$, and let $\mathcal{P}_c := \{v \in \mathbb{R}^d : \sum_{i\in[d]} v_i = c\}$ for some positive constant c denote a hyperplane parallel to the hyperplane in which probability vectors lie.

Let's first check that the columns of $\boldsymbol{R}$ sum up to 1. Any $v \in \mathcal{P}_c$ can be written as $v = c \cdot [\alpha_1, \alpha_2, ..., \alpha_{d-1}, 1 - \sum_{i\in[d-1]} \alpha_i]$ for some $[\alpha_1, ..., \alpha_{d-1}] \in \Delta_{d-1}$. Let r_i denote the i_{th} row of $\boldsymbol{R}$, then $\boldsymbol{R}v \in \mathcal{P}_c$ means $\sum_{i\in[d]} \langle r_i, v\rangle = \langle \sum_{i\in[d]} r_i, v\rangle = c$. Let β_j denote the j_{th} coordinate of $\sum_{i\in[d]} r_i$, then

$$\begin{aligned}
&\sum_{i\in[d-1]} \beta_i\alpha_i + \beta_d(1 - \sum_{i\in[d-1]} \alpha_i) = 1,\ \forall[\alpha_1, ..., \alpha_{d-1}] \in \Delta_{d-1} \\
\Rightarrow &\sum_{i\in[d-1]} (\beta_i - \beta_d)\alpha_i + \beta_d = 1,\ \forall[\alpha_1, ..., \alpha_{d-1}] \in \Delta_{d-1} \\
\Rightarrow &\ \beta_i = 1,\ \forall i \in [d].
\end{aligned} \tag{2.45}$$

It then follows that $\boldsymbol{R}^{-1} = \boldsymbol{R}^\top$ also has columns summing up to 1, since

$$\sum_{i\in[d]} (\boldsymbol{R}\boldsymbol{R}^{-1})_{ij} = \langle \sum_{i\in[d]} r_i, (\boldsymbol{R}^{-1})_j\rangle = \langle \mathbf{1}, (\boldsymbol{R}^{-1})_j\rangle = 1,\ \forall j \in [d]. \tag{2.46}$$

□

Define $\tilde{\boldsymbol{O}} := \boldsymbol{O}\boldsymbol{R}$, $\tilde{\boldsymbol{T}} := \boldsymbol{R}^\top\boldsymbol{T}\boldsymbol{R}$, Claim 2 ensures that row sum and column sum of $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ remain the same as those of $\boldsymbol{O}, \boldsymbol{T}$. When the rotation angle represented by $\boldsymbol{R}$ is sufficiently small, entries $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ remain in $[0, 1]$, hence such $\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{T}}$ form a valid example. We will provide a concrete example in the subsequent subsection.

□

An example of nonindentifiability The intuition of the nonidentifiability result in Theorem 6 is related to the non-uniqueness of matrix factorization: while adding additional pairwise prediction tasks introduces more equations on the product of matrices, these equations can be highly dependent, and there are cases where different set of matrices can simultaneously satisfy all the equations.

We now provide a concrete example for the non-identifiability of predicting $x_2|x_1$, $x_1|x_2$, $x_3|x_1$, and $x_1|x_3$, by finding 2 set of $\boldsymbol{O}, \boldsymbol{T}$ such that the corresponding predictors (of the form specified in equation 2.42) match. Let $d = 4, k = 3$,

$$\begin{aligned}
&\boldsymbol{O} = \begin{bmatrix} 0.23016003 & 0.3549092 & 0.16493077 \\ 0.30716059 & 0.06962305 & 0.37321636 \\ 0.2580854 & 0.26965425 & 0.22226035 \\ 0.20459398 & 0.3058135 & 0.23959252 \end{bmatrix}, \tilde{\boldsymbol{O}} = \begin{bmatrix} 0.24120928 & 0.35062535 & 0.15816537 \\ 0.28937626 & 0.07433156 & 0.38629218 \\ 0.26077674 & 0.26749114 & 0.22173212 \\ 0.20863772 & 0.30755194 & 0.23381033 \end{bmatrix}, \\
&\boldsymbol{T} = \begin{bmatrix} 0.56893146 & 0.35811118 & 0.07295736 \\ 0.35811118 & 0.10805638 & 0.53383243 \\ 0.07295736 & 0.53383243 & 0.39321021 \end{bmatrix}, \tilde{\boldsymbol{T}} = \begin{bmatrix} 0.59740926 & 0.30452087 & 0.09806987 \\ 0.30452087 & 0.1331689 & 0.56231024 \\ 0.09806987 & 0.56231024 & 0.33961989 \end{bmatrix}, \\
&\det(\boldsymbol{O}) = \det(\tilde{\boldsymbol{O}}) = 0.0110, \det(\boldsymbol{T}) = \det(\tilde{\boldsymbol{T}}) = -0.1611.
\end{aligned} \tag{2.47}$$

Note that $\boldsymbol{T}, \tilde{\boldsymbol{T}}$ are both symmetric as desired by the proof of Theorem 6, which means this is also a valid counter example for learning to predict $x_1|x_2$ and $x_1|x_3$, and hence for all of $x_2|x_1$, $x_1|x_2$, $x_3|x_1$, and $x_1|x_3$.

2.2.3.3 Nonidentifiability from large time gaps

As noted earlier, there is an inherent obstacle when using prediction tasks on tokens that are more than 1 time gaps apart. For instance, if we are predicting x_{t+1} given x_1 for some $t > 1$ with G-HMM, then we are still able to identify $\boldsymbol{M}$ from the posterior function, however it remains to to recover $\boldsymbol{T}$ from $\boldsymbol{T}^t$. For general matrices, it is clear that matching a power of a matrix does not imply the matrix itself is matched. For our case, even though requiring $\boldsymbol{T}$ to be stochastic adds additional constraints, matching the matrix power still does not suffice to identify the underlying matrix, as formalized in the following claim.

Claim 3 (Nonidentifiability of matrix powers (Claim 1 restated))**.** *For any positive integer t, there exist stochastic matrices $\boldsymbol{T}, \tilde{\boldsymbol{T}}$ satisfying Assumption 5, 7, such that $\boldsymbol{T} \neq \tilde{\boldsymbol{T}}$ and $\boldsymbol{T}^t = \tilde{\boldsymbol{T}}^t$.*

Proof. As in Theorem 6, the nonidentifiability comes from the non-uniqueness of matrix factorization. Specifically for this case, we will set $\tilde{\boldsymbol{T}}$ to be equal to $\boldsymbol{T}$ up to a special rotation that gets composed when taking the matrix power. That is, we want $\tilde{\boldsymbol{T}} = \boldsymbol{R}\boldsymbol{T} = \boldsymbol{T}\boldsymbol{R}$ for some matrix $\boldsymbol{R}$ that implicitly performs a rotation, so that $\tilde{\boldsymbol{T}}^t = \boldsymbol{T}^t\boldsymbol{R}^t$. Since $\boldsymbol{R}$ corresonds to a rotation, we can choose the rotation angle properly so that $\boldsymbol{R}^t = \boldsymbol{I}$, and hence $\tilde{\boldsymbol{T}}^t = \boldsymbol{T}^t$ but $\tilde{\boldsymbol{T}} \neq \boldsymbol{T}$.

Precisely, using notations for the G-HMM setup, set $a \in [0, 1]$, and let the parameters $(\boldsymbol{T}, \boldsymbol{M})$ be given by

$$\boldsymbol{T} = \begin{bmatrix} a & 0 & 1-a \\ 1-a & a & 0 \\ 0 & 1-a & a \end{bmatrix}, \boldsymbol{M} = \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & -\sqrt{3}/2 & \sqrt{3}/2 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}.$$

Let θ be some rotation angle, and denote by $\boldsymbol{R}(\theta) := \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ a rotation that acts on the first two dimensions. We will show that for any $\theta \in \mathbb{R}$, we have

$$\tilde{\boldsymbol{T}} := \left(\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\theta)\right)^{-1}\boldsymbol{M}\right) \cdot \boldsymbol{T} = \boldsymbol{T} \cdot \left(\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\theta)\right)^{-1}\boldsymbol{M}\right). \tag{2.48}$$

Assuming equation 2.48, since $\boldsymbol{R}(\theta)$ represents a rotation of angle θ, $\left(\boldsymbol{R}(\theta)\right)^{\tau}$ corresponds to a rotation of angle $\tau\theta$ for any integer τ (τ could be negative). Setting $\theta := \frac{2\pi}{t}$, we then have

$$\begin{aligned} \tilde{\boldsymbol{T}}^t =& \left(\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\theta)\right)^{-1}\boldsymbol{M} \cdot \boldsymbol{T}\right)^t = \boldsymbol{T}^t\left(\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\theta)\right)^{-1}\boldsymbol{M}\right)^t = \boldsymbol{T}^t\boldsymbol{M}^{-1}\left(\boldsymbol{R}(\theta)\right)^{-t}\boldsymbol{M} \\ =& \boldsymbol{T}^t\boldsymbol{M}^{-1} \cdot \boldsymbol{R}(2\pi) \cdot \boldsymbol{M} = \boldsymbol{T}^t. \end{aligned} \tag{2.49}$$

For $\tilde{\boldsymbol{T}}$ to serve as a valid example for our theorem, it remains to check that for every t, there exists a choice of a such that $\tilde{\boldsymbol{T}} := \boldsymbol{R}\boldsymbol{T}$, where $\boldsymbol{R} := \boldsymbol{M}^{-1}\left(\boldsymbol{R}(\frac{2\pi}{t})\right)^{-1}\boldsymbol{M}$, is a valid stochastic matrix. That is, $\tilde{\boldsymbol{T}}$ has 1) columns and rows each summing up to 1, and 2) entries bounded in $[0, 1]$. Let's first show that the columns and rows each sum up to 1. Noting that $\boldsymbol{M}^{-1} = \frac{1}{3}\begin{bmatrix} 2 & 0 & \sqrt{2} \\ -1 & -\sqrt{3} & \sqrt{2} \\ -1 & \sqrt{3} & \sqrt{2} \end{bmatrix}$, the column sums are

$$\boldsymbol{1}^\top\tilde{\boldsymbol{T}} = \boldsymbol{1}^\top\boldsymbol{M}^{-1}\boldsymbol{R}(\theta)^{-1}\boldsymbol{M}\boldsymbol{T} \overset{(i)}{=} \boldsymbol{1}^\top\boldsymbol{T}\boldsymbol{M}^{-1}\boldsymbol{R}(\theta)^{-1}\boldsymbol{M} = \sqrt{2}e_3^\top\boldsymbol{R}(\theta)\boldsymbol{M} = \sqrt{2}e_3^\top\boldsymbol{M} = \sqrt{2}\frac{1}{\sqrt{2}}\boldsymbol{1} = \boldsymbol{1}, \tag{2.50}$$

where step (i) uses equation 2.48. Similarly, the row sums are

$$\tilde{\boldsymbol{T}}\mathbf{1} = \boldsymbol{M}^{-1}\boldsymbol{R}(\theta)^{-1}\boldsymbol{M}\mathbf{1} = \boldsymbol{M}^{-1}\boldsymbol{R}(\theta)^{-1}\cdot\frac{3}{\sqrt{2}}\boldsymbol{e}_3 = \boldsymbol{M}^{-1}\cdot\frac{3}{\sqrt{2}}\boldsymbol{e}_3 = \mathbf{1}. \tag{2.51}$$

To show that there exists a choice of $\boldsymbol{T}$ such that entries of $\tilde{\boldsymbol{T}}$ are non-negative, we provide a concrete example where $\boldsymbol{T}$ is defined with $a = \frac{1}{2}$. It can be checked that $\tilde{\boldsymbol{T}} := \boldsymbol{M}^{-1}(\boldsymbol{R}(\frac{2\pi}{t}))^{-1}\boldsymbol{M}$ has non-negative entries for $t \in \{2, 3, 4, ..., 10\}$. For larger t, let $\theta = \frac{2\pi}{t}$, then we have by the Taylor expansion of $\boldsymbol{R}(\frac{2\pi}{t})$:

$$\begin{aligned}\boldsymbol{R}(\theta) :=& \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1-\theta^2/2+c_1\theta^4 & -\theta+c_2\theta^2 & 0\\ \theta+c_2\theta^2 & 1-\theta^2/2+c_1\theta^4 & 0\\ 0 & 0 & 1\end{bmatrix}\\ =&\boldsymbol{I}+\theta\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}+\theta^2\begin{bmatrix}-1/2+c_1\theta^2 & c_2 & 0\\ c_2 & -1/2+c_1\theta^2 & 0\\ 0 & 0 & 0\end{bmatrix}\end{aligned} \tag{2.52}$$

for some constants $c_1 \in [-\frac{1}{4!}, \frac{1}{4!}]$, $c_2 \in [-\frac{1}{2}, \frac{1}{2}]$. Substituting this into $\tilde{\boldsymbol{T}} := \boldsymbol{M}^{-1}\boldsymbol{R}(\theta)^{-1}\boldsymbol{M}$ gives

$$\begin{aligned}\tilde{\boldsymbol{T}} =& \begin{bmatrix}a-\frac{\theta}{\sqrt{3}}(1-a) & \frac{\theta}{\sqrt{3}}(1-2a) & 1-a+\frac{\theta}{\sqrt{3}}a\\ 1-a+\frac{\theta}{\sqrt{3}}a & a-\frac{\theta}{\sqrt{3}}(1-a) & \frac{\theta}{\sqrt{3}}(1-2a)\\ \frac{\theta}{\sqrt{3}}(1-2a) & 1-a+\frac{\theta}{\sqrt{3}}a & a-\frac{\theta}{\sqrt{3}}(1-a)\end{bmatrix}\\ &+\frac{1}{3}\begin{bmatrix}-1+2c_1\theta^2 & \frac{1}{2}-c_1\theta^2-\sqrt{3}c_2 & \frac{1}{2}-c_1\theta^2+\sqrt{3}c_2\\ \frac{1}{2}-c_1\theta^2-\sqrt{3}c_2 & -1+2c_1\theta^2+\sqrt{3}c_2 & \frac{1}{2}-c_1\theta^2\\ \frac{1}{2}-c_1\theta^2+\sqrt{3}c_2 & \frac{1}{2}-c_1\theta^2 & -1+2c_1\theta^2-\sqrt{3}c_2\end{bmatrix}\cdot\begin{bmatrix}a & 0 & 1-a\\ 1-a & a & 0\\ 0 & 1-a & a\end{bmatrix}\\ =&\frac{1}{2}\begin{bmatrix}1-\frac{\theta}{\sqrt{3}} & 0 & 1+\frac{\theta}{\sqrt{3}}\\ 1+\frac{\theta}{\sqrt{3}} & 1-\frac{\theta}{\sqrt{3}} & 0\\ 0 & 1+\frac{\theta}{\sqrt{3}} & 1-\frac{\theta}{\sqrt{3}}\end{bmatrix}+\frac{\theta^2}{6}\begin{bmatrix}-\frac{1}{2}+c_1\theta^2-\sqrt{3}c_2 & 1-2c_1\theta^2 & -\frac{1}{2}+c_1\theta^2+\sqrt{3}c_2\\ -\frac{1}{2}+c_1\theta^2 & -\frac{1}{2}+c_1\theta^2+\sqrt{3}c_2 & 1-2c_1\theta^2-\sqrt{3}c_2\\ 1-2c_1\theta^2+\sqrt{3}c_2 & -\frac{1}{2}+c_1\theta^2-\sqrt{3}c_2 & -\frac{1}{2}+c_1\theta^2\end{bmatrix}\\ \overset{(i)}{\geq}&\frac{1}{2}\begin{bmatrix}1-\frac{\theta}{\sqrt{3}} & 0 & 1+\frac{\theta}{\sqrt{3}}\\ 1+\frac{\theta}{\sqrt{3}} & 1-\frac{\theta}{\sqrt{3}} & 0\\ 0 & 1+\frac{\theta}{\sqrt{3}} & 1-\frac{\theta}{\sqrt{3}}\end{bmatrix}+\theta^2\begin{bmatrix}-0.25 & -0.16 & -0.25\\ -0.09 & -0.25 & 0.01\\ 0.01 & -0.25 & -0.09\end{bmatrix}\end{aligned} \tag{2.53}$$

where the inequality (i) is taken entry-wise. It can be checked that all entries are non-negative for $\theta \leq \frac{2\pi}{10}$.

Proof of equation 2.48 Let's conclude the proof by proving the commutativity in equation 2.48. Denote $\boldsymbol{R}_2(\theta) := \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}$, i.e. $\boldsymbol{R}(\theta) = \begin{bmatrix}\boldsymbol{R}_2(\theta) & 0\\ 0 & 1\end{bmatrix}$. Denote $\boldsymbol{U} := \begin{bmatrix}1 & -1/2 & -1/2\\ 0 & -\sqrt{3}/2 & \sqrt{3}/2\end{bmatrix}$, i.e. $\boldsymbol{M} = \begin{bmatrix}\boldsymbol{U}\\ \mathbf{1}^\top/\sqrt{2}\end{bmatrix}$. We can write

$$\boldsymbol{M}^\top\boldsymbol{R}(\theta)^\top\boldsymbol{M} = \begin{bmatrix}\boldsymbol{U}^\top & \mathbf{1}/\sqrt{2}\end{bmatrix}\begin{bmatrix}\boldsymbol{R}_2(\theta)^\top & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{U}\\ \mathbf{1}^\top/\sqrt{2}\end{bmatrix} = \boldsymbol{U}^\top\boldsymbol{R}_2(\theta)^\top\boldsymbol{U}+\frac{\mathbf{1}\mathbf{1}^\top}{2}. \tag{2.54}$$

Let $\boldsymbol{R}_2(\theta)$ denote a clockwise rotation of angle θ, then

$$\boldsymbol{U} = [v_1, \boldsymbol{R}_2(\frac{2\pi}{3})v_1, \boldsymbol{R}_2(\frac{4\pi}{3})v_1] = [\boldsymbol{R}_2(\frac{4\pi}{3})v_2, v_2, \boldsymbol{R}_2(\frac{2\pi}{3})v_2] = [\boldsymbol{R}_2(\frac{2\pi}{3})v_3, \boldsymbol{R}_2(\frac{4\pi}{3})v_3, v_3], \tag{2.55}$$

where $v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $v_2 = \begin{bmatrix} -1/2 \\ -\sqrt{3}/2 \end{bmatrix}$, $v_3 = \begin{bmatrix} -1/2 \\ \sqrt{3}/2 \end{bmatrix}$. Denote $\alpha_{ij} := v_i^\top \boldsymbol{R}_2^\top v_j$ for $i, j \in [3]$. Noting $\boldsymbol{T} = a\boldsymbol{I} + (1-a)\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} := a\boldsymbol{I} + (1-a)\boldsymbol{P}$, we have

$$\begin{aligned} & \boldsymbol{M}^\top \boldsymbol{R}(\theta)^\top \boldsymbol{M}\boldsymbol{T} = \boldsymbol{T}\boldsymbol{M}^\top \boldsymbol{R}(\theta)^\top \boldsymbol{M} \\ \Leftrightarrow & \boldsymbol{U}^\top \boldsymbol{R}_2(\theta)^\top \boldsymbol{U}(a\boldsymbol{I} + (1-a)\boldsymbol{P}) + \frac{\boldsymbol{1}\boldsymbol{1}^\top}{2}\boldsymbol{T} = (a\boldsymbol{I} + (1-a)\boldsymbol{P})\boldsymbol{U}^\top \boldsymbol{R}_2(\theta)^\top \boldsymbol{U} + \boldsymbol{T}\frac{\boldsymbol{1}\boldsymbol{1}^\top}{2} \\ \overset{(i)}{\Leftrightarrow} & \boldsymbol{U}^\top \boldsymbol{R}_2(\theta)^\top \boldsymbol{U}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{U}^\top \boldsymbol{R}_2(\theta)^\top \boldsymbol{U} \\ \Leftrightarrow & \begin{bmatrix} \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \end{bmatrix} \overset{(*)}{=} \begin{bmatrix} \alpha_{12} & \alpha_{13} & \alpha_{11} \\ \alpha_{22} & \alpha_{23} & \alpha_{21} \\ \alpha_{32} & \alpha_{33} & \alpha_{31} \end{bmatrix}. \end{aligned} \tag{2.56}$$

where step (i) uses $\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{T} = \boldsymbol{T}\boldsymbol{1}\boldsymbol{1}^\top = \boldsymbol{1}\boldsymbol{1}^\top$. The equality $(*)$ is true due to equation 2.55. □

2.2.4 Proofs for G-HMM: a geometric argument

Proof of Theorem 5: identifiability of predicting x_2 given x_1 for G-HMM We first prove Theorem 5, and then the helper lemmas.

Proof of Theorem 5. For G-HMM, the predictor for x_2 given x_1 is parameterized as $f^{2|1}(x_1) = \mathbb{E}[x_2|x_1] = \boldsymbol{M}\boldsymbol{T}\phi(x_1)$. If $\boldsymbol{M}, \boldsymbol{T}$ and $\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}}$ produce the same predictor, then

$$f^{2|1}(x) = \boldsymbol{M}\boldsymbol{T}\phi(x) = \tilde{\boldsymbol{M}}\tilde{\boldsymbol{T}}\tilde{\phi}(x) = \tilde{f}^{2|1}(x), \; \forall x \in \mathbb{R}^d. \tag{2.57}$$

Let $\boldsymbol{R} := (\tilde{\boldsymbol{M}}\tilde{\boldsymbol{T}})^\dagger(\boldsymbol{M}\boldsymbol{T}) \in \mathbb{R}^{k \times k}$, then $\tilde{\phi}(x) = \boldsymbol{R}\phi(x)$. The following lemma (proved at the end of this section) says that $\phi, \tilde{\phi}$ must then be equal up to a permutation of coordinates:

Lemma 8. *If there exists a non-singular matrix $\boldsymbol{R} \in \mathbb{R}^{k \times k}$ such that $\phi(x) = \boldsymbol{R}\tilde{\phi}(x), \forall x \in \mathbb{R}^d$, then $\boldsymbol{R}$ must be a permutation matrix.*

Combined with Lemma 7, we have $\tilde{\boldsymbol{M}}$ is equal to (up to a permutation) either $\boldsymbol{M}$ or $\boldsymbol{H}\boldsymbol{M}$, where $\boldsymbol{H}$ is the Householder reflection given in Lemma 7.

The remaining step is to show that $\boldsymbol{H}\boldsymbol{M}$ can be ruled out by requiring $\tilde{\boldsymbol{T}}$ to be a stochastic matrix. Note that matching both the predictor and the posterior function means we also have $\tilde{\boldsymbol{M}}\tilde{\boldsymbol{T}} = \boldsymbol{M}\boldsymbol{T}$, or $\tilde{\boldsymbol{T}} = (\tilde{\boldsymbol{M}}^\dagger\boldsymbol{M})\boldsymbol{T}$. Recall that $\boldsymbol{H} := \boldsymbol{I}_d - 2\hat{v}\hat{v}^\top$ for $\hat{v} = \frac{(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}}{\sqrt{\boldsymbol{1}^\top\boldsymbol{M}^\dagger(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}}}$. When $\tilde{\boldsymbol{M}} = \boldsymbol{H}\tilde{\boldsymbol{M}}$, the column sum of $\tilde{\boldsymbol{M}}^\dagger\boldsymbol{M}$ is

$$\begin{aligned} & \boldsymbol{1}^\top\tilde{\boldsymbol{M}}^\dagger\boldsymbol{M} = \boldsymbol{1}^\top\boldsymbol{M}^\dagger\boldsymbol{H}^{-1}\boldsymbol{M} = \boldsymbol{1}^\top\boldsymbol{M}^\dagger(\boldsymbol{I} - 2\hat{v}\hat{v}^\top)\boldsymbol{M} = \boldsymbol{1}^\top(\boldsymbol{I} - 2\boldsymbol{M}^\dagger\hat{v}\hat{v}^\top\boldsymbol{M}) \\ = & \boldsymbol{1}^\top - 2\cdot\boldsymbol{1}^\top\frac{\boldsymbol{M}^\dagger(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{M}^\dagger\boldsymbol{M}}{\boldsymbol{1}^\top\boldsymbol{M}^\dagger(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}} = \boldsymbol{1}^\top - 2\cdot\frac{\boldsymbol{1}^\top\boldsymbol{M}^\dagger(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}}{\boldsymbol{1}^\top\boldsymbol{M}^\dagger(\boldsymbol{M}^\dagger)^\top\boldsymbol{1}}\boldsymbol{1}^\top = \boldsymbol{1}^\top - 2\cdot\boldsymbol{1}^\top = -\boldsymbol{1}^\top. \end{aligned} \tag{2.58}$$

This means the column sum of $\tilde{\boldsymbol{T}}$ is $\boldsymbol{1}^\top\tilde{\boldsymbol{T}} = \boldsymbol{1}^\top(\tilde{\boldsymbol{M}}^\dagger\boldsymbol{M})\boldsymbol{T} = -\boldsymbol{1}^\top\boldsymbol{T} = -\boldsymbol{1}^\top$, which violates the constraint that $\tilde{\boldsymbol{T}}$ should be a stochastic matrix with positive entries and column sum 1. Hence it must be that $\boldsymbol{M} = \tilde{\boldsymbol{M}}$ and hence also $\boldsymbol{T} = \tilde{\boldsymbol{T}}$ (up to permutation), proving the theorem statement.

□

We now prove the two helper lemmas.

Proof of Lemma 7. Lemma 7 states that if $\boldsymbol{M}, \tilde{\boldsymbol{M}}$ parameterize $\phi, \tilde{\phi}$ respectively and that $\phi = \tilde{\phi}$, then $\tilde{\boldsymbol{M}}$ must equal to either $\boldsymbol{M}$ or a unique (and as we will see, somewhat special) transformation of $\boldsymbol{M}$.

To prove this, let's start with the case where $d = k$. First, let's check the conditions for $\phi = \tilde{\phi}$. For any $x \in \mathbb{R}^d$, we have

$$\begin{aligned}
&[\phi(x)]_i = \frac{\exp\left(-\frac{\|x-\mu_i\|^2}{2}\right)}{\sum_{j\in[k]}\exp\left(-\frac{\|x-\mu_j\|^2}{2}\right)} = \frac{\exp\left(-\frac{\|x-\tilde{\mu}_i\|^2}{2}\right)}{\sum_{j\in[k]}\exp\left(-\frac{\|x-\tilde{\mu}_j\|^2}{2}\right)} = [\tilde{\phi}(x)]_i, \ \forall i \in [k] \\
&\Rightarrow \frac{\exp\left(-\frac{\|x-\mu_i\|^2}{2}\right)}{\exp\left(-\frac{\|x-\tilde{\mu}_i\|^2}{2}\right)} = \frac{\exp\left(-\frac{\|x-\mu_j\|^2}{2}\right)}{\exp\left(-\frac{\|x-\tilde{\mu}_j\|^2}{2}\right)}, \forall i, j \in [k] \\
&\Rightarrow \|x-\mu_i\|^2 - \|x-\tilde{\mu}_i\|^2 = \|x-\mu_j\|^2 - \|x-\tilde{\mu}_j\|^2, \ \forall i, j \in [k] \\
&\Rightarrow 2\big((\tilde{\mu}_i - \mu_i) - (\tilde{\mu}_j - \mu_j)\big)^\top x = \big(\|\mu_j\|^2 - \|\tilde{\mu}_j\|^2\big) - \big(\|\mu_i\|^2 - \|\tilde{\mu}_i\|^2\big), \ \forall i, j \in [k].
\end{aligned} \tag{2.59}$$

Since the left hand side is linear in $x \in \mathbb{R}^d$ and the right hand side is a constant, it must be that both sides are 0. That is, the necessary conditions for $\phi = \tilde{\phi}$ are that for any $i, j \in [k]$, 1) $\tilde{\mu}_i - \mu_i = \tilde{\mu}_j - \mu_j$, and 2) $\|\mu_i\|^2 - \|\tilde{\mu}_i\|^2 = \|\mu_j\|^2 - \|\tilde{\mu}_j\|^2$. It can be checked that these two conditions are also sufficient for $\phi = \tilde{\phi}$.

Denote $v := \mu_i - \tilde{\mu}_i$. The norms of the means are known and equal by Assumption 9, which gives

$$\|\mu_i\|^2 - \|\tilde{\mu}_i\|^2 = \|\mu_i\|^2 - \|\mu_i - v\|^2 = (2\mu_i - v)^\top v = 0, \ \forall i \in [k]. \tag{2.60}$$

The last equality in equation 2.60 holds for a non-zero v when the span of $\{2\mu_i - v : i \in [k]\}$ is $(d-1)$-dimensional subspace. On the other hand, the span of $\{2\mu_i - v : i \in [k]\}$ is at least $(k-1)$ by Assumption 8. When $d = k$, it must be that the dimension is exactly $(d-1)$, which means v is an affine combination of $\{2\mu_i : i \in [k]\}$ by Claim 5.

Moreover, v has to be orthogonal to $\{2\mu_i - v : i \in [k]\}$, which leads to the unique choice of v that is the projection of the origin onto the $(d-1)$-dimensional subspace specified by the affine combinations of $\{2\mu_i : i \in [k]\}$.

Claim 4. *v is the projection of the origin to the hyperplane defined by $\{2\mu_i : i \in [k]\}$, and is the only solution to equation 2.60.*

Proof. It is clear that this choice of v satisfies $(2\mu_i - v)^\top v = 0, \forall i \in [k]$. To see that this is the unique choice, suppose there exists some v' lying in the hyperplane of $\{2\mu_i\}$, and denote $\delta := v' - v$.

Note that $\delta^\top v = 0$: let the hyperplane specified by $\{2\mu_i\}_{i\in[k]}$ be specified as $\{x : \langle \boldsymbol{u}, x\rangle = c\}$ for some $\boldsymbol{u} \in \mathbb{R}^d$ and $c \in \mathbb{R}$. Then v, the projection of the origin, can be written as $v = \frac{c}{\|\boldsymbol{u}\|} \cdot \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$, i.e. v is proportional to the normal vector $\boldsymbol{u}$. For any v' in the hyperplane, it satisfy $\langle \boldsymbol{u}, v'\rangle = c$, and

$$\begin{aligned}
\delta^\top v =& (v' - v)^\top v = \langle \frac{c}{\|\boldsymbol{u}\|}\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, v'\rangle - \left\|\frac{c}{\|\boldsymbol{u}\|}\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\right\|^2 \\
=& \frac{c}{\|\boldsymbol{u}\|^2} \cdot \langle \boldsymbol{u}, v'\rangle - \frac{c^2}{\|\boldsymbol{u}\|^2}\frac{\|\boldsymbol{u}\|^2}{\|\boldsymbol{u}\|^2} = \frac{c^2}{\|\boldsymbol{u}\|^2} - \frac{c^2}{\|\boldsymbol{u}\|^2} = 0.
\end{aligned} \tag{2.61}$$

Then for any v' satisfying equation 2.60,

$$\begin{aligned}(2\mu_i - v')^\top v' &= (2\mu_i - v - \delta)^\top (v+\delta) \\ &= \underbrace{(2\mu_i - v)^\top v}_{0} + 2\mu_i^\top \delta - \underbrace{v^\top \delta}_{0} - \underbrace{\delta^\top v}_{0} - \delta^\top \delta = (2\mu_i - \delta)^\top \delta = 0,\ \forall i \in [k].\end{aligned} \tag{2.62}$$

Since $\{2\mu_i - \delta\}_{i\in[k]}$ spans the $(k-1)$-dimensional hyperplane and that δ lies in the hyperplane, it must be that $\delta = 0$, i.e. $v' = v$. □

Note that this choice of v also satisfies $\|\mu_i - v\| = \|\mu_i\|$, since v and the origin are reflections w.r.t. the hyperplane that is the affine hull of $\{\mu_i : i \in [k]\}$. In other words, $\{\mu_i - v\}_{i\in[k]}$ is related to $\{\mu_i\}_{i\in[k]}$ via the Householder transformation of the form $\boldsymbol{H} := \boldsymbol{I}_d - 2\frac{vv^\top}{\|v\|^2}$, i.e. $\mu_i - v = \boldsymbol{H}\mu_i$. Denote $\hat{v} := \frac{v}{\|v\|_2}$. An explicit formula for $\hat{v}$ is $\hat{v} := \frac{\boldsymbol{M}^{-\top}\boldsymbol{1}}{\sqrt{\boldsymbol{1}^\top \boldsymbol{M}^{-1}\boldsymbol{M}^{-\top}\boldsymbol{1}}}$. This finishes the proof for $d = k$.

For $d > k$, the above argument still applies and $\boldsymbol{H}$ remains the only indeterminacy (up to permutation), where $\boldsymbol{H} := \boldsymbol{I}_d - 2\hat{v}\hat{v}^\top$ for $\hat{v} := \frac{(\boldsymbol{M}^\dagger)^\top \boldsymbol{1}}{\sqrt{\boldsymbol{1}^\top \boldsymbol{M}^\dagger (\boldsymbol{M}^\dagger)^\top \boldsymbol{1}}}$. The reason is that even though the ambient dimension d is larger, $\{\mu_i - v : i \in [k]\}$ has to have the same span as $\{\mu_i : i \in [k]\}$, since having the same predictor requires the column space of $\boldsymbol{M}, \tilde{\boldsymbol{M}}$ to match. Hence we only need to consider v in the k-dimensional column space of $\boldsymbol{M}$, which reduces to the case of $d = k$.

□

Proof for Lemma 8. Given the form of the predictor, matching two predictors $f, \tilde{f}$ means that the corresponding posteriors $\phi, \tilde{\phi}$ are matched up to a linear transformation. Lemma 8 states that such linear transformation must be a permutation.

Let's start by matching the Jacobian w.r.t. x on both sides. Recall the Jacobian of the posterior vector $\phi(x) \in \mathbb{R}^k$, where $[\phi(x)]_i = \frac{\exp(-\frac{\|x-\mu_i\|^2}{2})}{\sum_{j\in[k]}\exp(-\frac{\|x-\mu_j\|^2}{2})}$. Denote $o(x) := \left[-\frac{\|x-\mu_1\|^2}{2}, ..., -\frac{\|x-\mu_k\|^2}{2}\right] \in \mathbb{R}^k$, then $\nabla_x \phi(x) = \nabla_{o(x)}\text{softmax}(o(x)) \cdot \nabla_x o(x)$, where

$$\begin{aligned}\nabla_o [\text{softmax}(o)]_i &= [\text{softmax}(o)]_i \cdot (e_i - \text{softmax}(o)) = [\phi(x)]_i \cdot (e_i - \phi(x)), \\ \nabla_o \text{softmax}(o) &= \text{diag}(\phi(x)) - \phi(x)\phi(x)^\top, \\ \nabla_x o(x) &= -\left[x-\mu_1, ..., x-\mu_k\right]^\top.\end{aligned} \tag{2.63}$$

Hence the Jacobian is

$$\nabla_x \phi(x) = \left(\text{diag}(\phi(x)) - \phi(x)\phi(x)^\top\right) \cdot (\boldsymbol{M} - [x, x, ..., x])^\top. \tag{2.64}$$

Denote $\Delta := \boldsymbol{M} - [x, x, ..., x] \in \mathbb{R}^{d\times k}$, and similarly $\tilde{\Delta} = \tilde{\boldsymbol{M}} - [x, x, ..., x]$. Matching $\nabla_x \tilde{\phi}(x) = \nabla_x \boldsymbol{R}\phi(x)$ gives

$$\text{diag}(\boldsymbol{R}\phi(x))\tilde{\Delta}^\top - \boldsymbol{R}\phi(x)(\tilde{\Delta}\boldsymbol{R}\phi(x))^\top = \boldsymbol{R}\text{diag}(\phi(x))\Delta^\top - \boldsymbol{R}\phi(x)(\Delta\phi(x))^\top. \tag{2.65}$$

Let's take $x = x_c^{(i)} := c\mu_i$ for $c > 1$. We claim that this $x_c^{(i)}$ satisfies $\lim_{c\to\infty} \phi(x_c^{(i)}) \to e_i$. This is because $\forall j \neq i$,

$$\begin{aligned}
&\lim_{c\to\infty} \frac{[\phi(x_c^{(i)})]_j}{[\phi(x_c^{(i)})]_i} = \lim_{c\to\infty} \exp\Big(\frac{\|c\mu_i - \mu_i\|^2}{2} - \frac{\|c\mu_i - \mu_j\|^2}{2}\Big) \\
&= \lim_{c\to\infty} \exp\Big(-\frac{\big((2c-1)\mu_i - \mu_j\big)^\top(\mu_i - \mu_j)}{2}\Big) = \lim_{c\to\infty} \exp\Big(-\frac{2c\mu_i^\top(\mu_i-\mu_j)}{2}\Big) = 0
\end{aligned} \tag{2.66}$$

where the last equality is because $\mu_i^\top(\mu_i - \mu_j) > 0$ for any μ_i, μ_j lying on the same hypersphere.

With such choices of x, the two sides of equation 2.65 are now:

$$\begin{aligned}
&LHS = \operatorname{diag}(\boldsymbol{R}_i)\tilde{\Delta}^\top - \boldsymbol{R}_i\boldsymbol{R}_i^\top\tilde{\Delta}^\top = (\operatorname{diag}(\boldsymbol{R}_i) - \boldsymbol{R}_i\boldsymbol{R}_i^\top)\tilde{\Delta}^\top \\
&= RHS = \sum_{j\in[k]} [e_i]_j \boldsymbol{R}_j(\Delta_j)^\top - \boldsymbol{R}e_i(\Delta e_i)^\top = \boldsymbol{R}_i(\Delta_i)^\top - \boldsymbol{R}_i(\Delta_i)^\top = 0.
\end{aligned} \tag{2.67}$$

Since $x := c\mu_i$ for $c \to \infty$ lies outside the affine hull of $\{\tilde{\mu}_i\}_{i\in[k]}$, $\tilde{\Delta}$ is of full rank due to the following claim:

Claim 5. *Given a linearly independent set $\{u_i\}_{i\in[k]}$, if $\{u_i - v\}_{i\in[k]}$ is not linearly independent, then $v = \sum_{i\in[k]} \beta_i \cdot u_i$ where $\sum_{i\in[k]} \beta_i = 1$.*

Proof. Since $\{u_i - v\}_{i\in[k]}$ is linearly dependent, we can write some $u_j - v$ as the linear combination of other $\{u_i - v\}_{i\in[k], i\neq j}$. Let's take $j = k$ wlog, and denote the coefficients of the linear combination as $\{\alpha_i\}_{i\in[k-1]}$. Then

$$u_k - v = \sum_{i\in[k-1]} \alpha_i(u_i - v) \Rightarrow \big(1 - \sum_{i\in[k-1]} \alpha_i\big)v = -\sum_{i\in[k-1]} \alpha_i \cdot u_i + u_k \tag{2.68}$$

The right hand side is non-zero since $\{u_i\}_{i\in[k]}$ are linearly independent by assumption, hence $1 - \sum_{i\in[k-1]} \alpha_i \neq 0$, and we get

$$v = \sum_{i\in[k-1]} \underbrace{\frac{-\alpha_i}{1 - \sum_{i\in[k-1]}\alpha_i}}_{:=\beta_i} \cdot u_i + \underbrace{\frac{1}{1 - \sum_{i\in[k-1]}\alpha_i}}_{:=\beta_k} u_k. \tag{2.69}$$

Note that $\sum_{i\in[k]} \beta_i = 1$, hence v is an affine combination of $\{u_i : i \in [k]\}$. □

Since $\tilde{\Delta}$ is full rank, it must be $\operatorname{diag}(\boldsymbol{R}_i) - \boldsymbol{R}_i\boldsymbol{R}_i^\top = 0$, which implies $\boldsymbol{R}$ is a permutation matrix. This is because for any non-zero v s.t. $\operatorname{diag}(v) - vv^\top = 0$, the entries of v satisfy $v_i^2 = 1$, $v_iv_j = 0$ for $i \neq j$. Hence v has exactly one non-zero entry which is ± 1. Since $\boldsymbol{R}\phi(x) = \tilde{\phi}(x)$ where $\phi(x), \tilde{\phi}(x)$ are both probability vectors with non-negative entries, this non-zero entry has to be 1 (and not -1). Since $\boldsymbol{R}$ is of rank-k by Assumption 8, this non-zero entry is at different positions for different $\boldsymbol{R}_i$, hence $\boldsymbol{R}$ is a permutation matrix.

□

2.2.4.1 Identifiability of predicting $x_{t_2} \otimes x_{t_3} | x_{t_1}$, G-HMM

Theorem 7 shows that triplet prediction tasks (i.e. predict 2 tokens given 1) suffices for the identifiability of HMM, using tools from the uniqueness of tensor decomposition. The next theorem shows that the same conclusion also applies for G-HMM:

Theorem 8 (Identifiability from masked prediction on three tokens, G-HMM)**.** *Let (t_1, t_2, t_3) be any permutation of $(1, 2, 3)$, and consider the prediction task $x_{t_2} \otimes x_{t_3} | x_{t_1}$. Under Assumption 5, 8, 9, if the optimal predictors under the G-HMM distributions with parameters $(\boldsymbol{M}, \boldsymbol{T})$ and $(\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}})$ are the same, then $(\boldsymbol{M}, \boldsymbol{T}) = (\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}})$ up to a permutation of the hidden state labels.*

Proof. Similar to the discrete case, we will prove $x_2 \otimes x_3 | x_1$ and $x_1 \otimes x_3 | x_2$ separately; the proof for $x_1 \otimes x_2 | x_3$ is analogous to $x_2 \otimes x_3 | x_1$ by symmetry and hence omitted. The proofs also follow a similar strategy as in the proof for Theorem 7, that is, to construct a 3-tensor using the predictor, on which applying Kruskal's theorem provides identifiability.

Case 1, $x_2 \otimes x_3 | x_1$: Let $\mathcal{X} := \{x^{(i)} \in \mathbb{R}^d : i \in [k]\}$ be a linearly independent set, and consider the following 3-tensor:

$$\begin{aligned}
\boldsymbol{W} :=& \sum_{x_i \in \mathcal{X}} x_1 \otimes \mathbb{E}[x_2 \otimes x_3 | x_1] = \sum_{x_1 \in \mathcal{X}} x_1 \otimes \mathbb{E}_{h_2|x_1}[\mathbb{E}[x_2 \otimes x_3 | x_1] | h_2] \\
=& \sum_{x_1 \in \mathcal{X}} x_1 \otimes \sum_{h_2} P(h_2|x_1) \mathbb{E}[x_2|h_2] \otimes \mathbb{E}[x_3|h_2] \\
=& \sum_{i \in [k]} \sum_{x_1 \in \mathcal{X}} P(h_2 = i | x_1) x_1 \otimes \mathbb{E}[x_2 | h_2 = i] \otimes \mathbb{E}[x_3 | h_2 = i] \\
=& \sum_{i \in [k]} \underbrace{\Big(\sum_{x_1} (\boldsymbol{T}\phi(x_1))^\top e_i^{(k)} x_1 \Big)}_{:= a_i} \otimes \boldsymbol{M}_i \otimes (\boldsymbol{M}\boldsymbol{T})_i.
\end{aligned} \tag{2.70}$$

The matrices formed by second and third components are both of rank-k by Assumption 8. Hence in order to apply Kruskal's theorem on $\boldsymbol{W}$, it suffices to show that there exists a choice of $\mathcal{X}$ such that the matrix $\boldsymbol{A} := [a_1, ..., a_k]$ is of rank k. One such choice is to let $x^{(i)} = \mu_i$, which gives

$$\begin{aligned}
a_i :=& \sum_{j \in [k]} \phi(x_1 = \mu_j)^\top \boldsymbol{T}^\top e_i^{(k)} \mu_j = \boldsymbol{M}[\phi(\mu_1), ..., \phi(\mu_k)]^\top \boldsymbol{T}^\top e_i^{(k)}, \\
\boldsymbol{A} :=& [a_1, ..., a_k] = \boldsymbol{M}[\phi(\mu_1), ..., \phi(\mu_k)]^\top \boldsymbol{T}^\top.
\end{aligned} \tag{2.71}$$

Since $\boldsymbol{M}$, $\boldsymbol{T}$ are both of rank k by Assumption 8, we only need to argue that the matrix $\Phi := [\phi(\mu_1), ..., \phi(\mu_k)] \in \mathbb{R}^{k \times k}$ is of full rank. Recall that for a mixture of k Gaussians with identify covariance and mean $\{\mu_i \in \mathbb{R}^d : i \in [k]\}$, the posterior function ϕ is defined entrywise as

$$[\phi(x)]_i = \frac{\exp\big(-\frac{\|x-\mu_i\|_2^2}{2}\big)}{\sum_{j \in [k]} \exp\big(-\frac{\|x-\mu_j\|_2^2}{2}\big)}, \ \forall i \in [k]. \tag{2.72}$$

To show Φ is of full rank, we can equivalently show that a columnwise scaled version of Φ is full rank. In particular, let's look at the matrix $\hat{\Phi} \in \mathbb{R}^{k \times k}$, where $\hat{\Phi}_{ij} = \exp(-\frac{\|\mu_i - \mu_j\|^2}{2})$; that is, each column of $\hat{\Phi}$ can be considered as a scaled version of the column in Φ without the normalization for a unit ℓ_1 norm. It can be seen that $\hat{\Phi}$ is a Gaussian kernel matrix which is known to be full rank.

Therefore we have shown that each component of the tensor $\boldsymbol{W} := \sum_{i\in[k]} a_i \otimes \boldsymbol{M}_i \otimes (\boldsymbol{MT})_i$ has Kruskal rank k, which allows to recover columns of $\boldsymbol{M}, \boldsymbol{MT}$ up to permutation and scaling by Kruskal's theorem. The indeterminacy in scaling is further removed since the norms of $\{\boldsymbol{M}_i\}_{i\in[d]}$ are known by Assumption 9.

On the other hand, for any $\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}}$ that form the same predictor as $\boldsymbol{M}, \boldsymbol{T}, \boldsymbol{W}$ can also be written as

$$\begin{aligned} \boldsymbol{W} &= \sum_{x_1\in\mathcal{X}} x_1 \otimes \mathbb{E}[x_2 \otimes x_3|x_1] = \sum_{x_1\in\mathcal{X}} x_1 \otimes \tilde{\mathbb{E}}[x_2\otimes x_3|x_1] \\ &= \sum_{i\in[k]} \Big(\sum_{x_1} (\tilde{\boldsymbol{T}}\tilde{\phi}(x_1))^\top e_i^{(k)} x_1 \Big) \otimes \tilde{\boldsymbol{M}}_i \otimes (\tilde{\boldsymbol{M}}\tilde{\boldsymbol{T}})_i. \end{aligned} \tag{2.73}$$

Hence columns of $\boldsymbol{M}, \tilde{\boldsymbol{M}}$ and $\boldsymbol{MT}, \tilde{\boldsymbol{M}}\tilde{\boldsymbol{T}}$ are both matched up to a shared permutation, which proves identifiability.

Case 2, $\mathbb{E}[x_1 \otimes x_3|x_2]$: For the task of predicting x_1, x_3 given x_2, the predictor takes the form

$$\mathbb{E}[x_1 \otimes x_3|x_2] = (\boldsymbol{OT}^\top)\text{diag}(\phi(x_2))(\boldsymbol{OT})^\top. \tag{2.74}$$

Let $\mathcal{X} := \{\mu_i : i \in [k]\}$ as in the previous case, and consider the 3-tensor

$$\begin{aligned} \boldsymbol{W} &:= \sum_{x_2\in\mathcal{X}} x_2 \otimes \mathbb{E}[x_1\otimes x_3|x_2] = \sum_{x_2\in\mathcal{X}} x_2 \otimes \mathbb{E}_{h_2|x_2}(\mathbb{E}[x_1|h_2]\otimes\mathbb{E}[x_3|h_2]) \\ &= \sum_{h_2}\sum_{x_2\in\mathcal{X}} p(h_2|x_2) x_2 \otimes \mathbb{E}[x_1|h_2]\otimes\mathbb{E}[x_3|h_2] \\ &= \sum_{i\in[k]} \underbrace{\Big(\sum_{x_2\in\mathcal{X}} (\phi(x_2))^\top e_i^{(k)} x_2 \Big)}_{:=a_i} \otimes (\boldsymbol{MT}^\top)_i \otimes (\boldsymbol{MT})_i, \end{aligned} \tag{2.75}$$

The first component is of rank-k as shown in the proof for $x_2 \otimes x_3|x_1$, and the other two components are of rank-k by Assumption 8. Thus Kruskal's theorem applies and the columns of $\boldsymbol{MT}, \boldsymbol{MT}^\top$ are recovered up to a shared permutation.

The first component $\{a_i\}_{i\in[k]}$ are also recovered, which means that if $\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}}$ form the same predictor as $\boldsymbol{M}, \boldsymbol{T}$, then for any linearly independent set $\mathcal{X}$ with k elements (not necessarily the previous choice of $\{\mu_i\}_{i\in[k]}$) such that $\mathcal{X}$ leads to a full rank $\boldsymbol{A}$, we have $\boldsymbol{A} = \tilde{\boldsymbol{A}}$ where $\tilde{\boldsymbol{A}}$ is parameterized by $\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}}$. For any such $\mathcal{X} = \{x^{(i)} : i \in [k]\}$, denote $\boldsymbol{X} := [x^{(1)}, ..., x^{(k)}]$, then

$$\boldsymbol{A} = \boldsymbol{X}[\phi(x^{(1)}), ..., \phi(x^{(k)})]^\top \boldsymbol{T}^\top = \boldsymbol{X}[\tilde{\phi}(x^{(1)}), ..., \tilde{\phi}(x^{(k)})]^\top \tilde{\boldsymbol{T}}^\top = \tilde{\boldsymbol{A}}. \tag{2.76}$$

Since $\boldsymbol{X}$ is of rank-k by the choice of $\mathcal{X}$, this means

$$[\tilde{\phi}(x^{(1)}), ..., \tilde{\phi}(x^{(k)})] = \underbrace{\tilde{\boldsymbol{T}}^{-1}\boldsymbol{T}}_{:=\boldsymbol{R}}[\phi(x^{(1)}), ..., \phi(x^{(k)})] \Rightarrow \tilde{\phi}(x^{(i)}) = \boldsymbol{R}\phi(x^{(i)}),\ \forall i \in [k]. \tag{2.77}$$

Moreover, for any valid choice of $\mathcal{X}$, matrices defined with points in sufficiently small neighborhoods of $x^{(i)}$ are still of full rank by the upper continuity of matrix rank. Hence the equality in equation 2.77 holds for points in these neighborhoods, and thus the Jacobian on both sides should be equal. Then, the exact same proof of Lemma 8 applies, and we get $\tilde{\phi}, \phi$ are equal up to a permutation of coordinates. Thus $\tilde{\boldsymbol{M}}$ must be equal to (up to permutation) either $\boldsymbol{M}$ or $\boldsymbol{HM}$ for a Householder reflection $\boldsymbol{H}$ by

Lemma 7. Finally, the solution of $\boldsymbol{HM}$ is eliminated since it would lead to a $\tilde{\boldsymbol{T}}$ that is not a valid stochastic matrix, as shown in the proof of Theorem 5.

□

2.2.4.2 Identifiability from pairwise conditional distribution

We show that matching the entire conditional *distribution* for G-HMM provides identifiability. Though this is implied by Theorem 5, which states that matching the conditional *expectation* already suffices, having access to the full conditional distribution allows an even simpler proof.

Theorem 9 (Identifiability of conditional distribution)**.** *Let $\boldsymbol{M}, \boldsymbol{T}$ and $\tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}}$ be two set of parameters satisfying Assumption 5 and 8. If $p(x_2|x_1; \boldsymbol{M}, \boldsymbol{T}) = p(x_2|x_1; \tilde{\boldsymbol{M}}, \tilde{\boldsymbol{T}})$, $\forall x_1, x_2 \in \mathbb{R}^d$, then $\boldsymbol{M} = \tilde{\boldsymbol{M}}, \boldsymbol{T} = \tilde{\boldsymbol{T}}$ up to a permutation of labeling.*

Proof. First note that the conditional distribution of x_2 given x_1 is a mixture of Gaussian, with means $\{\mu_i\}_{i\in[k]}$ and mixture weights given by $P(h_2|x_1) = \boldsymbol{T}P(h_1|x_1)$, hence we can directly apply the identifiability of Gaussian mixtures to recover the means $\{\mu_i\}_{i\in[k]}$:

Lemma 9 (Proposition 4.3 in Lindsay and Basak [1993])**.** *Let Q_k denote a Gaussian mixture with means $\{\xi_j\}_{j\in[k]} \in \mathbb{R}^d$. Suppose $\exists l \in [d]$ such that the set $\{[\xi_j]_l\}$ has distinct values, then one can recover $\{\xi_j\}_{j\in[k]}$ from moments of Q_k.*

We note that the assumption on the existence of a coordinate $l \in [k]$ is with out loss of generality, since we can first rotate the means to a different coordinate system in which this condition holds, then rotation back the means. Such rotation is guaranteed to exist since finding such rotation is equivalent to finding a vector $\boldsymbol{v}$ s.t. $\boldsymbol{v}^\top(\mu_i - \mu_j) \neq 0$ for every $i, j \in [k]$, for which the solution set is $\mathbb{R}^d \setminus \cup_{i,j\in[k]}\{\boldsymbol{u} : \boldsymbol{u}^\top(\mu_i - \mu_j) = 0\} \neq \emptyset$.

Recovering $\{\mu_i\}_{i\in[k]}$ means the scaled likelihood and the posterior both match, i.e. $\psi = \tilde{\psi}$, and $\phi(x) = P(h|x) = \frac{\psi}{\|\psi\|_1}$. The conditional distribution is

$$p(x_2|x_1) = \sum_{i,j\in[k]} p(x_2|h_2)p(h_2|h_1)p(h_1|x_1) = \frac{1}{(2\pi)^{d/2}}\psi(x_2)^\top \boldsymbol{T}\phi(x_1). \tag{2.78}$$

Choose a set $\mathcal{X} := \{x^{(i)}\}_{i\in[k]}$ such that $\Psi_{\mathcal{X}} := [\psi(x^{(1)}), ..., \psi(x^{(k)})] \in \mathbb{R}^{k\times k}$ is full rank. $\Phi_{\mathcal{X}} := [\phi(x^{(1)}), ..., \phi(x^{(k)})] \in \mathbb{R}^{k\times k}$ is also full rank since its columns are nonzero scalings of columns of $\Psi_{\mathcal{X}}$. Then we have

$$\Psi_{\mathcal{X}}^\top \boldsymbol{T}\Phi_{\mathcal{X}} = \tilde{\Psi}_{\mathcal{X}}^\top \tilde{\boldsymbol{T}}\tilde{\Phi}_{\mathcal{X}} = \Psi_{\mathcal{X}}^\top \tilde{\boldsymbol{T}}\Phi_{\mathcal{X}} \Rightarrow \boldsymbol{T} = \tilde{\boldsymbol{T}}. \tag{2.79}$$

□

2.2.5 Discussion

In this section, we took a model parameter identifiability view of self-supervised learning, which offers a complementary perspective to the current focus of feature learning for downstream performance. We focused on the masked prediction task on data generated by HMM and its conditionally-Gaussian variant G-HMM. We showed that parameter recovery is determined by the *task difficulty*, which can be tuned by both changing the parametric form of the data generative model, and by changing the masked prediction task.

We emphasize that this is a first-cut effort in the research program of analyzing SSL through the lens of model identifiability; we aim to build on this foundation to extend our analyses from HMMs to more complicated latent sequence and latent variable models. We also note that we have focused here on population analyses, and model identifiability. It would be of interest to build off this to develop and analyze the corresponding finite-sample learning algorithms for parametric generative models given SSL tasks, with sample complexity results, both in the realizable case, as well as in the agnostic case where we have model mis-specification. Given the use of conditional MLEs and regressions in SSL, and the natural robustness of these to model mis-specifications, we conjecture that these approaches will be much more robust when compared to say spectral methods.

Chapter 3

Understanding model classes: a case study on Transformers for reasoning

The choice of the model class is a crucial decision in the machine learning pipeline, as it impacts both the capacity and inductive biases of the model. The capacity determines the existence of a solution, and the inductive biases affect the properties of the solutions produced. In practice, the model class is determined by architectural choices. In this chapter, we focus on the Transformer architecture [Vaswani et al., 2017]. Transformer-based language models have undergone rapid evolution over the past few years, transitioning from generating broken English within the research community to being deployed across several critical public-facing sectors. While these advancements are remarkable, researchers and practitioners continue to uncover flaws and limitations in these models, some of which are surprisingly simple yet perplexing. This ongoing discovery process motivates us to develop a deeper and more precise understanding of the capabilities and inner workings of Transformers.

We focus our attention on *algorithmic reasoning*, a category of reasoning tasks typically solvable with known algorithms. Reasoning is a fundamental ability underlying most use cases of language models, and the algorithmic aspect offers the additional benefit of facilitating easy error checks. Examples of these tasks include solving arithmetic problem [Lee et al., 2023, Jelassi et al., 2023], determining parity [Bhattamishra et al., 2020a, Chiang and Cholak, 2022], and parsing matches parentheses [Hahn, 2020, Yao et al., 2021a].

We will use *finite-state automaton* as a sandbox for studying algorithmic reasoning, where one reasoning step is mapped to one state transition in a finite-state automaton. We will then present a positive representability result that Transformer can implement parallel reasoning solutions to the automaton simulation task, which we refer to as "shortcuts", as the number of sequential steps in these solutions is much smaller than the number of sequential transitions [Liu et al., 2023a] (Section 3.4). This offers a computational advantage over the classical recurrent neural networks (RNNs), whose sequential reasoning steps is the same as the number of transitions.

However, having sufficient representation power does not guarantee good solutions in practice. Using a base factor identified in the shortcut constructions, we stress test the robustness of the learned

Transformers [Liu et al., 2023b] and find that these practical solutions fail to robustly generalizing out-of-distribution (OOD) (Section 3.5). Our results reveal undesirable inductive biases in the attention mechanism, making Transformers compare unfavorably to RNNs at solving sequential reasoning tasks in practice.

Nevertheless, understanding the feasible solutions can provide practical insights. In Section 3.6 (based on Wen et al. [2023]), we show that characterizing the set of optimal solutions in a 2-layer Transformer leads to practical implications on interpretability and improvements on OOD generalization. Our sandbox is a formal languages named Dyck, which consists of balanced parentheses and is used to model the hierarchical structure in natural languages. We show that 2-layer Transformer contains an infinite number of optimal solutions even for a task as simple as modeling Dyck. Consequently, this implies that "myopic" interpretability methods which examine only a single component of a network are provably unrealiable. Our theoretical results also identify a necessary condition for a 2-layer Transformer to solve Dyck, which can be leveraged as a regularization term to improve the OOD performance of the trained model.

In the following, we will start with the necessary background of Transformer (Section 3.1) and automata (Section 3.3.1), and present the shortcut solutions in Section 3.4, challenges of OOD generalization in Section 3.5, and the implication to interpretability in Section 3.6.

3.1 The Transformer architecture

An L-layer *Transformer* is a sequence-to-sequence network $f_{\text{tf}} : \mathbb{R}^{T\times d} \times \Theta_{\text{tf}} \to \mathbb{R}^{T\times d}$, consisting of alternating self-attention blocks and feedforward blocks

$$f_{\text{tf}} := f^{(L)}_{\text{mlp}} \circ f^{(L)}_{\text{attn}} \circ f^{(L-1)}_{\text{mlp}} \circ ... \circ f^{(1)}_{\text{attn}}.$$

The parameter space Θ_{tf} is the Cartesian product of those of the individual blocks (without recurrent weight sharing across layers, by default). We define these two types of blocks below.

Attention. A *self-attention block* is a sequence-to-sequence function, which consists of one or multiple attention heads. [1] An attention *head* updates each position of the sequence using a weighted sum of all positions, which can be abstractly thought of as $[f_{attn}(x_1, \cdots, x_T)]_t = \sum_{t'\in[T]} \alpha_{t,t'}\phi(x_{t'})$ for some weights $\{\alpha_{t,t'}\}$ and some function $\phi : \mathbb{R}^d \to \mathbb{R}^d$.

Concretely, a single-headed ($H = 1$) [2] self-attention function $f_{\text{attn}} : \mathbb{R}^{T\times d} \times \Theta_{\text{attn}} \to \mathbb{R}^{T\times d}$ is parameterized by $\theta_{\text{attn}} = (W_Q, W_K, W_V, W_C)$ with $W_Q, W_K, W_V, W_C^\top \in \mathbb{R}^{d\times k}$ for an inner embedding dimension k. The pairwise *attention weights* $\{\alpha_{t,t'}\}$ are computed using the *query* and *key* matrices as $\alpha_{t,t'} \propto \langle W_Q^\top x_t, W_K^\top x_{t'}\rangle$ for each position $t' \in [T]$, interpreted as cosine similarity in a (asymmetrically) projected k-dimensional space. A variant that is commonly used in practice and will be considered in this chapter is to additionally apply a causal masking on α, which sets $\alpha_{t,t'} = 0, \forall t' > t$. [3] The attention

[1]The name "head" likely borrows the terminology of (neural) Turing machines [Graves et al., 2014], where a head is used to access memory locations. For Transformers, a length-T sequence can be considered as having T memory locations.

[2]In general, for any positive integer H, a *multi-headed self-attention block* consists of a sum of H copies of the above construction, each with its own parameters.

[3]This is called "causal" in the sense that an earlier time step t is not affected by a later time step t'.

weights are then normalized such that $\sum_{t'} \alpha_{t,t'} = 1, \forall t$. The function ϕ computes $\phi(x) = W_C^\top W_V^\top x$. In a single equation:

$$f_{\mathrm{attn}}(X; W_Q, W_K, W_V, W_C) := \mathsf{CausalAttn}(X W_Q W_K^\top X^\top) X W_V W_C,$$

where $\mathsf{CausalAttn} : \mathbb{R}^{T\times T} \to \mathbb{R}^{T\times T}$ applies a row-wise causally-masked T-dimensional softmax function. The standard softmax function $\mathsf{softmax}(z) : \mathbb{R}^T \to \mathbb{R}^T$ is used for normalization and defined by

$$[\mathsf{softmax}(z)]_t := \frac{e^{z_t}}{\sum_{t' \in [T]} e^{z_{t'}}};$$

the causally-masked softmax at row t is defined to be $\mathsf{softmax}(z_{1:t})$ on the first t coordinates, and 0 on the rest. To implement the causal masking operation, it is customary to set the entries above the diagonal of the attention score matrix $X W_Q W_K^\top X^\top$ to $-\infty$, then obtaining $\mathsf{CausalAttn}(X W_Q W_K^\top X^\top)$ via a row-wise softmax (letting $e^{-\infty}$ evaluate to 0).

The above version with the softmax is often called *soft attention*: the softmax performs continuous selection, taking a convex combination of its inputs. In contrast, *hard attention* refers to attention heads which perform truly sparse selection (putting weight 1 on the position with the highest score, and 0 on all others), which has been shown to lead to weaker representational power than soft attention [Hahn, 2020, Hao et al., 2022, Merrill et al., 2022].

Feedforward MLP. An L'-layer *position-wise feedforward MLP block* is a sequence-to-sequence network $f_{\mathrm{mlp}} : \mathbb{R}^{T\times d} \times \Theta_{\mathrm{mlp}} \to \mathbb{R}^{T\times d}$, parameterized by $\theta_{\mathrm{mlp}} = (W_1, b_1, \ldots, W_{L'}, b_{L'})$. For a choice of activation function $\sigma : \mathbb{R} \to \mathbb{R}$ (which is always ReLU in our theoretical constructions, for simplicity), f_{mlp} applies the same nonlinear map $(x \mapsto W_{L'} x + b_{L'}) \circ \sigma \circ \ldots \circ \sigma \circ (x \mapsto W_1 x + b_1)$ to each row t of the input matrix $X \in \mathbb{R}^{T\times d}$ (with the same parameters per position t); here, σ is applied pointwise.

Finally, an extra term $P \in \mathbb{R}^{T\times d}$ is added to the first layer's input, the matrix of *position encodings*.

Residual connections. It is typical to add *residual connections* which bypass each block. That is, letting id denote the identity function in $\mathbb{R}^{T\times d}$, the network (with position encodings) becomes

$$f_{\mathrm{tf}} := (\mathrm{id} + f_{\mathrm{mlp}}^{(L)}) \circ (\mathrm{id} + f_{\mathrm{attn}}^{(L)}) \circ (\mathrm{id} + f_{\mathrm{mlp}}^{(L-1)}) \circ ... \circ (\mathrm{id} + f_{\mathrm{attn}}^{(1)}) + (\mathrm{id} + P).$$

At the level of granularity of the results in this paper (up to negligible changes in the width and weight norms), this changes very little from the viewpoint of representation. A residual connection can be implemented (or negated) by appending two ReLU activations to a non-residual network:

$$x = (x)_+ - (-x)_+.$$

Similarly, a residual connection can be implemented with one attention head (with internal embedding dimension $k = d$), as long as it is able to select its own position (which will be true in all of our constructions).

In some of our constructions, we choose to use residual connections (sometimes restricted to certain dimensions); it will be very natural to view the embedding space $\mathbb{R}^d$ as a "workspace", where residual connections ensure that downstream layers can access the input (and position) embeddings, as well as outputs of all earlier layers. We will specify whether to use residual connections in each construction, to make the proofs as clear as possible. When we do so, we do not add the extra weights explicitly to f_{attn} and f_{mlp}.

Layer normalization. There are two types of layer normalization (abbreviated as "layernorm" or "LN"): the *Post-LN* which applies layernorm after adding the residual connection and is the original design in Vaswani et al. [2017], and the *Pre-LN* which applies layernorm before adding the residual connection. It is known that the use and placement of layernorm affect the training stability [Xiong et al., 2020]. For simplicity of presentation of our results though, we will omit the normalization layers. It would be straightforward (but an unnecessary complication) to modify the function approximation gadgets in our constructions to operate with unit-norm embeddings.

Padding tokens. Finally, for results in Section 3.4, it will greatly simplify the constructions to add padding tokens: to simulate a semiautomaton at length T, we will choose to prepend τ tokens, with explicitly chosen embeddings, which do not depend on the input $\sigma_{1:T}$. Theorem 1 uses $\tau = \Theta(T)$ padding, and Theorem 2 uses $\tau = 1$. In both cases, padding is not strictly necessary (the same functionality could be implemented by the MLPs without substantially changing our results), but we find that it leads to the most intuitive and concise constructions.

Complexity measures. We define the following quantities associated with a Transformer network, and briefly outline their connection to familiar concepts in circuit complexity:

- The dimensions according to the definition of a sequence-to-sequence network: *sequence length* T and *embedding dimension* d. Up to a factor of bit precision, this corresponds to the number of inputs in a classical Boolean circuit. We will exclusively define architectures where d is independent of T.
- Its *depth* L, the number of repeated $f_{\text{mlp}} \circ f_{\text{attn}}$ blocks. When each of these modules contains a constant number of sequential computations, this coincides with the usual notion of circuit depth, up to a constant factor. This is true in practice and our theoretical treatment (the attention and MLP have a constant number of layers).
- The other shape parameters from the definition of the architecture: *number of heads* (per layer and position) H[4], and *internal embedding dimension* k. When f_{mlp} is an L'-layer MLP, it has *MLP intermediate widths* $d_1, \ldots, d_{L'-1}$. We will exclusively think of L' as a small constant, so that the number of sequential matrix multiplications in the entire network is within a constant factor of L.
- Its *attention width* w_{attn} is defined to be the maximum of $\{d, Hk\}$, and its *MLP width* w_{mlp} is defined as the maximum of $\{d_1, \ldots, d_{L'-1}\}$. Taking $w = \max(w_{\text{attn}}, w_{\text{mlp}})$ as a coarse upper bound we will use to summarize the number of per-position trainable embeddings in our constructions. To map this to the usual notion of *circuit size*, note that the computations are repeated position-wise. Thus, Transformers induce a computational graph with $O(T \cdot L \cdot w)$ gates and

[4]There will be a notational collision between h, H denoting attention heads, and $h \in H$ denoting an element in a group. We keep the overloaded notation for clarity, and this will certainly be unambiguous.

$O(T \cdot L \cdot w^2)$ wires. The position-wise parameter-sharing induces a special notion of circuit uniformity.

- A bound on its ∞-weight norms: the largest absolute value of any trainable parameter. These can be converted into norm-based generalization bounds via the results in Edelman et al. [2022]. Note that the results in this paper go beyond the *sparse variable creation* constructions of bounded-norm attention heads; in general, the norms scale with T. The attention heads express meaningful non-sparse functions. Aside from the positive experimental results, we do not directly investigate generalization in this paper.

- The bit precision (length of finite-precision truncation of real numbers in a computational graph implementing f_{tf}), which lets us implement approximate real-valued computations as Boolean (or discrete arithmetic) circuits. With infinite-precision real numbers, there are pathological constructions for RNNs [Siegelmann and Sontag, 1992] and Transformers [Merrill et al., 2021] which give single parameters of neural networks infinite representational power. Throughout this work, our circuits will work with $O(\log T)$ bit precision, which can represent real numbers (as integers $\lfloor x \cdot 2^c \rfloor$ in their binary representation, for some choice of $c = \Theta(\log T)$) with magnitude up to $O(\text{poly}(T))$, with $O(1/\text{poly}(T))$ approximation error. Since this is far from the focus of our results, we will elide details for the remainder of this paper, returning to these considerations only to make Theorem 13 more concrete. All of our constructions are robust up to this noise level: this is because the internal weight norms and activations are bounded by a quantity at most polynomial in T, and the function approximation construction in Lemmas 10 and 11 can tolerate $1/\text{poly}(T)$ perturbations using $\text{poly}(T)$ weight norms.

Why Transformers?

Transformer, with its numerous variants, is the most widely adopted architecture for language models as of 2024. The term "Transformer" was proposed in Vaswani et al. [2017], though many elements essential to Transformers had been proposed in earlier works. It thus might be helpful to briefly review the historical contexts. Transformer was originally an encoder-decoder structure[5] designed for translation, a type of sequence-to-sequence modeling task. Before Transformers, there had been decades of research on sequence modeling using recurrent neural networks. The earliest versions, Jordan and Elman RNNs [Jordan, 1997, Elman, 1990], were proposed in the 1980s and shown to be Turing complete by [Siegelmann and Sontag, 1992] assuming infinite precision. For practical considerations though, finite-precision, finite-state-size RNNs face several challenges, including training instabilities, capacity constraints, and the failure to model long-range dependencies. As mitigations, later work has introduced gating (such as LSTM [Hochreiter and Schmidhuber, 1997b]) and attention (over bi-LSTM states [Bahdanau et al., 2014] or a memory bank implemented using a matrix [Graves et al., 2014]).

Transformers are heavily influenced by these prior approaches but with key differences. Bahdanau et al. [2014] allows for modeling arbitrarily long pairwise dependency between tokens similar to Transformers, but it computes attention scores using a generic feedforward network, whereas Transformer uses a simpler form which computes the inner product in a lower-dimensional space and removes the RNN component from the architecture. Neural Turing Machine [Graves et al., 2014]

[5]This chapter considers decoder-only Transformers.

borrowed terminology in Turing machine to refer to the attention outputs as "heads," which is also adopted by Transformers. However, the memory bank in Neural Turing Machine is of fixed size, whereas Transformer enjoys an unbounded memory whose size grows with the sequence length. The later has provable advantages over a fixed-size memory [Wen et al., 2024].

It is worth noting that despite the ability to represent shallow solutions (as we will see in Section 3.4), Transformers are not the most computationally efficient due to the quadratic time required for computing pairwise attention scores. There has been in linear-time methods, including efficient Transformers [Yi et al., 2021] and linear RNNs (such as state-space models) [Gu et al., 2021, Gu and Dao, 2023, Orvieto et al., 2023]. Understanding the implications of linear-time methods [Sarrof et al., 2024, Yang et al., 2024] and hybrid architectures [Wen et al., 2024, Lieber et al., 2024, Ren et al., 2024] is a promising future direction.

3.2 Overview of results: finite-state automata for modeling sequential algorithmic reasoning

Reasoning abilities represent a fundamental aspect of intelligence that is essential in almost all scenarios that involve decision making. In the following, we will consider a simple form of reasoning, which can be considered as sequentially taking actions upon receiving new information.

Such actions can be modeled as state transitions in a *semiautomaton*, which computes state sequences from inputs by application of a recurrent transition function. Semiautomata describe the underlying dynamics of *automata*, which are simply semiautomata equipped with mappings from states to outputs. With unbounded state spaces, automata can represent *all* algorithms; however, even bounded automata form a rich class of sequence processing algorithms, containing regular expression parsers and finite-state transducers. In reinforcement learning and control, semiautomata are better known as deterministic Markov models (where σ_t are actions); thus, in addition to algorithmic reasoning, our results also have implications to Transformer dynamics models.

Formally, a *semiautomaton* $\mathcal{A} := (Q, \Sigma, \delta)$ consists of a set of states Q, an input alphabet Σ, and a transition function $\delta : Q \times \Sigma \to Q$. We consider Q and Σ to be finite sets. For all positive integers T and a starting state $q_0 \in Q$, $\mathcal{A}$ defines a map from input sequences $(\sigma_1, \ldots, \sigma_T) \in \Sigma^T$ to state sequences $(q_1, \ldots, q_T) \in Q^T$: $q_t := \delta(q_{t-1}, \sigma_t)$ for $t = 1, \ldots, T$. This is a deterministic Markov model, in the sense that at time t, the future states $q_{t+1}, \ldots, q_T$ only depend on the current state q_t and the future inputs $\sigma_{t+1}, \ldots, \sigma_T$. Every semiautomaton induces a *transformation semigroup* [6] $\mathcal{T}(\mathcal{A})$ of functions $\rho : Q \to Q$ under composition, generated by the per-input-symbol state mappings $\delta(\cdot, \sigma) : Q \to Q$. When $\mathcal{T}(\mathcal{A})$ contains the identity function and that all of the functions are invertible, $\mathcal{T}(\mathcal{A})$ is a *permutation group*. An elementary example is a parity counter, which the word occurrence example will be modeled by: the state is a bit (i.e. 0 or 1), and the inputs are {"toggle the bit", "do nothing"}; the transformation semigroup is the cyclic group of order 2. We will also take a close look at the *flip-flop* automaton and bounded-depth *Dyck*, which we will formally define in Section 3.5 and Section 3.6.

[6] A semigroup is a generalization of a group. Recall that a group consists of a set and an invertible associative binary operation on the set, and the set needs to contain an identify element with respect to the binary operation. A semigroup relaxes both the identity condition and the invertibility condition.

Task: simulating semiautomata When using semiautomata as a model for reasoning, performing reasoning corresponds to the task of *simulation*: given a semiautomaton $\mathcal{A}$, starting state q_0, and input sequence $(\sigma_1, \ldots, \sigma_T)$, output the state trajectory $\mathcal{A}_{T,q_0}(\sigma_1, \ldots, \sigma_T) := (q_1, \ldots, q_T)$. Let $f : \Sigma^T \to Q^T$ be a function (which in general can depend on $\mathcal{A}, T, q_0$). We will say that f *simulates* $\mathcal{A}_{T,q_0}$ if $f(\sigma_{1:T}) = \mathcal{A}_{T,q_0}(\sigma_{1:T})$ for all input sequences $\sigma_{1:T}$. Finally, for a positive integer T, we say that a function class $\mathcal{F}$ of functions from $\Sigma^T \to Q^T$ simulates $\mathcal{A}$ at length T if, for each $q_0 \in Q$, there is a function in $\mathcal{F}$ which simulates $\mathcal{A}_{T,q_0}$

Model: neural sequence models Let's discuss the neural sequence models used for modeling the sequential task of automata simulation. A *sequence-to-sequence neural network* of length T and embedding dimension d is a function $f_{\text{nn}} : \mathbb{R}^{T\times d} \times \Theta \to \mathbb{R}^{T\times d}$, with *trainable parameters* $\theta \in \Theta$. Equipped with an *encoding layer* $E : \Sigma \to \mathbb{R}^d$ and *decoding layer* $W : \mathbb{R}^d \to Q$ (both applied position-wise), and fixing some parameters θ, the function $(W \circ f_{\text{nn}} \circ E) : \Sigma^T \to Q^T$ has the same input and output types as $\mathcal{A}_{T,q_0}$. We consider two main types of models with different modes of computation: recurrent neural networks, which computes sequentially on a length-T sequence, and Transformers, whose computation is parallel across all sequence positions.

More specifically, a *recurrent neural network (RNN)* is defined by iterated composition of a *recurrent unit* $g : \mathbb{R}^d \times \mathbb{R}^d \times \Theta \to \mathbb{R}^d$. For a given initial hidden state $h_0 \in \mathbb{R}^d$, and input sequence $u_1, \ldots, u_T \in \mathbb{R}^d$, it produces an output hidden state sequence $h_t := g(h_{t-1}; u_t; \theta), \qquad t = 1, \ldots, T$. Thus, fixing the parameters θ, an RNN is an infinite semiautomaton, with $Q = \Sigma = \mathbb{R}^d$. Thus, RNNs can trivially simulate any semiautomaton by embedding the looped transition function δ, as long as g can represent δ.

An L-layer (or depth-L) *Transformer* consists of alternating self-attention blocks and feedforward MLP blocks

$$f_{\text{tf}} := (\text{id} + f^{(L)}_{\text{mlp}}) \circ (\text{id} + f^{(L)}_{\text{attn}}) \circ (\text{id} + f^{(L-1)}_{\text{mlp}}) \circ \ldots \circ (\text{id} + f^{(1)}_{\text{attn}}) \circ (\text{id} + P),$$

where id denotes the identity function (residual connections), and P encodes the positions t.[7] We use fairly standard positional encodings in both theory and experiments. Briefly, an attention layer performs ℓ_1-normalized mixing operations across positions t, while a constant-layer MLP block performs position-wise function approximation (with no mixing between positions). Importantly, the standard Transformer is convolutional (in that the weights in f_{attn} and f_{mlp} are shared across positions t), but *not* recurrent: parameters are shared across positions within a layer but are not shared across blocks of different layers.

Typical Transformers are shallow, in the sense that $L \ll T$. In practice, this makes their inference and gradient computations highly parallelizable, with the number of sequential computation steps scaling linearly in L, while RNNs require a scaling linear in T. While there is always a canonical way for RNNs to simulate semiautomata, the answer to the analogous question for shallow Transformers is far less obvious. We will see next that shallow Transformers are in fact sufficiently powerful representationally, yet there are challenges in optimization and generalization.

[7]We omit layer normalization; this discrepancy can be shown to be superficial.

3.2.1 Transformer can implement shortcut for simulating automata

Modern deep learning systems demonstrate increasing capabilities of algorithmic reasoning. Particularly in modalities such as natural language, math, and code, neural networks can successfully parse and synthesize sequences containing symbolic information and compositional structure. To exhibit these functionalities, these networks are required to learn and execute the relevant discrete algorithms within their internal representations. A core open question in this domain is that of mechanistic understanding: *how do neural networks encode the primitives of algorithmic reasoning?*

When considering this question, there is an apparent mismatch between classical sequential models of computation (e.g. Turing machines) and the Transformer, the state-of-the-art architecture for neural algorithmic reasoning. If we are to think of algorithms as sequentially-executed computational rules, why should we use a shallow[8] and non-recurrent architecture to represent them?

We study this question through the lens of *semiautomata*, which compute state sequences $q_1, \dots, q_T$ from inputs $\sigma_1, \dots, \sigma_T$ by application of a recurrent transition function δ (and initial state q_0):

$$q_t = \delta(q_{t-1}, \sigma_t).$$

Semiautomata describe the underlying dynamics of *automata*, which are simply semiautomata equipped with mappings from states to outputs. With unbounded state spaces, automata can represent *all* algorithms; however, even bounded automata form a rich class of sequence processing algorithms, containing regular expression parsers and finite-state transducers. In reinforcement learning and control, semiautomata are better known as deterministic Markov models (where σ_t are actions); thus, in addition to algorithmic reasoning, this work also pertains to Transformer dynamics models.

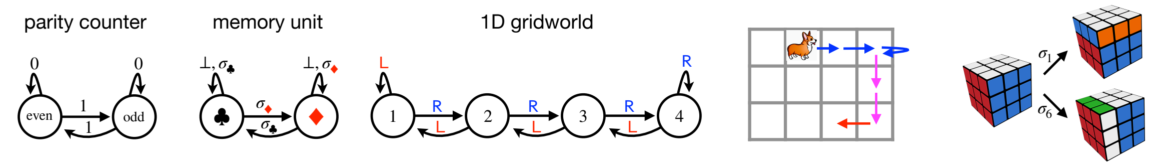
We perform a theoretical and empirical investigation of whether (and how) non-recurrent Transformers perform the computations of semiautomata. We find that Transformers learn *shortcut solutions*, which correctly and efficiently simulate the sequential transitions of semiautomata using a shallow parallel circuit, rather than naively iterating the single-step recurrence. Shortcuts arise from hierarchical reparameterizations of a semiautomaton's global transition dynamics.

Our contributions. Our theoretical results provide structural guarantees for the representability of semiautomata (thus, iterative algorithms) by one pass through a shallow, non-recurrent Transformer. In particular, we show that:

- Shortcut solutions, with depth logarithmic in the sequence length, always exist (Theorem 10).
- Constant-depth shortcuts exist for *solvable* semiautomata (Theorem 11). There *do not* exist constant-depth shortcuts for non-solvable semiautomata, unless $\mathsf{TC}^0 = \mathsf{NC}^1$ (Theorem 13). These are understood via the Krohn-Rhodes theorem, a landmark result in semigroup theory.
- For a natural class of semiautomata corresponding to path integration in a "gridworld" with boundaries, we show that there are even shorter shortcuts (Theorem 12), beyond those guaranteed by the general structure theorems above.

We accompany these with an extensive set of experimental findings:

[8]In practice, a Transformer's context length is typically far greater than its depth.



$Q = \{\text{even}, \text{odd}\}$ $\Sigma = \{0, 1\}$

$Q = \{\clubsuit, \diamondsuit\}$ $\Sigma = \{\sigma_\clubsuit, \sigma_\diamondsuit, \perp\}$

$Q = \{1, 2, 3, 4\}$ $\Sigma = \{\text{L}, \text{R}\}$

$Q = \{1..3\} \times \{1..4\}$ $\Sigma = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$

$Q = \{54 \text{ stickers}\}$ $\Sigma = \{6 \text{ face rotations}\}$

Figure 3.1: Various examples of semiautomata. From left to right: a mod-2 counter, a 2-state memory unit, Grid$_4$, a 2-dimensional gridworld constructible via a direct product Grid$_3 \times$ Grid$_4$, and a Rubik's Cube, whose transformation semigroup is a very large non-abelian group.

- *Transformers learn shortcuts with standard training* (Section 3.4.2). Across a wide variety of semiautomaton simulation problems, we find that shallow non-autoregressive Transformers successfully learn shortcut solutions: despite the non-convex optimization problem, gradient-based training works. This suggests that shortcuts are plausible mechanisms for algorithmic reasoning in non-synthetic sequence models, and lies beyond our current theoretical understanding.
- *Shortcuts are statistically brittle* (Section 3.4.3). We identify empirical weaknesses of the shortcuts found by Transformers: poor out-of-distribution generalization (including to unseen sequence lengths) and worse performance than RNNs under limited supervision. Toward mitigating these drawbacks and obtaining the best of both worlds, we show that with *recency-biased scratchpad training*, autoregressive Transformers can easily be guided to learn the iterative RNN-like solutions (*chain-of-thought* generation).

Related work

Emergent reasoning in neural sequence models. Neural sequence models, both recurrent [Wu et al., 2016, Peters et al., 2018, Howard and Ruder, 2018] and non-recurrent [Vaswani et al., 2017, Devlin et al., 2018b], have become an era-defining tool for parsing and transducing data with combinatorial structure, such as natural language and code. A nascent frontier is to leverage neural dynamics models, again both recurrent [Hafner et al., 2019] and non-recurrent [Chen et al., 2021a, Janner et al., 2021], for decision making. At the highest level, the present work seeks to understand the mechanisms by which these models perform combinatorial and algorithmic reasoning.

Computational models within neural networks. Despite the preponderance of positive empirical results, many mysteries remain towards understanding the internal mechanisms of neural networks capable of algorithmic reasoning. It is known that self-attention realizes low-complexity circuits [Hahn, 2020, Elhage et al., 2021, Merrill et al., 2021, Edelman et al., 2022], declarative programs [Weiss et al., 2021], and Turing machines [Dehghani et al., 2019, Pérez et al., 2021, Giannou et al., 2023]. Interpretable symbolic computations have been extracted from trained models [Clark et al., 2019a, Vig, 2019, Tenney et al., 2019, Wang et al., 2022a]. Our conclusions are closest to the literature on the universal representation on Turing machines (which are automata with unbounded states); however, our work is unique in characterizing the smaller (but still important) classes of machines whose execution loops can be efficiently unrolled into a *single pass* of a shallow Transformer.

At a technical level, the most relevant theoretical work to ours is [Barrington and Thérien, 1988], which acts as a "Rosetta Stone" between circuit complexity and semigroup theory. The core technical ideas for Theorems 10 (NC^1 prefix sum), 11 (Krohn-Rhodes), and 13 (Barrington) are inspired by the results and discussions therein. For readers familiar with circuit complexity: our theoretical results establish that Transformers (a certain family of arithmetic circuits) efficiently embed the constructions involved in the NC^1 and ACC^0 solutions to semigroup word problems. The notions of efficiency (depth, parameter count, and weight norms) are standard in deep learning but not circuit complexity; our embedding avoids suboptimal $\text{poly}(T)$ factors in these complexity measures. Theorem 12 comes from an improved parallel algorithm for the special case of "gridworld" semigroups; to our knowledge, this construction is novel, and may be of independent interest.

Learning elementary algorithms with Transformers. Our work provides a unifying lens on many recent investigations on whether (and how) Transformers represent certain classes of fundamental algorithmic computations. These include bounded-depth Dyck languages [Yao et al., 2021b], modular prefix sums [Anil et al., 2022b], adders [Nogueira et al., 2021, Nanda and Lieberum, 2022], regular languages [Bhattamishra et al., 2020a], and sparse logical predicates [Edelman et al., 2022, Barak et al., 2022a], which are all special cases of simulating finite-state automata. Thus, our work provides guarantees of shallow Transformer solutions in all of these settings. Zhang et al. [2022] empirically analyze the behavior and inner workings of Transformers on random-access group operations and note "shortcuts" (which skip over explicit program execution) similar to those we study.

We provide an expanded discussion of related work in Appendix 3.7.

3.2.2 Exposing Transformer's OOD failures using flip-flop

Recent advancements in scale have yielded large language models (LLMs) with extraordinary proficiency in nuanced reasoning with factual knowledge. Despite these achievements, LLMs are known to produce incorrect outputs, often referred to colloquially as "hallucinations" or "distractions" [Ji et al., 2023]. Generally, hallucinations refer to the phenomenon that a model's outputs are syntactically and grammatically accurate but factually incorrect. There are various types of hallucinations, and the focus of this work is the "closed-domain" variety [Saparov and He, 2022, OpenAI, 2023], where the model predictions contain factually incorrect or made-up information *according to a given context*, regardless of their correctness in the real world.

Perhaps surprisingly, such hallucinations can be observed even on simple algorithmic reasoning tasks. As a warmup, consider the queries shown in Figure 1 (and Appendix 3.5.6.1), where we prompt LLMs to solve addition problems of various lengths. The responses simultaneously illustrate the following:

1. *Nontrivial algorithmic generalization:* In cases where the models succeed, it is unlikely that these exact numerical sequences appeared in the training data. To correctly output the first digit of the answer, the LLM must resolve a long dependency chain which generally depends on every digit in the input. Somewhere within these networks' internal representations, implementations of addition algorithms have emerged.

2. *Sporadic errors ("hallucinations"):* These internal algorithms can be brittle and unreliable, especially when processing long inferential chains. Their failures can be subtle and unpredictable.

In this work, we consider *flip-flop language* processing, a minimal unit of sequential computation

User: What is 8493 + 2357?
GPT-3.5: `10850` ✓
GPT-4: `10850` ✓

User: What is 84935834 + 23572898?
GPT-3.5: `108008732` ×
GPT-4: `108508732` ✓

User: What is 9991999919909993 + 6109199190990097?
GPT-3.5: `16111199190810090` ×
GPT-4: `16101199100890090` ×
Answer: `16101199110900090`

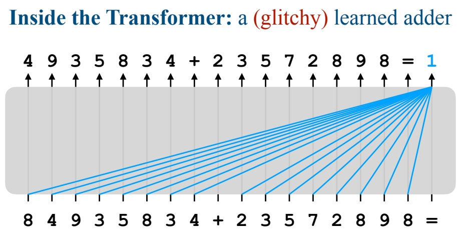


Figure 3.2: Cherry-picked integer addition prompts, showing how state-of-the-art LLMs can generalize non-trivially on algorithmic sequences, but sporadic reasoning errors persist. The first digit of the correct answer depends on every input; thus, an autoregressive model must propagate a "carry" bit across these long-range dependencies in a single pass. This (and many other algorithmic reasoning capabilities) can be implemented by a Transformer model using internal *flip-flops*.

which consists of memory operations on a single bit (see Example 2) and underlies virtually all[9] syntactic parsing and algorithmic reasoning capabilities (including implementing adders, and far more). A *flip-flop language modeling* (FFLM) task is defined on sequences of `write`, `read`, and `ignore` instructions: `write` sets the memory state to a certain value which is later retrieved by `read`, while `ignor`ing any contents in between. We find that when trained to complete flip-flop sequences, the Transformer architecture exhibits a long tail of reasoning errors, unlike previous-generation recurrent models such as the LSTM [Hochreiter and Schmidhuber, 1997a]. We coin the term *attention glitch* for this phenomenon, and hypothesize that this captures a systematic failure mode of Transformer-based LLMs when internally representing long chains of algorithmic reasoning.

Our contributions are as follows:

- **FFLM: a minimalistic long-range dependency benchmark.** We propose *flip-flop language modeling*, a parametric family of synthetic benchmarks for autoregressive sequence modeling, designed to isolate and probe reasoning errors like those demonstrated in Figure 3.2.
- **An empirical failure of attention to attend.** We find that while Transformer models can appear to learn flip-flop languages perfectly on held-out samples from the training distribution, they make a long tail of unpredictable reasoning errors (*attention glitches*), on both long-range and short-range dependencies. We evaluate various direct and indirect mitigations, including commonly-used regularization techniques and *attention-sharpening regularizers* — a plug-and-play way to sparsify self-attention architectures. We find that attention-sharpening reduces reasoning errors by an order of magnitude, but none of our attempts were successful in driving the number of errors to exactly 0. Meanwhile, even tiny recurrent models work perfectly.
- **Preliminary mechanistic analyses.** We provide some theoretical and empirical explorations which account for some of the internal mechanisms for attention glitches, and why they are so difficult to eliminate completely.

[9]More precisely, whenever the desired algorithm needs to "store memory" (i.e. contains a non-invertible state transformation); see Section 3.5.1.2.1.

Related work

The challenge of learning long-range dependencies is a long-standing one in the statistical modeling of sequences [Samorodnitsky et al., 2007]. The Transformer architecture [Vaswani et al., 2017], a paradigm-shifting sequence model, enables the scalable learning of a feedforward hierarchy of meaningful long-range dependencies. Yet, factual errors over long contexts persist in these models; this is the subject of many careful probes in deep NLP [Khandelwal et al., 2018, Tay et al., 2020, Guo et al., 2022, Ji et al., 2023].

The sporadically erroneous outputs of LLMs, which have otherwise demonstrated impressive human-like linguistic reasoning, have been popularly called "hallucinations". There has been a large amount of research into mitigating this, where the main ideas include explicitly writing out the intermediate reasoning steps [Nye et al., 2021b, Wei et al., 2022b], and self-consistency [Wang et al., 2022b]. Nevertheless, there is no clear formal definition of hallucinations (other than "wrong answers"), and a functional taxonomy of these errors is still under development [Saparov and He, 2022, Ji et al., 2023]. The present work considers a minimal setting, in which closed-domain hallucinations are completely well-defined; we discuss connections to LLM hallucinations in Section 3.5.5.

Long-range dependency and reasoning benchmarks. Many datasets and tasks have been designed to isolate considerations similar to ours [Tay et al., 2020, Wu et al., 2021, Zhang et al., 2021a, 2022, Saparov and He, 2022, Shi et al., 2023, Eldan and Li, 2023]. Aside from being focused on the "smallest" and "purest" sequential reasoning capability (see Section 3.5.1.2.1), FFLM is distinguished by a few factors:

- **"L_∞" objective:** Unlike usual benchmarks, we consider any model with less than 100% accuracy as exhibiting a *reasoning error*. Aside from the motivation of completely eliminating hallucinations, we argue that this stringent notion of correctness is needed to avoid error amplification when flip-flops are embedded in more complex networks, based on the decomposition result discussed in Section 3.4. We will discuss this more in Section 3.5.1 and Section 3.5.5.
- **Parametric, procedurally generated, and generalizable:** Our empirical study precisely quantifies long-tail errors via a large number of replicates over the randomness of both model initialization and data generation. Our methodology can be adapted and resized to probe language models of any size.

Comparison with *Transformers Learn Shortcuts to Automata*. Liu et al. [2023a] study the parallel circuits efficiently realizable by low-depth Transformers. The authors identify *shortcut solutions*, which exactly replicate length-T recurrent computations ("chains of algorithmic reasoning") in the absence of recurrence, with very few ($O(\log T)$; sometimes $O(1)$) layers. Their results contain a general structure theorem of *representability*, and preliminary positive empirics for *generalization* and *optimization*, demonstrating that Transformers can learn these shallow solutions via gradient-based training on samples. In contrast, the present work is a fine-grained study of the issue of generalization. Our main empirical contributions are a minimally sufficient setup (FFLM) and a set of large-scale[10]

[10] $\sim 10^4$ 19M-parameter Transformers were trained in the making of this paper; see Appendix 3.5.6.6.

controlled experiments, towards providing reasonable scientific foundations for addressing the unpredictable reasoning errors of LLMs.

3.2.3 Infinite solutions to Dyck and implications to interpretability

With the growing deployment of Transformer in various applications, it is increasingly essential to understand the inner working of these models. There have been great advancement in the field of interpretability presenting various types of evidence [Clark et al., 2019b, Vig and Belinkov, 2019, Wiegreffe and Pinter, 2019, Nanda et al., 2023, Wang et al., 2023], some of which, however, can be misleading despite being highly intuitive [Jain and Wallace, 2019, Serrano and Smith, 2019, Rogers et al., 2020, Grimsley et al., 2020, Brunner et al., 2020, Meister et al., 2021].

In this work, we aim to understand the theoretical limitation of a family of interpretability methods by characterizing the set of viable solutions. We focus on "myopic" interpretability methods, i.e. methods based on examining individual components only. We adopt a particular toy setup in which Transformers are trained to generate *Dyck grammars*, a classic type of formal language grammar consisting of balanced parentheses of multiple types. Dyck is a useful sandbox, as it captures properties like long-range dependency and hierarchical tree-like structure that commonly appear in natural and programming language syntax, and has been an object of interest in many theoretical studies of Transofmers [Hahn, 2020, Yao et al., 2021a, Liu et al., 2022c, 2023a]. Dyck is canonically parsed using a stack-like data structure. Such stack-like patterns (Figure 3.3) have been observed in the attention heads [Ebrahimi et al., 2020], which was later bolstered by mathematical analysis in Yao et al. [2021a].

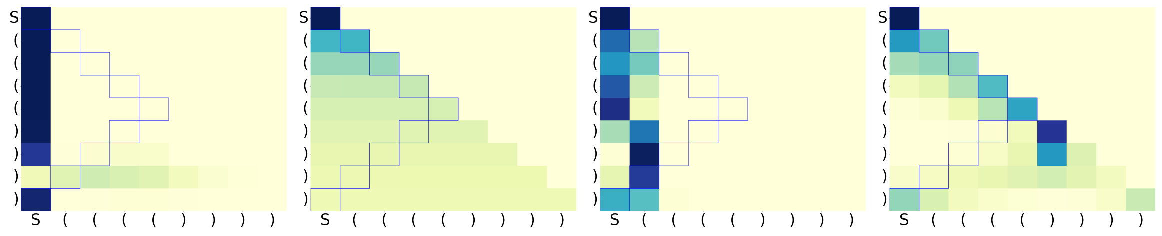
From a representational perspective and via explicit constructions of Transformer weights, recent work [Liu et al., 2023a, Li et al., 2023] show that Transformers are sufficiently expressive to admit very different solutions that perform equally well on the training distribution. Thus, the following questions naturally arise:

(Q1) Do Transformer solutions found empirically match the theoretical constructions given in these representational results (Figure 3.3)? In particular, are interpretable stack-like pattern in Ebrahimi et al. [2020] the norm or the exception in practice?

(Q2) More broadly, can we understand in a principled manner the fundamental obstructions to reliably "reverse engineering" the algorithm implemented by a Transformer by looking at individual attention patterns?

(Q3) Among models that perform (near-)optimally on the training distribution, even if we cannot fully reverse engineer the algorithm implemented by the learned solutions, can we identify properties that characterize performance beyond the training distribution?

Theoretical understanding of representability Methodologically, our work joins a long line of prior works that characterize the solution of neural networks via the lens of simple synthetic data, from class results on RNN representability [Siegelmann and Sontag, 1992, Gers and Schmidhuber, 2001, Weiss et al., 2018, Suzgun et al., 2019, Merrill, 2019, Hewitt et al., 2020], to the more recent Transformer results on parity [Hahn, 2020], Dyck [Yao et al., 2021a], topic model [Li et al., 2023], and formal grammars in general [Bhattamishra et al., 2020b, Li and Risteski, 2021, Zhang et al., 2022, Liu et al., 2023a,



(a) With Position Embedding

(b) With Position Embedding

(c) Without Position Embedding

(d) Without Position Embedding

Figure 3.3: **Second-layer attention patterns of two-layer Transformers on Dyck**: typical attention patterns do *not* exactly match the intuitively interpretable stack-like pattern prescribed in Ebrahimi et al. [2020], Yao et al. [2021a]. The blue boxes indicate the locations of the last unmatched open brackets, as they would appear in a stack-like pattern. All models reach $\geq 97\%$ accuracy (defined in Section 3.6.3.1). In the heatmap, darker color indicates larger value.

Zhao et al., 2023]. Our work complements prior works by showing that although representational results can be obtained via intuitive "constructive proofs" that assign values to the weight matrices, the model does not typically converge to those intuitive solutions in practice. Similar messages are conveyed in Liu et al. [2023a], which presents different types of constructions using different numbers of layers. In contrast, we show that there exist multiple different constructions even when the number of layers is kept the same.

Our contributions. We first prove several theoretical results to provide evidence for why individual components (e.g. attention patterns or weights) of a Transformer should not be expected to be interpretable. In particular, we prove:

- A **perfect balance** condition (Theorem 1) on the attention pattern that is sufficient and necessary for 2-layer Transformers with a *minimal first layer* (Assumption 10) to predict optimally on Dyck of *any* length. We then show that this condition permits abundant *non-stack-like* attention patterns that do not necessarily reflect any structure of the task, including *uniform* attentions (Corollary 1).
- An **approximate balance** condition (Theorem 2), the *near-optimal* counterpart of the condition above, for predicting on *bounded*-length Dyck. Likewise, non-stack-like attention patterns exist.
- **Indistinguishability from a single component** (Theorem 3), proved via a *Lottery Ticket Hypothesis* style argument that any Transformer can be approximated by pruning a larger random Transformer, implying that interpretations based exclusively on local components may be unreliable.

We further accompany these theoretical findings with an extensive set of empirical investigations.

Is standard training biased towards interpretable solutions? While both stack-like and non-stack like patterns can process Dyck theoretically, the inductive biases of the architecture or the optimization process may prefer one solution over the other in practice. In Section 3.6.3.1, based on a wide range of Dyck distributions and model architecture ablations, we find that Transformers that generalize near-perfectly in-distribution (and reasonably well out-of-distribution) do *not* typically produce stack-like attention patterns, showing that the results reported in prior work [Ebrahimi et al., 2020] should not be expected from standard training.

Do non-interpretable solutions perform well in practice? Our theory predicts that balanced (or even uniform) attentions suffice for good in- and out-of-distribution generalization. In Section 3.6.3.2, we empirically verify that with standard training, the extent to which attentions are balanced is positively correlated with generalization performance. Moreover, we can guide Transformers to learn more balanced attention by regularizing for the balance condition, leading to better length generalization.

3.3 Preliminary

This section provides the necessary background for results in this chapter. Section 3.1 introduces the Transformer architecture, with a discussion on various complexity measures. Section 3.3.1 then provides a self-contained intro to automata and group theory, which are the algebraic structures that underlie the constructions in this chapter. Section 3.3.2 describes some circuit complexity classes relevant to results in this section.

We will use the following notational conventions for indices, vectors, matrices, and functions.

- For a natural number n, $[n]$ denotes the *index set* $\{1, 2, \dots, n\}$.
- For a vector $v \in \mathbb{R}^n$ and $i \in [n]$, v_i denotes the i-th entry. When v is an expression, we use $[v]_i$ for clarity. Vectors can be instantiated by square brackets (like $[1\ 2\ 3] \in \mathbb{R}^3$). They can be indexed by slices: $v_{a:b}$ denotes $[v_a\ v_{a+1}\ \dots\ v_b]$. We adhere to the convention that all vectors are column vectors.
- For a matrix $M \in \mathbb{R}^{m \times n}, i \in [m], j \in [n]$: M_{ij} (or $[M]_{ij}$) denotes the (i, j)-th entry. $M_{i,:}$ and $M_{:,j}$ denote the i-th row and j-th column, respectively. Importantly, we note the convention that this "slice" notation converts all vectors into column vectors.
- When the first dimension of a matrix $M \in \mathbb{R}^{T \times d}$ is to be interpreted as a sequence length, we will implicitly convert a sequence of vectors $v_1, \dots, v_T \in \mathbb{R}^d$ into a matrix $(v_1, \dots, v_T) \in \mathbb{R}^{T \times d}$ whose *rows* the vectors v_t. This is an arbitrary choice (compared to concatenating columns and obtaining a matrix in $\mathbb{R}^{d \times T}$), selected to adhere to previously standardized notation for the Transformer.
- We will sometimes index vectors and matrices with named indices (such as $\perp$ for padding tokens) instead of integers, for clarity.
- e_i denotes the i-th elementary (one-hot) unit vector. Likewise as above, we sometimes use non-integer indices (e.g. $e_\perp$).
- For vectors $u, v \in \mathbb{R}^d$, $\langle u, v \rangle = u^\top v$ both denote the inner product.
- For a function $f : X \times Y \to Z$ and all $y \in Y$, we will let $f(\cdot, y) : X \to Z$ denote the restriction of f to y (and similarly for other restrictions). This appears in the per-input state transition functions $\delta(\cdot, \sigma) : Q \to Q$, as well as the functions represented by neural networks for a particular choice of weights.
- For functions f, g, $f \circ g$ denotes composition: $(f \circ g)(x) := f(g(x))$. When we compose neural networks $f : X \times \Theta_f \to Y, g : Y \times \Theta_g \to Z$ with parameter spaces Θ_f, Θ_g, we will use $f \circ g : X \times (\Theta_f \times \Theta_g) \to Z$ to indicate the composition $f(g(x; \theta_g)\theta_f)$.

- $(\cdot)_+ : \mathbb{R} \to \mathbb{R}$ denotes the ReLU (a.k.a. positive part) function: $(x)_+ = \max\{x, 0\}$. In function compositions, we use σ to denote the entry-wise ReLU (e.g. $f \circ \sigma \circ g$).

3.3.1 Automata, semigroups, and groups

A semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ has a state space Q, an input alphabet Σ, and a transition function $\delta : Q \times \Sigma \to Q$. For any natural number T and a starting state $q_0 \in Q$, by repeated composition of the transition function δ, one can use $\mathcal{A}$ to define a map from a sequence of inputs $(\sigma_1, \dots, \sigma_T) \in \Sigma^T$ to a sequence of states $(q_1, \dots, q_T) \in Q^T$ via:

$$q_t := \delta(q_{t-1}, \sigma_t), \forall t \in [T].$$

Here and below, it is helpful to use a matrix-vector notation to express the computation of semiautomata. For a given semiautomaton we can always identify the state space Q with index set $\{1, \dots, |Q|\}$ and use a one-hot encoding of states into $\{0,1\}^{|Q|}$. For each input symbol $\sigma \in \Sigma$, we associate a transition matrix $\delta(\cdot, \sigma) \in \{0,1\}^{|Q| \times |Q|}$ with entries $[\delta(\cdot, \sigma)]_{q', q} = \mathbf{1}\{\delta(q, \sigma) = q'\}$. This implies that for all q, σ, we have $e_{\delta(q,\sigma)} = \delta(\cdot, \sigma) e_q$, so that the computation of the semiautomaton amounts to repeated matrix-vector multiplication.

While semiautomata are remarkably expressive, we discuss a few simple examples throughout this background section to elucidate the key concepts.

Example 1 (Parity)**.** *Let $Q = \Sigma = \{0, 1\}$ and let $\delta(q, 0) = q$ and $\delta(q, 1) = 1 - q$. Then, starting with $q_0 = 0$, the state at time t, q_t, is 1 if the binary sequence $(\sigma_1, \dots, \sigma_t)$ has an odd number of 1s.*

Example 2 (Flip-flop)**.** *Let $Q = \{1, 2\}, \Sigma = \{\perp, 1, 2\}$ and let δ be given by*

$$\delta(\cdot, \perp) = I_{2\times 2}, \quad \delta(\cdot, 1) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad \delta(\cdot, 2) = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$$

As the name suggests, this semiautomaton implements a simple memory operation where the state at time t is the value of the most recent non-$\perp$ input symbol.

Example 3 (1D gridworld)**.** *Let S be a natural number, $Q = \{0, 1, \dots, S\}$ and $\Sigma = \{L, \perp, R\}$. Then the transition matrices are given by:*

$$\delta(\cdot, \perp) = I_{S+1 \times S+1}, \quad \delta(\cdot, L) = \begin{pmatrix} 1 & \ddots & & \\ 0 & & I_{S\times S} & \\ \vdots & & & \ddots \\ 0 & \dots & \dots & 0 \end{pmatrix} \quad \delta(\cdot, R) = \begin{pmatrix} 0 & \dots & \dots & 0 \\ \ddots & & & \vdots \\ & I_{S\times S} & & 0 \\ & & \ddots & 1 \end{pmatrix}.$$

This semiautomaton describes the movement of an agent along a line segment where actions -1 and $+1$ correspond to decrementing and incrementing the state respectively, except that the decrement input has no effect at state 0 and the increment input has no effect at state S.

Note that we have chosen a convention which differs slightly from what will be used in the discussion of main results (i.e. Figure 3.1): we enumerate the indices starting from 0 rather than 1. This is because the proofs are stated more naturally when the boundaries of the gridworld are identified with the indices 0 and S.

For a semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ each input symbol $\sigma \in \Sigma$ defines a function $\delta(\cdot, \sigma) : Q \to Q$. These functions can be composed in the standard way, and we use $\delta(\cdot, \sigma_{1:t})$ to denote the t-fold function composition. Note that $\delta(q_0, \sigma_{1:t})$ is precisely the value of the state at time t on input $\sigma_{1:t}$. Thus, the set of all functions that can be obtained by composition of the transition operator, formally

$$\mathcal{T}(\mathcal{A}) := \{\delta(\cdot, \sigma_{1:t}) : t \in \mathbb{N}, \sigma_{1:t} \in \Sigma^t\},$$

plays a central role in describing the computation of the semiautomaton. This object is a *transformation semigroup*. We now turn to describing the necessary algebraic background.

Recall that a *group* $(\mathcal{G}, \cdot)$ is a set $\mathcal{G}$ equipped with a binary operation $\cdot : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ such that

- (*identity*) There exists an identity element $e \in \mathcal{G}$ such that $e \cdot g = g \cdot e = g$ for all $g \in \mathcal{G}$.
- (*invertibility*) Every element $g \in \mathcal{G}$ has an inverse $g^{-1} \in \mathcal{G}$ such that $g \cdot g^{-1} = g^{-1} \cdot g = e$
- (*associativity*) The binary operation is associative: $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.

A *monoid* is less structured than a group; there must be an identity element and the binary operation must be associative, but invertibility is relaxed. A *semigroup* is even less structured: the only requirement is that the binary operation is associative.

It is common to let $\mathcal{G}$ be a subset of functions from $Q \to Q$ where Q is some ground set and let the binary operation be function composition. In this case, the structure is called a *permutation group* or *transformation monoid/semigroup* depending on which subset of the above properties hold. For transformation groups, since every element has an inverse under function composition, it is immediate that every element is some permutation over the ground set.

In fact, taking $\mathcal{G}$ to be a subset of functions as above is without loss of generality: by Cayley's theorem every group is isomorphic (equivalent after renaming elements) to a transformation group on some ground set, and we can take the ground set to have the same number of elements as the original group (for finite groups). Analogously, all semigroups are isomorphic to a transformation semigroup, but the ground set may need one additional element (for the identity); this is Cayley's theorem for semigroups. It is also clear that every transformation semigroup can be realized by some semiautomaton by trivially having the input symbols correspond to the functions in $\mathcal{G}$.[11] Therefore we have lost no structure when passing from finite semiautomata to finite semigroups.

Before discussing the compositional structure of semigroups, we give one more canonical example.

Example 4 (Cyclic group). *Let S be a natural number let $Q = \{0, 1, \ldots, S-1\}$ and let $\Sigma = \{1\}$ have only one element. The dynamics are given by $\delta(q, 1) = (q+1) \mod S$. Clearly this semiautomaton implements counting modulo S. The underlying group is the* cyclic group, *denoted C_S, which is isomorphic to the integers mod S with addition as the binary operation. Note that in this case, the operation is commutative, which makes the group* abelian.

Let us now turn to the compositional structure of groups and semigroups. Since it is without loss of generality to consider transformation (semi)groups, we always take the binary operation to be function composition. A *subgroup* H of a group G is a subset of the elements of G that is also a group,

[11]More succinctly, inputs can correspond to a generating set of the group, but this is not relevant for our results.

denoted as $H \leq G$. In particular it must be closed under the binary operation. N is a *normal subgroup*, denoted $N \triangleleft G$, if in addition to being a subgroup, it satisfies that $\{gn : n \in N\} = \{ng : n \in N\}$. (These sets are known as the left and right cosets of N in G and denoted gN and Ng respectively.) [12] Normal subgroups can also arise as the kernel of a mapping from G to a subgroup H of G. Let $\phi : G \to H$ be a mapping that preserves the group operation (i.e., a group homomorphism) and let $\ker(\phi) := \{g : \phi(g) = \mathrm{id}\}$. Then $\ker(\phi)$ is a normal subgroup of G. We will see below that normal subgroups provide a weak form of commutativity, that allows us to construct more complex groups out of simpler ones.

Direct products. The most natural way to compose larger groups from smaller ones is via the *direct product*. Given two groups G and H, we can form a new group with elements $\{(g,h) : g \in G, h \in H\}$ with a binary operation that is applied component-wise $(g,h) \cdot (g',h') = (g \cdot g', h \cdot h')$ (here, $\cdot$ is overloaded to be the group operation for all three groups). This direct product group is denoted $G \times H$. In the context of permutation groups, say G is a permutation group over ground set Q_G and H is over ground set Q_H. Then $G \times H$ has ground set $Q_G \times Q_H$ and every function in $G \times H$ factorizes component-wise, i.e., every element in $G \times H$ is identified with a permutation $(q_G, q_H) \mapsto (g(q_G), h(q_H))$ where $g \in G, h \in H$.

Observe that $G \times H$ contains normal subgroups which are isomorphic to both G and H. To see this, take $N = \{(e_G, h) : h \in H\}$ where e_G is the identity element in G. Then since $ge_G = e_G g$ and since H is closed under its group operation, we have $(g,h)N = N(g,h)$ for all $(g,h) \in G \times H$. A symmetric argument shows that G is also a normal subgroup of the direct product.

Note that we can analogously define direct products in the absence of the group axioms, and thus for monoids and semigroups. This gives a natural construction of the semigroup corresponding to moving around both axes of a 2-dimensional rectangular gridworld, as a concatenation of two non-interacting 1-dimensional gridworlds:

Example 5 (2D gridworld)**.** *If G_S is the transformation semigroup of the 1-d grid world with $S+1$ states, then $G_S \times G_S$ corresponds to a 2-dimensional gridworld. A semiautomaton that yields this transformation semigroup has state space $Q = \{(i,j) : i,j \in \{0,\ldots,S\}\}$ and 5 actions: increment or decrement i or j, subject to boundary effects, or do nothing.*

The definition of direct product extends straightforwardly to more than two terms $G_1 \times G_2 \times \ldots \times G_n$; we identify the items with tuples $(g_1, g_2, \ldots, g_n)$.

Semidirect products. However, it is possible to compose larger groups so that one of the subgroups is *not* a normal subgroup. This operation is called a *semidirect product*, with the group law $(g,h) \cdot (g',h') = (g \cdot \phi_h(g'), h \cdot h')$ for some ϕ_h to be defined later. Observe that in the direct product $G \times H$, we have constructed the elements from ordered pairs $(g \in G, h \in H)$, lifting G and H into a shared *product space* (i.e., the Cartesian product of the underlying sets of G and H), defining the group operation as simply applying those of G and H separately.

In fact, there are other ways, to define the group operation in the product space, but a difficulty arises: we need to find other nontrivial multiplication rules on pairs (g,h), and we cannot take for

[12] An equivalent definition of a normal group is a subgroup N such that $g^{-1}ng \in N, \forall g \in \mathcal{G}, n \in N$.

granted that an arbitrary binary operation satisfies the group axioms. We would like to define other operations $(g, h) \cdot (g', h')$ which output an element of g and an element of h. An attempt would be to pick two arbitrary injective homomorphisms ϕ_G, ϕ_H which embed G and H into a "shared space," so that elements of G and H can be multiplied together:

$$(g, h) \cdot (g', h') := \phi_G(g) \cdot \phi_H(h) \cdot \phi_G(g') \cdot \phi_H(h').$$

However, we need to ensure that this group operation is closed. Since all elements of the group are of the form (g, h) where $g \in G$ and $h \in H$, we must find a pair $(\tilde{g}, \tilde{h})$ that yields the right hand side of the above display when embedded into the shared space via ϕ_G, ϕ_H. For this, the most natural choice is $\tilde{g} = g \cdot g'$ and $\tilde{h} = h \cdot h'$, and thus we must check that:

$$\begin{aligned} \phi_G(g \cdot g') \cdot \phi_H(h \cdot h') &= \phi_G(g) \cdot \phi_G(g') \cdot \phi_H(h) \cdot \phi_H(h') \\ &= \phi_G(g) \cdot \phi_H(h) \cdot \phi_G(g') \cdot \phi_H(h') = (g, h) \cdot (g', h') \end{aligned}$$

However, the middle equality may not hold, because $\phi_G(g')$ and $\phi_H(h)$ are not guaranteed to commute. (Observe that for the special case of $g \mapsto (g, e_H), h \mapsto (e_G, h)$, these two elements always commute, giving rise to the direct product.)

Eliding ϕ_G, ϕ_H and simply using g, h as elements of the shared space, a sufficient condition for this to hold is that $hg'h^{-1} \in G$, since then, for some $\tilde{g} \in G$,

$$(g, h) \cdot (g, h') = ghg'(h^{-1}h)h' = g(hg'h^{-1})hh' = g\tilde{g}hh',$$

which is of the form $\phi_G(\cdot) \cdot \phi_H(\cdot)$ since both G and H are themselves closed. This condition is precisely that G is a normal subgroup.

There is a degree of freedom here: for each pair h and g', we can choose which element of G is given by $hg'h^{-1}$. When we make this choice we must ensure all of the group axioms are preserved, e.g., when $h = e_H$ we should always have $e_H g' e_H = g'$. Suppose we make this choice and define $\phi_h : g \mapsto hgh^{-1} \in G$ (this ϕ is a homomorphism from $H \to \mathrm{Aut}(G)$, where $\mathrm{Aut}(\cdot)$ denotes the *automorphism group*, the group of bijections on G that preserve the group axioms, under composition). Then, these ordered pairs do indeed form a group, but the group operation is

$$(g, h) \cdot (g', h') = g\phi_h(g')hh'$$

This object is the semidirect product, and it is denoted $G \rtimes H$. Note that the choice of mapping ϕ is unspecified in the notation, and, in general, different choices of ϕ will yield different structures for the semidirect product.

Finally, when $G = N \rtimes H$, both N and H are subgroups of G, but N is also a normal subgroup. To see this, we need to check that $hN = Nh$ for any $h \in H$. This is equivalent to $hnh^{-1} \in N$ for each h, n, but we defined the group operation to be $hnh^{-1} = \phi_h(n) \in N$, specifically so this would hold. On the other hand, H may not be a normal subgroup, and in this sense the semidirect product is a generalization of the direct product (for which both subgroups are normal). However, when the mapping ϕ is trivial, that is $\phi_h(n) = n$ then both N and H are normal subgroups, and one can verify that in this case the semidirect product and direct product coincide.

Example 6 (Dihedral group)**.** *Consider a semiautomaton with* $Q = \{0, \ldots, S-1\} \times \{-1, +1\}$ *and input* *alphabet* $\Sigma = \{\text{advance}, \text{reverse}\}$. *The transitions are given by:*

$$\delta((s,b), \text{advance}) = (s + b \bmod S, b)$$
$$\delta((s,b), \text{reverse}) = (s, -b)$$

The transformation semigroup for this semiautomaton is $C_S \rtimes C_2$ where C_S is the cyclic group on S elements (cf. Example 4). C_2 has two elements, the identity e and one element h such that $hh = e$. C_S has S elements where each element g is a function that adds some number $k \in \{0, \ldots, S-1\}$ to the input modulo S. The inverse g^{-1} is naturally to subtract k to the input, modulo S. The homomorphism ϕ in the semidirect product is such that $\phi_e(g) = g$ and $\phi_h(g) = g^{-1}$.

Wreath products. We define one more type of product between groups N and H: the *wreath product* $N \wr H := (N \times \ldots \times N) \rtimes H$. This is a group containing $|N|^{|H|} \cdot |H|$ elements (rather than $|N| \cdot |H|$, like the direct and semidirect products). Intuitively, it is defined by creating one copy of N per element in H via the direct product, then letting H specify a way to *exchange* these copies. Formally, $N \wr H$ is the unique group generated by

$$(g_1, \ldots, g_{|H|}, h) \quad \forall g_i \in N, h \in H,$$

where

$$(g_1, \ldots, g_{|H|}, e_H) \cdot (g'_1, \ldots, g'_{|H|}, e_H) := (g_1 \cdot g'_1, \ldots, g_{|H|} \cdot g'_{|H|}, e_H) \quad \forall g_i, g'_i \in N,$$

and

$$(g_1, \ldots, g_{|H|}, e_H) \cdot (e_N, \ldots, e_N, h) := (g_{\pi_h(1)}, \ldots, g_{\pi_h(|H|)}, e_H) \quad \forall g_i \in N, h \in H, \tag{3.1}$$

where we have enumerated the elements of H in arbitrary order, such that each $\pi_h : [H] \to [H]$ is the permutation defined by right multiplication $h' \mapsto h'h$ (by convention).

To write this explicitly as a semidirect product $N \wr H := (N \times \ldots \times N) \rtimes H$, the homomorphism into the direct product's automorphism group $\phi : H \to \text{Aut}(N \times \ldots \times N)$ is given by 3.1: for each $h \in H$, ϕ is the automorphism defined by permuting the indices between the terms in the direct product, according to the permutation induced by right multiplication by h.

Example 7 (Rubik's Cube)**.** *A naive way to construct the Rubik's Cube is to assign labels $\{1, \ldots, 54\}$ to the stickers on the cube, and define the Rubik's Cube group G via the sticker configurations reachable by the 6 face turns (which each specify a permutation $\delta_L, \delta_R, \delta_U, \delta_D, \delta_B, \delta_F : [54] \to [54]$ of the stickers). This establishes G as a subgroup of S_{54}. First, notice that the 6 central stickers never move (so this is really improvable to S_{48}). Next, notice that the $24 = 8 \times 3$ vertex stickers never switch places with the $24 = 12 \times 2$ edge stickers. The vertex stickers form a subset of the wreath product $C_3 \wr S_8$, while the edge stickers form a subset of the wreath product $C_2 \wr S_{12}$. In all, this realizes G as a subgroup of a direct product of wreath products:*

$$G \leq (C_3 \wr S_8) \times (C_2 \wr S_{12}).$$

Among other consequences towards solving the Rubik's Cube, this gives an improved upper bound on the size of G (which turns out to still be off by a factor of 12, because of nontrivial invariants preserved by the face rotations, a.k.a. unreachable configurations).

Quotients, simple groups, and maximal subgroups. When N is a normal subgroup of G, the quotient group G/N is defined as $\{gN : g \in G\}$ with binary operation $(gN)(g'N) = (gg')N$. The fact that

N is a normal subgroup implies that this is a well defined group. We can also check that if $G = N \rtimes H$ then the quotient group G/N is isomorphic to H, which matches the intuition for multiplication and division.

A G group is *simple* if it has no non-trivial normal subgroups. Intuitively, a simple group cannot be factorized into components; this generalizes the fact that a prime number admits no non-trivial factorization. When G is not simple then it has a non-trivial normal subgroup, say N. We call N proper if $N \neq G$. We call a proper subgroup N *maximal* if there is no other proper normal subgroup $N' \triangleleft G$ such that $N \triangleleft N'$. Equivalently, N is a maximal proper normal subgroup if and only if G/N is simple. This is akin to extracting a prime factor from a number, since the quotient group G/N cannot be further factorized. We will revisit this idea of factorization when defining *composition series* and *solvable groups* in Section 3.4.4.3.2.

Group extensions. Finally, we provide some additional terminology related to these different notions of products, which provide a cleaner unifying language in which to state our constructions. Let N, H be arbitrary groups. Which groups G contain a normal subgroup isomorphic N, such that the quotient G/N is isomorphic to H? Such a group G is said to be an *extension* of N over H. The direct product $G = N \times H$ is known as the *trivial extension*. A semidirect product $G = N \rtimes H$ is known as a *split extension*. However, not all extensions are split extensions; the smallest example is the *quaternion group* Q_8, the group of unit quaternions $\{\pm 1, \pm i, \pm j, \pm k\}$ under multiplication ($i^2 = j^2 = k^2 = ijk = -1$), which cannot be realized as a semidirect product of smaller groups. In general, it is very hard to derive interesting properties of a group extension based on the properties of N and H. Fortunately, there *is* a characterization of general extensions. The Krasner-Kaloujnine *universal embedding theorem* [Krasner and Kaloujnine, 1951] states that all extensions G can be found as subgroups of the wreath product $N \wr H$. The proof of Theorem 11 essentially shows how to *implement* the different kinds of group extensions, given constructions which implement the substructures N, H. In the worst case, we will have to implement a wreath product.

3.3.2 Shallow circuit complexity classes

We provide an extremely abridged selection of relevant concepts in circuit complexity. For a systematic introduction, refer to [Arora and Barak, 2009]. In particular, we discuss each circuit complexity class and inclusion below:

$$\mathsf{NC}^0 \subset \mathsf{AC}^0 \subset \mathsf{ACC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1.$$

- NC^0 is the class of constant-depth, constant-fan-in, polynomial-sized $\mathsf{AND}/\mathsf{OR}/\mathsf{NOT}$ circuits. If a constant-depth Transformer only uses the constant-degree sparse selection constructions in [Edelman et al., 2022], it can be viewed as representing functions in this class. However, the representational power of these circuits is extremely limited: they cannot express any function which depend on a number of inputs growing with T.

- AC^0 is the class of constant-depth, unbounded-fan-in, polynomial-sized AND/OR circuits, allowing NOT gates only at the inputs. A classic result is that the parity of T bits is not in AC^0 [Furst et al., 1984]; Hahn [2020] concludes the same for bounded-norm (and thus bounded-Lipschitz-constant) constant-depth Transformers.

- ACC^0 extends AC^0 with an additional type of unbounded-fan-in gate known as MOD_m for arbitrary number m, which checks if the sum of the input bits is a multiple of m. Theorem 2 comes from the fact that the semigroup word problem (which is essentially identical to semiautomaton simulation) is in this class; see [Barrington and Thérien, 1988].

- TC^0 extends AC^0 with an additional type of unbounded-fan-in gate called MAJ, which computes the majority of an odd number of input bits (a *threshold* gate). It is straightforward to simulate modular counters using a polynomial number of parallel thresholds (i.e. $\mathsf{ACC}^0 \subseteq \mathsf{TC}^0$). Whether this inclusion is strict (*can you simulate a threshold in constant depth with modular counters?*) is a salient open problem in circuit complexity. Threshold circuits are a very natural model for objects of interest in machine learning like decision trees and neural networks [Merrill et al., 2021].

- NC^1 is the class of $O(\log T)$-depth, constant-fan-in, polynomial-sized $\mathsf{AND}/\mathsf{OR}/\mathsf{NOT}$ circuits. It is an extremely popular and natural complexity class capturing *efficiently parallelizable* algorithms. It is unknown whether any of the inclusions in the "larger" classes $\mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{P}$ are strict.

3.4 Transformer's *shortcut* solutions to automata

In this section, we demonstrate that Transformer is uniquely suitable for implementing *shortcut*, which are $o(T)$-step solutions for T-step sequential reasoning problems. We describe two types of shortcuts. The first one takes $\log(T)$ steps, an exponential improvement over the T-step iterative solutions by RNN; The second one removes the dependency on T altogether, although it is only applicable to a (broad) subset of automata. We empirically show that approximate shortcuts can be found through finite-sample training in practice.

3.4.1 Theory: shortcuts abound

To simulate a semiautomaton at length T, a T-layer Transformer can implement the same sequential solution as an RNN: let the t-th layer embed the state transition $q_{t-1} \mapsto q_t$. We define *shortcuts* as solutions which implement the same functionality with a significantly smaller depth.

Definition 3 (Shortcut solution)**.** *Let $\mathcal{A}$ be a semiautomaton. For every $T \geq 1$, let f_T be a sequence-to-sequence neural network which simulates $\mathcal{A}$ at length T. Then, we call this sequence $\{f_T\}_{T\geq 1}$ a* shortcut *to $\mathcal{A}$ if the sequence of network depths $D := \{D(f_T)\}_{T\geq 1}$ satisfies $D \leq o(T)$.*

By this definition, shortcuts are quite general, and some are less interesting than others. For example, it is always possible to construct a constant-depth neural network which memorizes all $|\Sigma|^T$ values of $\mathcal{A}_{T,q_0}$, but these networks must be exceptionally wide. There are also solutions which emulate transitions in "chunks", letting each of (say) $\sqrt{T}$ layers perform $\sqrt{T}$ consecutive state transitions; however, without exploiting the structure of the semiautomaton, this would require width $\Omega(|\Sigma|^{\sqrt{T}})$. To rule out these cases and focus on interesting shortcuts for Transformers, we want the other size parameters (attention and MLP width) to be small: say, scaling at most polynomially in T, $|Q|$, and $|\Sigma|$. To construct such shortcuts, we need ideas beyond explicit iteration of state transitions.

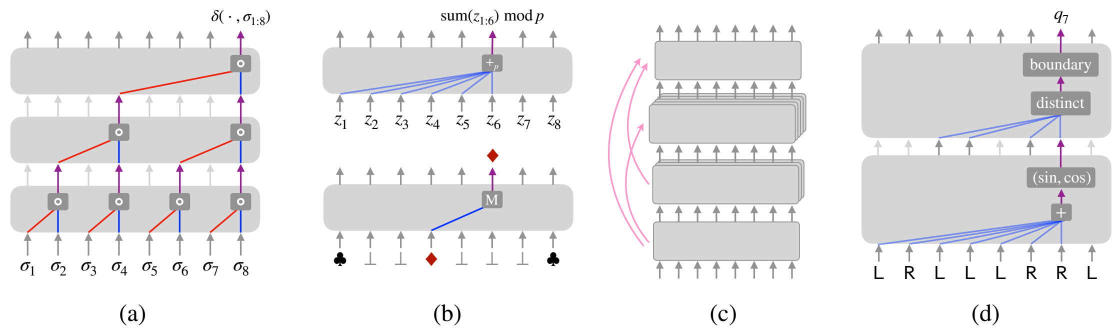


Figure 3.4: Intuitions for the theoretical constructions. *(a)* Divide-and-conquer function composition yields logarithmic-depth shortcuts (Theorem 10). *(b)* The two "atoms" of the constant-depth Krohn-Rhodes decomposition (Theorem 11) of a solvable semiautomaton: *modular addition* and sequentially resettable *memory*. *(c)* Information flow of the *cascade product*, which is used to glue these atoms together, and easily implemented with residual connections. *(d)* An even shorter shortcut solution for gridworld simulation (Theorem 12; see Appendix 3.4.4.4).

3.4.1.1 Upper bounds: $O(\log T)$-layer and $\tilde{O}(|Q|^2)$-layer constructions

We begin by noting that polynomial-width shortcuts *always* exist. This may seem counterintuitive if we restrict ourselves to viewing a Transformer's intermediate activations as representations of states q_t, like the RNN solution. Instead, a Transformer can encode and hierarchically compose transformations $\delta(\cdot, \sigma) : Q \to Q$ (see Figure 3.4a), leading to far shallower solutions:

Theorem 10 (Simulation is generically parallelizable; informal)**.** *Transformers can simulate all semiautomata $\mathcal{A} = (Q, \Sigma, \delta)$ at length T, with depth $O(\log T)$, embedding dimension $O(|Q|)$, attention width $O(|Q|)$, and MLP width $O(|Q|^2)$.*

This is proven in Section 3.4.4.2, and leverages the ability of a self-attention head to approximate *hard* attention (i.e. concentrate its mixing weights on a single position). However, self-attention heads can also perform *soft* attention (i.e. depend on a large number of previous positions), enabling even shallower implementations of certain sequential computations. For example, the parity automaton can be simulated by a single Transformer layer (see Lemma 15): soft attention computes prefix sums in parallel, then the MLP computes "mod 2". This leads to a significantly more nuanced question: *when are there even shallower shortcuts?* At first glance, such solutions may seem rare, and specialized to simple cases such as parity.

Our resolution to this question comes from the Krohn-Rhodes decomposition theorem [Krohn and Rhodes, 1965], a landmark result which vastly generalizes the uniqueness of prime integer factorizations, and created the mathematical field of algebraic automata theory [Rhodes et al., 2010]. The conclusion is quite unintuitive: allowing for both hard and soft modes of attention, constant-depth shortcuts are surprisingly common!

Theorem 11 (Transformer Krohn-Rhodes; informal)**.** *Transformers can simulate all solvable*[13] *semiautomata $\mathcal{A} = (Q, \Sigma, \delta)$, with depth $O(|Q|^2 \log |Q|)$, embedding dimension $2^{O(|Q| \log |Q|)}$, attention width $2^{O(|Q| \log |Q|)}$, and MLP width $|Q|^{O(2^{|Q|})} + 2^{O(|Q| \log |Q|)} \cdot T$.*

[13]See Definition 8. Intuitively, the only obstructions are when the semiautomata contain non-solvable groups such as S_5, the group of all permutations of 5 elements.

The proof is provided in Section 3.4.4.3, with a user-friendly exposition of the relevant algebraic concepts in Section 3.3.1. A few high-level notes:

- Intuitively (illustrated in Figures 3.4b and 3.4c), the Krohn-Rhodes decomposition "factorizes" every solvable semiautomaton into modular counters and memory units, glued together via a feedforward cascade product (Definition 6) whose depth only depends on $|Q|$, not T. These two types of "prime" semiautomata can be efficiently simulated by depth-1 Transformers.
- The decomposition depends on the transformation semigroup $\mathcal{T}(\mathcal{A})$. It is non-constructive (much like how the existence of prime factorizations doesn't entail a procedure to *find* them). Computationally, these solutions still have to be found by a search procedure. Remarkably, we find that gradient-based training succeeds empirically, despite the worst-case computational hardness of related problems.

What makes Transformers special (vs. other universal function approximators)? The same underlying semiautomaton-to-circuit constructions could be applied to any universal function approximator (like a vanilla MLP with the same depth). Transformers embed all of the constructions in Theorems 10 and 11 with exceptional efficiency, in terms of the network complexity measures discussed in Section 3.3. Most importantly, the constructions leverage Transformers' positional weight sharing, which removes all[14] suboptimal T factors from the parameter count. We discuss this further in Appendix 3.7.

Even shallower shortcuts, beyond Krohn-Rhodes. Finally, we show that on a natural class of problems, the computational model of self-attention leads to further fine-grained improvements over the guarantees of Krohn-Rhodes theory. Motivated by the application of Transformers in modeling environment dynamics, we consider the semiautomaton Grid_n corresponding to a "gridworld": n states on a line, with inputs "move left if possible" and "move right if possible" (see Figure 3.1, middle). We show that self-attention enables an extremely concise solution, with depth independent of both T *and* $|Q| = n$:

Theorem 12 (Depth-2 shortcut for gridworld; informal)**.** *For all positive integers n, T, Transformers can simulate* Grid_n *at length T, with depth 2,*[15] *embedding dimension $O(1)$, attention width $O(n)$, and MLP width $O(T)$.*[16]

The proof builds a parallel *nearest boundary detector* for the two boundary (i.e. leftmost and rightmost) states, and can be found in Section 3.4.4.4. We note that gridworlds are known to be extremal cases for the holonomy decomposition in Krohn-Rhodes theory (Maler [2010] discusses this, calling it the *elevator automaton*). It would be interesting to generalize our improvement and characterize the class of problems for which self-attention affords $O(1)$ instead of $\mathrm{poly}(|Q|)$-depth solutions.

[14]The only part of Theorem 11 requiring T non-tied neurons is the implementation of mod-p gates. It disappears entirely if we can add auxiliary MLP neurons with periodic activation functions such as $x \mapsto \sin(x)$.

[15]This requires max-pooling. If we do not use max-pooling, we can instead use an MLP with width $2^{O(n)}$ and depth $O(1)$, or width $O(n)$ and depth $O(\log n)$.

[16]As with Theorem 11, the width can be reduced to $O(n)$ if we employ periodic activation functions.

3.4.1.2 Lower bounds: $\Omega(\log T)$ layers required, assuming $\mathsf{TC}^0 \neq \mathsf{NC}^1$

Can Theorem 11 be improved to handle non-solvable semiautomata? (Equivalently: can Theorem 10 be improved to constant depth?) It turns out that as a consequence of a classic result in circuit complexity [Barrington, 1986], this question is equivalent to the major open question of $\mathsf{TC}^0 \stackrel{?}{=} \mathsf{NC}^1$ (thus: conjecturally, no). Unless these complexity classes collapse, Theorems 10 and 11 are optimal. In summary, simulating non-solvable semiautomata [17] with constant depth is provably hard:

Theorem 13 (Transformer Barrington). *Let $\mathcal{A}$ be a non-solvable semiautomaton. Then, for sufficiently large T, no $O(\log T)$-precision Transformer with depth independent of T and width polynomial in T can continuously simulate $\mathcal{A}$ at length T, unless $\mathsf{TC}^0 = \mathsf{NC}^1$.*

Proof. This follows straightforwardly from the fact that simulating $\mathcal{A}$ at length T is NC^1-complete under NC^0 reductions: given any $O(\log T)$-depth bounded-fan-in $\mathsf{AND}/\mathsf{OR}/\mathsf{NOT}$ circuit $\mathcal{C}$, and a depth-D circuit $\mathcal{C}'$ which simulates a semiautomaton whose transformation monoid contains a non-solvable subgroup, there is a procedure which generates a depth-$O(D)$ circuit to simulate $\mathcal{C}$; see [Barrington and Thérien, 1988]. This in turn comes from the construction used in Barrington's theorem [Barrington, 1986], which characterizes NC^1 as exactly the set of languages recognizable by *bounded-width branching programs*. For a closely related reference which follows almost exactly the same argument, see [Mereghetti and Palano, 2000].

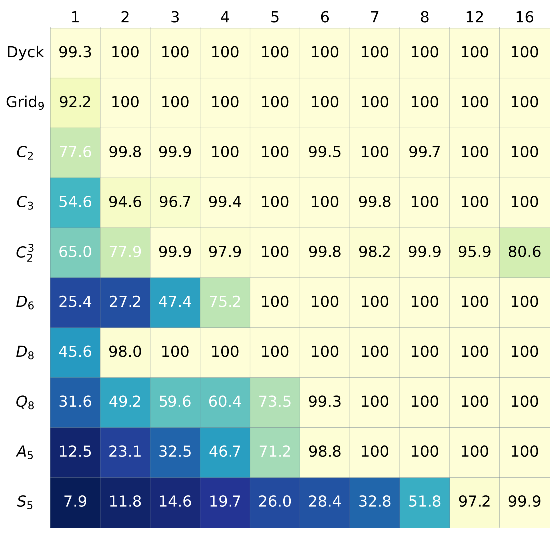
Thus, it suffices to show that a constant-depth Transformer is in TC^0. The details of manipulating floating-point numbers with discrete circuits are peripheral to the main results in this paper, so we provide a brief proof sketch. A similar argument is used by Merrill et al. [2021] to establish that *"saturated"* Transformers (a multi-index analogue of hard-attention Transformers), with $O(\log T)$ bit precision, can be represented with a TC^0 circuit. We outline a proof (which applies to the formal setting considered by Merrill et al. [2021]) for the notion of Transformers defined in this paper.

With $O(\log T)$ bits of precision, all n-way (including unary) arithmetic operations mapping $\mathbb{R}^n \to \mathbb{R}$ can be represented with a constant-depth, $\mathrm{poly}(T)$-width AC^0 circuit, as long as n does not depend on T. Although improvements are certainly possible, it suffices to consider the circuit which memorizes the i-th bit of the output, which has width $2^{n \log T} \leq O(\mathrm{poly}(T))$. Thus, the position-wise non-interacting matrix operations (multiplication by $X \mapsto W_Q X$, etc., the feedforward MLP layers, and the encoding and decoding layers) can be simulated with $\mathrm{poly}(T)$ width.

The only subtlety arises when there is a T-way summation over $O(\log T)$-bit numbers, which occur in the softmax and attention mixture layers. For this operation, we can use the construction from [Reif and Tate, 1992], which can even add T $\mathrm{poly}(T)$-bit numbers in TC^0. □

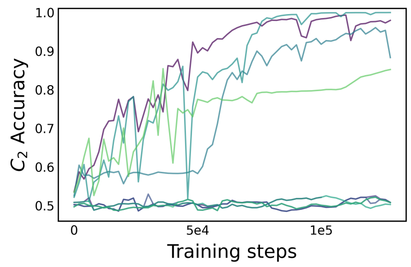
Finally, we note that although our width bounds might be improvable, an exponential-in-$|Q|$ number of hypotheses (and hence a network with $\mathrm{poly}(|Q|)$ parameters) is unavoidable if one wishes to learn an arbitrary $|Q|$-state semiautomaton from data: there are $|Q|^{|Q| \cdot |\Sigma|}$ of them, which generate $|Q|^{\Omega(|Q|^2)}$ distinct semigroups [Kleitman et al., 1976]. If we wish to study how machine learning models can efficiently identify large algebraic structures, we will need finer-grained inductive biases to specify which semiautomata to prefer, a direction for future work.

[17] The smallest example of a non-solvable semiautomaton has $|Q| = 60$ states, whose transitions generate A_5 (all of the even permutations).

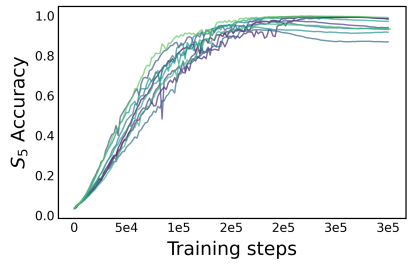


	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid$_9$	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

(a) Accuracy across tasks (rows) and network depths (columns); details in Appendix 3.4.5.1.1.



(b) Training curves for C_2 (i.e. parity; 10 replicates).



(c) Training curves for S_5 (10 replicates).

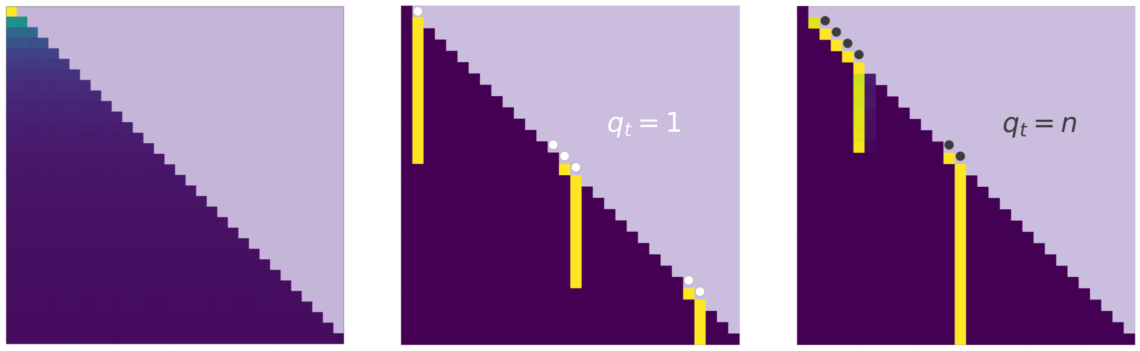
Figure 3.5: Overview of the empirical results in Section 3.4.2, on in-distribution learnability of shortcuts by standard Transformer training. *(a)* Truncated table of results (in-distribution accuracy); rows specify semiautomaton simulation problems, and columns specify network depth. *(b),(c)* Training is highly unstable.

3.4.2 Empirical shortcuts found by SGD

The results in Section 3.4.1.1 only provide a precise understanding of *representability*: they show that shortcut solutions exist within the parameter space of a shallow Transformer. To understand whether Transformers can actually learn these shortcuts from data, we must introduce the additional considerations of *generalization* and *optimization*. It is notoriously difficult to derive meaningful analyses which account for all of these factors in deep learning; thus, we do not attempt to do so in this work.[18] Instead, we approach the end-to-end question with an empirical lens: trained on sequences arising from a variety of automata, does a shallow (depth-$L \ll T$) Transformer converge to correct simulators of these automata?

For a selection of 19 semiautomata corresponding to various groups and semigroups (detailed descriptions in Section 3.4.5.1.1), we train shallow Transformer (GPT-2-like [Radford et al., 2019a]) models to map randomly sampled sequences $(\sigma_1, \ldots, \sigma_T)$ to their corresponding state sequences $(q_1, \ldots, q_T)$, and evaluate their accuracy on held-out sequences. We vary the depth L from 1 to 16, and use freshly-sampled sequences of length $T = 100$. In this setup, the number of sequences encountered during training ($\leq 10^6$) is far smaller than the number of distinct input sequences ($|\Sigma|^{100}$). Thus, brute-force memorization cannot solve this task, and generalization is necessary to achieve nontrivial performance.

[18]Our bounds on the parameter count and weight norms do imply classical generalization bounds for appropriately norm-constrained Transformers [Edelman et al., 2022], but these are too coarse-grained to provide non-vacuous predictions of generalization behavior.



(a) 1st layer, uniform attention (b) 4th layer, left boundary detector (c) 4th layer, right boundary detector

Figure 3.6: Attention heads implement a *nearest boundary detector* for the 1-dimensional gridworld task (see Figure 3.1 and Theorem 12): the lower triangle shows the causal attention patterns (the upper triangle is masked out, hence all 0), where a brighter color corresponds to a higher attention score. Positions for the actual boundaries are marked by white (for the left boundary i.e. state 1) or gray (for the right boundary, i.e. state n) dots. This shows that our theoretical construction in Theorem 12 agrees with the solutions found in practice.

Strikingly, we obtain positive results ($>99\%$ in-distribution accuracy[19]) for *every* finite-state semiautomaton we considered, including ones which generate the non-solvable groups A_5 and S_5. Figure 3.5 gives a selection of our full results (in Section 3.4.5.1). We find that more complex semiautomata (corresponding to non-abelian groups) require deeper networks to learn, in agreement with our theoretical constructions.

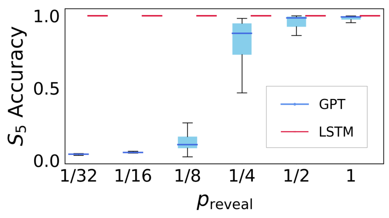
What about the small fraction of mistakes? Our theoretical results show that there are logarithmic-depth (S_5, A_5) and constant-depth (all the others) solutions which simulate these semiautomata with exactly 100% accuracy. Of course, with such long sequences, black-box evaluation of whether this accuracy is reached in the population distribution is computationally infeasible. However, we note that without periodic activations (or some other mechanism for extrapolating to unseen count values), our theoretical constructions require MLPs to memorize the mod-n function. This will be revisited in the out-of-distribution evaluation experiments in Section 3.4.3.2, but there is even a corresponding implication for in-distribution mistakes: if a model never sees certain "outlier" counts during training, it is expected to make mistakes on those outliers when they appear in evaluation.

Which shallow solutions are learned? Our theoretical results identify shortcut solutions which follow multiple, mutually incompatible paradigms. In general, we do not attempt a full investigation of *mechanistic interpretability* of the trained models. In particular, we do not claim that the networks discover implementations are isomorphic to those described in the proofs of Theorem 10, 11, and 12. However, as a preliminary exploration, we visualize some of the attention patterns in Figure 3.6 within successfully-trained models, finding attention heads which perform *flat summations* (with uni-

[19]Our primary goal is to understand if gradient-based training can find shortcut solutions at all, rather than whether such training is stable. Accordingly, unless otherwise noted, we report the performance of the *best* model among 20 replicates. See Section 3.4.2 for details and sensitivity analyses.

Task	$\text{Dyck}_{4,8}$	Grid_9	S_5	C_4	D_8
Observation	stack top	$\mathbb{1}_{\text{boundary}}$	$\pi_{1:t}(1)$	$\mathbb{1}_{0 \bmod 4}$	location
Accuracy	100.0	100.0	99.6	99.9	100.0

(a) Accuracies with indirect supervision (details in Appendix 3.4.5.2.1). LSTM gets 100% on all tasks.



(b) Varying p_{reveal} (log spacing).

Figure 3.7: Overview of the empirical results in Section 3.4.3.1. *(a)* Learning in the latent-state setting, with various observation maps $\varphi(q_t)$. *(b)* Learning from incomplete state sequences: final accuracy vs. position-wise probability of a hidden token, for GPT and LSTM; the mean and standard deviations are taken over 25 runs.

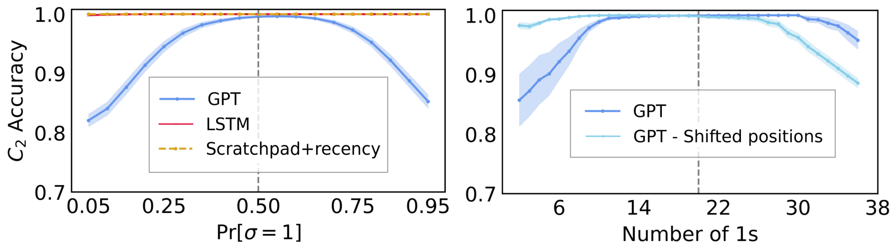


Figure 3.8: OOD generalization on C_2 (parity): Transformers fail to generalize to different distributions (*left*) because shortcuts fail to generalize to unseen counts (*right*; the 1s are uniformly distributed in the sequence). In contrast, recurrent solutions (LSTM, and Transformer with recency-biased scratchpad training) maintain perfect accuracy.

form attention) and *conditional resets*, agreeing with the construction in Theorem 12.

Optimization instability. Although sufficiently deep networks find the solutions with non-negligible probability, the training dynamics are unstable; Figure 3.5b,c show example training curves, which exhibit high variance, negative progress, or accuracy that decays with continued training. In the same vein as the "synthetic reasoning tasks" introduced by Zhang et al. [2022], we hope that semiautomaton simulation will be useful as a clean, nontrivial testbed (with multiple difficulty knobs) for debugging and improving training algorithms, and perhaps the neural architectures themselves. More details are deferred to Appendix 3.4.5.1.1.

3.4.3 Experiments under more challenging settings

The results from Section 3.4.1 and Section 3.4.2 show that Transformers can learn shortcuts end-to-end, unobstructed by depth, generalization, or optimization. However, the experiments in Section 3.4.2 are idealized in several ways; a natural question is whether these findings are robust to various challenges that arise in practice. In this section, we investigate the robustness of the shallow Transformer solutions, compared to those found by RNNs (the "natural" architecture for simulating semiautomata). We consider harder forms of supervision (Section 3.4.3.1) and evaluation (Section 3.4.3.2); details are deferred to Section 3.4.5.2.

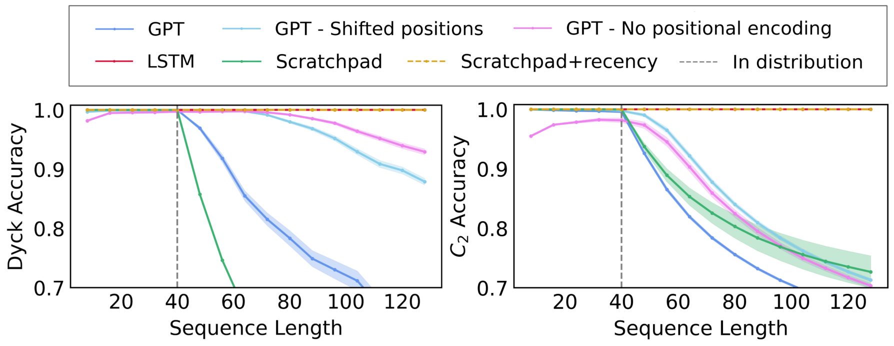


Figure 3.9: Length generalization on Dyck (*left*) and C_2 (*right*): Transformers fail to generalize to longer sequences, but can be improved by modifying positional encodings. In contrast, recurrent solutions (LSTM, and Transformer with recency-biased scratchpad training) maintain perfect accuracy.

3.4.3.1 Incomplete and indirect supervision

Successful learning with partial observations.

Consider the case of *partial observability*. For any semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ and a (generally non-invertible) *observation* function $\varphi : Q \to \tilde{Q}$, we can define the problem of predicting $\tilde{q}_t := \varphi(q_t)$. If we can only obtain observations $\tilde{q}_t$ (i.e., the state is latent), this fully captures the problem of learning a finite-state *automaton* from data. The results in this paper have shown that this is equivalent to the fully-observable case in terms of *representation*. However, the *learning* problem can be much harder; indeed, this may account for Bhattamishra et al. [2020a]'s negative results on learning regular languages with constant-depth Transformers. Note that this also captures autoregressive next-token prediction tasks induced by distributions (e.g., generating Dyck languages [Yao et al., 2021b]) where the sequence's continuations depend on a latent semiautomaton's state (e.g., the current stack for Dyck). Despite these potential challenges, we find that Transformers are able to find solutions with good in-distribution performance for all partially observable settings we consider; see Figure 3.7a.

Learning from incomplete state sequences: RNNs are better. Next, we consider the setting which is identical to that described in Section 3.4.2, but each state q_t is randomly revealed from the training data with some probability $0 \leq p_{\text{reveal}} \leq 1$. As with partial observability, this does not affect representation issues, but can make learning/optimization much harder. Figure 3.7b shows the accuracy of S_5 for models trained on length 100 sequences for various p_{reveal}. It can be seen that Transformers may be unable to find good solutions when the labels become sparser, whereas LSTM's performance stays robust across all choices of p_{reveal}, potentially due to a more favorable recurrent inductive bias [Abnar et al., 2020].

3.4.3.2 Out-of-distribution shortcomings of shortcut solutions

Out-of-distribution generalization: RNNs are better. The theoretical construction of modular counters (Lemma 15) suggests a possible failure mode: if attention performs prefix addition and the MLP computes the sum modulo n, the MLP could fail on sums unseen during training. This suggests that if the distribution over $\sigma_{1:T}$ shifts between training and testing (but the semiautomaton remains the same), a non-recurrent shortcut solution might map inputs into an intermediate latent variable space (like the sum) which fails to generalize.

Indeed, we observe that with the same models which obtain the positive in-distribution results in Section 3.4.2, accuracy degrades as distribution shift increases; see Figure 3.8 *(left)*, where the performance drops as the probability of seeing input $\sigma = 1$ deviates from the training distribution ($\Pr[\sigma = 1] = 0.5$). From the viewpoint of mechanistic interpretation, one possible explanation is that Transformers learn shortcuts that calculates parity by first counting the number of 1s in the input sequence then computing modulo 2, and hence struggle to deal with sequences where the count is less frequently seen during training. To verify this hypothesis, we further compare the accuracy against number of 1s in the sequence (Figure 3.8 *(right)*); details are deferred to Section 3.4.5.2.3.

Length generalization: RNNs are better. More ambitiously, we could try to use these models to extrapolate to longer sequence lengths T than those seen in the training data. Promoting this difficult desideratum of *length generalization* is an intricate problem in its own right; see Yao et al. [2021b], Anil et al. [2022b] for more experiments similar to ours. Figure 3.9 shows the performance on sequences of various lengths. In contrast to LSTM's perfect performance on all scenarios, Transformer's accuracy drops sharply as we move to lengths unseen during training. This is not purely due to unseen values of the positional encoding: randomly shifting the positions during training can cover all the positions seen during testing, which helps improve the length generalization performance but cannot make it perfect; we see similar results for removing positional encodings altogether. Finally, we empirically show that the above flaws are circumventable. Using a combination of *scratchpad* (a.k.a. "chain-of-thought") [Nye et al., 2021a, Wei et al., 2022c] and recency bias [Press et al., 2022], we demonstrate that Transformers can be guided towards learning recurrent (depth-T) solutions, which generalize out-of-distribution and to longer sequence lengths (Figure 3.9, yellow curves). Details are deferred to Section 3.4.5.2.3.

Discussion: shortcuts as "unintended" solutions. Throughout the deep learning literature, the term *shortcut* is often used to refer to undesired (i.e., misleading, spurious, or overfitting) statistical properties of learned representations [Geirhos et al., 2020, Robinson et al., 2021]. Meanwhile, under our computational ($o(T)$ circuit depth) definition, shortcut solutions are perfectly valid ways to represent recurrent computations. The results in this section establish a connection between these notions: partially-learned computational shortcuts can be statistical shortcuts. Specifically, a non-recurrent architecture can "hallucinate" intermediate variables other than the state (e.g. the "count" variable for the parity automaton), is thus sensitive to the coverage of these variables in the training data. This leads to out-of-distribution generalization failures (e.g. on rare counts) which are not present in recurrent models. We will provide a detailed study in this topic in Section 3.5.

Computational-statistical tradeoffs. The experiments in this section highlight a statistical penalty for learning recurrent computations with a non-recurrent architecture. However, the computational advantage of a shallow architecture is extremely appealing: maximally leveraging parallel computation, training and inference can be done much faster ($O(\log T)$ or $O(1)$ time, compared to $O(T)$). This highlights a delicate tradeoff between RNNs and Transformers, where neither architecture dominates the other, even when considering this elementary class of algorithmic problems. Attaining the best of both worlds with a practical architecture is an interesting avenue for future work.

3.4.4 Proofs

3.4.4.1 Useful definitions and lemmas

Formal definitions of simulation. We first recall the notions of simulation introduced in Section 3.2:

- A *function* can simulate an automaton for particular choices of T, q_0. For a semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$, a function $f : \Sigma^T \to Q^T$ *simulates* $\mathcal{A}_{T,q_0}$ if $f(\sigma_{1:T}) = \mathcal{A}_{T,q_0}(\sigma_{1:T})$ for all input sequences $\sigma_{1:T}$. Here, the right-hand side denotes the sequence of states $q_{1:T}$ induced by the input sequence $\sigma_{1:T}$ under the transitions δ starting from state q_0.
- A *function class* can simulate multiple functions associated with a semiautomaton. For a semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ and a positive integer T, a function class $\mathcal{F}$ (a set of functions $f : \Sigma^T \to Q^T$) *simulates* $\mathcal{A}$ at length T if, for every $q_0 \in Q$, there is function $f_{q_0} \in \mathcal{F}$ which simulates $\mathcal{A}_{T,q_0}$.

Our proofs rely on composing "gadgets" which simulate various substructures of the transformation semigroup $\mathcal{T}(\mathcal{A})$. Thus, it will be useful to establish a third notion of simulation, which works for functions in the embedding space $\mathbb{R}^d$ rather than the symbol spaces Q, Σ. For clarity, we give this notion a different name (*continuous simulation*):

- For a semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$, a function $f : \mathbb{R}^d \to \mathbb{R}^d$ *continuously simulates* $\mathcal{A}_{T,q_0}$ if there exist functions $E : \Sigma \to \mathbb{R}^d, W : \operatorname{im} f \to Q$ such that $W \circ f \circ E$ simulates $\mathcal{A}_{T,q_0}$.

When W is a linear threshold function $z \mapsto \arg\max_q [Wz]_q$, this corresponds to a standard classification head. However, our constructions may leverage other encodings of discrete objects.

Function approximation. We provide some simple function approximation results below.

Lemma 10 (1D discrete function interpolation with an MLP)**.** *Let $\mathcal{X}$ be a finite subset of $\mathbb{R}$, such that $|x| \le B_x$ for all $x \in \mathcal{X}$, and $|x - x'| \ge \Delta$ for all $x \ne x' \in \mathcal{X}$. Let $f : \mathcal{X} \to \mathbb{R}^d$ be such that $\|f(x)\|_\infty \le B_y$ for all $x \in \mathcal{X}$. Then, there is a 2-layer ReLU network for which*

$$f_{\text{mlp}}(x + \xi; \theta_{\text{mlp}}) = f(x) \qquad \forall x \in \mathcal{X}, \quad |\xi| \le \Delta/4.$$

The inner dimension is $d' = 4|\mathcal{X}|$, and the weights satisfy

$$\|W_1\|_\infty \le \frac{4}{\Delta}, \quad \|b_1\|_\infty \le \frac{4B_x}{\Delta} + 2, \quad \|W_2\|_\infty \le B_y, \quad b_2 = 0.$$

Proof. For each $x_0 \in \mathcal{X}$, we construct an indicator $\psi_{x_0}(x)$ for x_0, out of 4 ReLU units. Letting $\Delta' := \Delta/4$, the construction is

$$\psi_{x_0}(x) := \left(\frac{x - (x_0 - 2\Delta')}{\Delta'}\right)_+ - \left(\frac{x - (x_0 - \Delta')}{\Delta'}\right)_+ \\ - \left(\frac{x - (x_0 + \Delta')}{\Delta'}\right)_+ + \left(\frac{x - (x_0 + 2\Delta')}{\Delta'}\right)_+ .$$

The second layer simply sums these indicators, weighted by each $f(x_0)$. □

Lemma 11 (General discrete function interpolation with an MLP)**.** *Let $\mathcal{X}$ be a finite subset of $\mathbb{R}^{d_{\text{in}}}$, such that $\|x\|_\infty \leq B_x$ for all $x \in \mathcal{X}$, and $\|x - x'\|_\infty \geq \Delta$ for all $x \neq x' \in \mathcal{X}$. Let $f : \mathcal{X} \to \mathbb{R}^{d_{\text{out}}}$ be such that $\|f(x)\|_\infty \leq B_y$ for all $x \in \mathcal{X}$. Then, there is a 3-layer ReLU network for which*

$$f_{\text{mlp}}(x + \xi; \theta_{\text{mlp}}) = f(x) \qquad \forall x \in \mathcal{X}, \quad |\xi| \leq \Delta/4.$$

Letting $\mathcal{X}_i$ denote the set of unique values in coordinate i, the inner MLP dimensions are as follows:

$$d_1 = 4 \sum_{i \in [d_{\text{in}}]} |\mathcal{X}_i|, \quad d_2 = |\mathcal{X}|.$$

The weights satisfy

$$\|W_1\|_\infty \leq \frac{4}{\Delta}, \quad \|b_1\|_\infty \leq \frac{4B_x}{\Delta} + 2, \quad \|W_2\|_\infty \leq 1, \quad \|b_2\|_\infty \leq d_{\text{in}}, \quad \|W_3\|_\infty \leq B_y, \quad b_3 = 0.$$

Proof. The first layer uses the same construction as that in Lemma 10, creating indicators for each $x \in \mathcal{X}_i$ for each i. For each $x \in X$, the second layer has an activation which sums the indicators from each x_i, with bias $-d_{\text{in}}$ (thus creating indicators for each x). The third layer outputs $f(x)$ for each indicator. □

When we apply Lemmas 10 and 11 in recursive constructions, and $B_x/\Delta \geq 1$, we will opt to use the bound $\|b_1\|_\infty \leq 6B_x/\Delta$, to reduce the clutter of propagating the 2 term without resorting to asymptotic notation.

We also introduce a simpler version of Lemma 10 for the special case of the threshold function $f(x) := \mathbb{1}[x > 0]$:

Lemma 12 (Threshold with an MLP)**.** *Let $\mathcal{X}$ be a subset of $\mathbb{R}$, and $|x| \geq \Delta$ for all $x \in \mathcal{X}$. Then, there is a 2-layer ReLU network for which*

$$f_{\text{mlp}}(x + \xi; \theta_{\text{mlp}}) = \mathbb{1}[x > 0] \qquad \forall x \in \mathcal{X}, \quad |\xi| \leq \Delta/4.$$

The inner dimension is $d' = 2$, and the weights satisfy

$$\|W_1\|_\infty \leq \frac{1}{\Delta}, \quad \|b_1\|_\infty \leq 1/2, \quad \|W_2\|_\infty \leq 1, \quad b_2 = 0.$$

Proof. We construct the threshold using 2 ReLU units. The construction is

$$\psi(x) := \left(\frac{x + \Delta}{2\Delta}\right)_+ - \left(\frac{x - \Delta}{2\Delta}\right)_+ .$$

□

Selection via soft attention. We record some useful lemmas pertaining to approximating hard coordinate selection with soft attention. The following is a simplified version of Lemma B.7 from [Edelman et al., 2022] (which generalizes this to multi-index selection):

Lemma 13 (Softmax approximates hard max). *Let $z \in \mathbb{R}^T$. Let $\mathsf{softmax}(z) : \mathbb{R}^T \to \mathbb{R}^T$ denote the T-dimensional softmax function:*

$$[\mathsf{softmax}(z)]_t := \frac{e^{z_t}}{\sum_{t' \in [T]} e^{z_{t'}}}.$$

Let $t^* := \arg\max_t z_t$. Suppose that for all $t' \neq t^*$, $z_{t'} \leq z_{t^*} - \gamma$. Then,

$$\|\mathsf{softmax}(z) - e_{t^*}\|_1 \leq 2T \cdot e^{-\gamma}.$$

Proof. Without loss of generality, $\max z = \gamma$ (since the softmax function is invariant under shifting all inputs by the same value), so that all other coordinates are non-positive. Also, assume $T \geq 2$ (the $T = 1$ case is trivial). We have

$$[\text{softmax}(z)]_{t^*} = \frac{e^\gamma}{e^\gamma + \sum_{t \neq t^*} e^t} \geq \frac{e^\gamma}{e^\gamma + T - 1} = 1 - \frac{T-1}{e^\gamma + T - 1} \geq 1 - \frac{T-1}{e^\gamma},$$

and for $t' \neq t$,

$$[\text{softmax}(z)]_{t'} = \frac{e^{t'}}{e^\gamma + \sum_{t \neq t^*} e^t} \leq \frac{1}{e^\gamma}.$$

Thus, the 1-norm of the difference is bounded by

$$\frac{T-1}{e^\gamma} + (T-1) \cdot \frac{1}{e^\gamma} < \frac{2T}{e^\gamma},$$

as claimed. □

Positional embeddings. We note the following elementary fact about 2-dimensional circular embeddings.

Proposition 2 (Circular embeddings). *Consider $p_1, \ldots, p_T$, the T equally-spaced points on the 2-dimensional circle:*

$$[p_t]_1 := \cos\left(\frac{2\pi t}{T}\right), \quad [p_t]_2 := \sin\left(\frac{2\pi t}{T}\right).$$

Then, for any $t \neq t'$,

$$|\langle p_t, p_{t'} \rangle| \leq 1 - \frac{2\pi^2}{T^2} < 1 - \frac{19.7}{T^2}.$$

3.4.4.2 Proof of Theorem 10: Logarithmic-depth shortcuts via parallel prefix sum

In this section, we give the full statement and proof of the universal existence of logarithmic-depth shortcuts.

Theorem 10 (Simulation is generically parallelizable). *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a semiautomaton, $q_0 \in Q$, and $T \geq 1$. Then, there is a depth-$\lceil \log_2 T \rceil$ Transformer which continuously simulates $\mathcal{A}_{T,q_0}$, with embedding dimension $2|Q| + 2$, MLP width $|Q|^2 + |Q|$, and ∞-weight norms at most $\max\{4|Q| + 2, 10T\sqrt{\log|Q| + \log T}\}$. It has $H = 2$ heads with embedding dimension $|Q|$ implying $2|Q| + 2$ attention width, and a 3-layer MLP.*

Proof. The basic idea is that all prefix compositions $\delta(\cdot, \sigma_t) \circ \ldots \circ \delta(\cdot, \sigma_1)$ can be evaluated in logarithmic depth using a binary tree whose leaves are the per-input transition functions $\delta(\cdot, \sigma) : Q \to Q$.

The attention heads select the pairs of functions that need to be composed, while the feedforward networks implement function composition. The network will manipulate functions in terms of their *transition maps*: for example, the encoding of $f := (1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 2)$ is

$$\sum_{q \in \{1,2,3\}} f(q) \cdot e_q = [1\ 1\ 2].$$

Small nuances. We will produce a construction for the case where T is a power of 2; general T can be handled via padding. To simplify the construction, we also introduce T padding positions $-(T-1), \ldots, 0$ at the beginning; while this greatly simplifies the positional selection construction, this padding construction could be replaced with a slightly more complicated MLP. Also, in this construction, we do not need to use residual connections; the parallel prefix sum algorithm we use can be executed "in place", saving a logarithmic factor in the width. We do assume access to the 2 positional embeddings at each layer; in the absence of residual connections, the identity function restricted to these 2 dimensions can be implemented by the MLP and attention heads.

Let $L = \log_2 T$ be the depth of the binary tree. We choose $d := 2|Q| + 2$. Instead of indexing the dimensions by $[d]$, we give them names:

- *Left function encoding* dimensions (q, L) for each $q \in Q$.
- *Right function encoding* dimensions (q, R) for each $q \in Q$.
- *Positional encoding* dimensions $\mathsf{P}_1, \mathsf{P}_2$.

Without loss of generality, let $Q = [|Q|] = \{1, \ldots, Q\}$ (selecting an arbitrary enumeration of the state space). Also, assume $|Q| \geq 2$ (if not, add a dummy state). We choose $E(\sigma_t) := \sum_{q \in Q} \delta(q, \sigma_t) \cdot e_{(q,\mathsf{R})}$, mapping each input symbol to the "transition map" of its transitions. At the padding positions $-(T-1), \ldots, 0$, we will encode the "go to q_0" function: $\sum_{q \in Q} q_0 \cdot e_{(q,\mathsf{R})}$.

Function composition gadget. We first introduce the construction for function composition with a 3-layer ReLU MLP, which will be used by all layers. It gives an exponential improvement over the generic universal function approximation gadget from Lemma 11.

Lemma 14. *There exists a 3-layer ReLU MLP $\phi_{\mathrm{mlp}} : \mathbb{R}^d \to \mathbb{R}^d$, with fixed parameters W_1, b_1, W_2, b_2, W_3 whose dimensions and weights only depend on Q, such that for all $f, g : Q \to Q$, ϕ_{mlp} outputs the transition map of $f \circ g$ given the concatenated transition maps of f and g. That is, for all $|Q|^{2|Q|}$ choices of f, g:*

$$\phi\left(\sum_{q \in Q} g(q) \cdot e_{(q,\mathsf{L})} + \sum_{q \in Q} f(q) \cdot e_{(q,\mathsf{R})}\right) = \sum_{q \in Q} (f \circ g)(q) \cdot e_{(q,\mathsf{R})}.$$

The intermediate dimensions are $d_1 = |Q|^2 + |Q|$ and $d_2 = |Q|^2$, and weight norms are bounded by $4|Q| + 2$.

Proof. The first layer uses Lemma 10 to create $|Q|^2$ indicators: one to recognize each value along the $e_{(q,\mathsf{L})}$ direction. Let us index these by $q, q' \in Q$. Then, this gives us $W_1 \in \mathbb{R}^{d \times 4|Q|^2}, b_1 \in \mathbb{R}^{4|Q|^2}, W_2' \in \mathbb{R}^{|Q|^2}$ such that

$$[((z \to W_2' z) \circ \sigma \circ (z \mapsto W_1 z + b_1))(z)]_{q,q'} = \mathbf{1}[e_{(q,\mathsf{L})}^\top z = q'].$$

We also add Q more weights which let the inputs pass through along the $e_{(q,\mathsf{R})}$ directions (add Q more rows $e_{(q,\mathsf{R})}^\top$ to W_1, W_2', calling these indices $\bullet q$ for all $q \in Q$; set biases to 0), for a total of $4|Q|^2 + |Q|$ hidden units and $|Q|^2 + |Q|$ output dimensions of W_2'.

The second layer implements multiplication between the indicators and function values. The outputs are again indexed by $q, q' \in Q$. We define $W_2'' \in \mathbb{R}^{|Q|^2 \times (|Q|^2 + |Q|)}$ and $b_2'' \in \mathbb{R}^{|Q|^2}$ to be such that

$$[W_2'']_{(q,q'),(\bar{q},\bar{q}')} := |Q| \cdot \mathbf{1}[(q,q') = (\bar{q},\bar{q}')], \quad [W_2'']_{(q,q'),\bullet\bar{q}} := \mathbf{1}[q' = \bar{q}], \quad [b_2'']_{(q,q')} = -|Q|,$$
$$\forall q, q', \bar{q}, \bar{q}' \in Q.$$

Overall, so far we have

$$[(\sigma \circ (z \mapsto W_2'' W_2' z + b_2'') \circ \sigma \circ (z \mapsto W_1 z + b_1))(z)]_{q,q'} = g(q') \cdot \mathbf{1}[e_{(q,\mathsf{L})}^\top z = q'].$$

The third layer $W_3 \in \mathbb{R}^{d \times |Q|^2}$ simply converts these activations back into an transition map:

$$W_3 = \sum_{q' \in Q} e_{(q,\mathsf{R})} e_{q,q'}^\top.$$

Finally, we note the weight norms:

$$\|W_1\|_\infty \le 4|Q|, \quad \|b_1\|_\infty \le 4|Q| + 2, \quad \|W_2'' W_2'\|_\infty \le 4|Q|, \quad \|b_2''\|_\infty = |Q|, \quad \|W_3\|_\infty = 1.$$

$\square$

Recursive parallel scan. The rest of the construction uses a standard parallel algorithm for computing all prefix function compositions: *at layer $l \in [L]$, compose the function at position t with the function at position $t - 2^{l-1}$*. This is a standard algorithm for computing all prefix compositions of associative binary operations with a logarithmic-depth circuit [Hillis and Steele Jr., 1986]. We choose the position embeddings to enable implementing these "look-backs" with rotation matrices. For each $t \in \{-T+1, \dots, 0, 1, \dots, T\}$, we use the circle embeddings

$$P_{t,\mathsf{P}_1} := \cos\left(\frac{\pi t}{T}\right), \quad P_{t,\mathsf{P}_2} := \sin\left(\frac{\pi t}{T}\right).$$

In detail, for each $1 \le l \le L$:

- Let $\theta := -\frac{\pi 2^{l-1}}{T}$, $\gamma := 100T^2(\log|Q| + \log T)$.
- Let $H := 2, k := |Q|$. Recall that $|Q| \ge 2$. We will index the heads by superscripts $[\mathsf{L}], [\mathsf{R}]$.
- Select $W_Q^{[\mathsf{L}]} = W_Q^{[\mathsf{R}]} = W_K^{[\mathsf{R}]} := \sqrt{\gamma} \cdot (e_{\mathsf{P}_1} e_1^\top + e_{\mathsf{P}_2} e_2^\top)$.
- Select $W_K^{[\mathsf{L}]} := \sqrt{\gamma} \cdot (e_{\mathsf{P}_1} e_1^\top + e_{\mathsf{P}_2} e_2^\top) \rho_\theta$, where ρ_θ is the rotation matrix

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

  in the e_1, e_2 basis.
- Select $W_V^{[\mathsf{L}]} = W_V^{[\mathsf{R}]} := \sum_{q \in Q} e_{(q,\mathsf{L})} e_q^\top$.

- Select $W_C^{[\mathsf{L}]} = \sum_{q \in Q} e_q e_{(q,\mathsf{L})}^\top$, $W_C^{[\mathsf{R}]} = \sum_{q \in Q} e_q e_{(q,\mathsf{R})}^\top$.

At layer l, let $\alpha^{[\mathsf{L}]}, \alpha^{[\mathsf{R}]} \in \mathbb{R}^{2T}$ denote the attention mixture weights of the two heads. With this choice of γ, Lemma 13 and Proposition 2, for each $t \in [T]$, we are guaranteed that $\left\|\alpha^{[\mathsf{R}]} - e_t\right\|_1$ and $\left\|\alpha^{[\mathsf{L}]} - e_{t-2^{l-1}}\right\|_1$ are both at most $\frac{0.1}{|Q| \cdot T}$. Thus, by Hölder's inequality (noting that this mixture is over T vectors of ∞-norm at most $|Q|$), this attention layer's output is 0.1-close in the ∞-norm to the concatenated transition maps of the functions at positions t and $t - 2^{l-1}$, allowing us to invoke the perturbation-robust function approximation guarantee of Lemma 10 with $\Delta = 1$. For the MLP, we use the function composition gadget.

Thus, at the final layer, the (q, R) dimensions at position t contains the transition map of the prefix composition

$$\delta(\cdot, \sigma_t) \circ \ldots \circ \delta(\cdot, \sigma_1) \circ (q \mapsto q_0).$$

It suffices to choose W to be $z \mapsto e_{(q,\mathsf{R})}^\top z$ for an arbitrary q to read out the sequence of states as scalar outputs in $[|Q|]$. To output a one-hot encoding, an additional MLP (appended to the end of the final layer) would be required. $\square$

3.4.4.3 Proof of Theorem 11: Constant-depth shortcuts via Krohn-Rhodes decomposition

We begin with the full statement of the theorem:

Theorem 11 (Transformer Krohn-Rhodes)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a solvable semiautomaton (see Definition 8), $q_0 \in Q$, and $T \geq 1$. Then, there is a depth-$O(|Q|^2 \log |Q|)$ Transformer which continuously simulates $\mathcal{A}_{T,q_0}$, with embedding dimension $O(2^{|Q|} |\mathcal{T}(\mathcal{A})|)$, MLP width $|Q|^{O(2^{|Q|})} + O(2^{|Q|} |Q| \, |\mathcal{T}(\mathcal{A})| \, T)$, attention width $O(|Q| 2^{|Q|} |\mathcal{T}(\mathcal{A})|)$ heads, and weight norms are bounded by $6|Q| \, T \log T + 6 \max\{|Q|, |\Sigma|\}$.*

We will begin by presenting self-contained constructions for the two atoms in the Krohn-Rhodes decomposition: a modular counter and a memory unit. In Appendix 3.4.4.3.2, we will introduce necessary background from Krohn-Rhodes theory (including the definition of a solvable semiautomaton). In Appendix 3.4.4.3.3 and 3.4.4.3.4, we will complete the proof of Theorem 11.

3.4.4.3.1 Base cases: modular counting and memory

Base case 1: modular addition. We will start with a construction of a tiny network which lets us simulate any semiautomaton whose transformation semigroup is a cyclic group. Later on, we will use copies of this unit to handle all solvable groups. The construction simply uses attention to perform a flat prefix sum, and an MLP to compute the modular sum.

Definition 4 (Modular counter semiautomaton)**.** *For any positive integer n, define the mod-n* modular counter *semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$:*

$$Q := \{0, \ldots, n-1\},$$

$$\Sigma := \{0, \ldots, n-1\},$$

$$\delta(q, \sigma) := (q + \sigma) \text{ mod } n, \quad \forall q \in Q, \sigma \in \Sigma.$$

Lemma 15 (Simulating a modular counter)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be the mod-n modular counter semiautomaton. Let $q_0 \in Q$, and $T \geq 1$. Then, there is a depth-1 Transformer which continuously simulates $\mathcal{A}_{T,q_0}$, with embedding dimension 3, width $4nT$, and ∞-weight norms at most $4nT + 2$. It has $H = 1$ head with embedding dimension $k = 1$, and a 2-layer ReLU MLP.*

Proof. The intuition is simply that the lower triangular matrix causal mask can implement simulation in this cyclic group by performing unweighted prefix sums. The only subtlety is that selecting $W_Q = W_K = 0$ does not quite give us prefix sums: the attention mixture weights at position t are $\frac{1}{t}\sum_{t' \in [t]} e_{t'}$, while we would like the normalizing factor to be uniform across positions ($1/T$ rather than $1/t$). It is possible to undo this normalization using the MLP; however, a particulaly simple solution is to use an additional padding input $\perp$ and 1-dimensional position embeddings to "absorb" a fraction of the attention proportional to $1 - t/T$.

We proceed to formalize this construction, beginning with the input embedding and attention block:

- Select $d := 3, k := 1, H := 1$. Intuitively, the 3 dimensions implement {input/output, padding, position} "channels".
- Select input symbol embeddings $E(\sigma) := \sigma \cdot e_1 \in \mathbb{R}^d$ for each $\sigma \in \Sigma$.
- Include an extra position $\perp$, with embedding $E(\perp) := e_2$ and position encoding $P_{\perp,:} := 0$. Think of this as padding at position 0; it is not masked out by the causal attention mask at any position $t \geq 1$.
- For $t \in [T]$, select $P_{t,:} := \gamma_t e_3$, where $\gamma_t := \log(2T - t)$ is such that $\frac{1}{e^{\gamma_t} + t} = \frac{1}{2T}$.
- Select $W_Q := e_3, W_K := e_2, W_V := e_1, W_C^\top := e_1$.
- We do not need residual connections.

In the output of this attention module, for any input sequence $\sigma_{1:T}$, the 1st channel of the output at position t is then

$$s := \frac{1}{2T} \sum_{t \in [T]} \sigma_t.$$

where $z_t \in \{0, \ldots, n-1\}$ is such that $\delta(\cdot, \sigma_t) = g^{z_t}$. The MLP simply needs to memorize the function

$$s \cdot e_1 \mapsto (Ts \bmod n) \cdot e_1.$$

We invoke Lemma 10, with $\Delta = \frac{1}{2T}, B_x = \frac{n-1}{2} < \frac{n}{2}, B_y = n$. The number of possible values of S (the cardinality of $\mathcal{X}$ in Lemma 10) is at most nT. □

Base case 2: memory lookups. It turns out that to simulate *semigroups* instead of groups, the only additional ingredient is a *memory unit*, a semiautomaton for which there are "read" and "write" operations. The minimal example of this is a *flip-flop* (Example 2), a semiautomaton which can sequentially remember and retrieve a single bit $\in \{0, 1\}$, and whose transformation semigroup is the *flip-flop monoid*. It will be convenient to generalize this object to Q states:

Definition 5 (Memory semiautomaton)**.** *For a given state set Q, define the* memory *semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$:*

$$\Sigma = Q \cup \{\perp\},$$
$$\delta(q, \sigma) := \sigma \quad \forall q \in Q, \sigma \in \Sigma, \sigma \neq \perp,$$
$$\delta(q, \perp) := q \quad \forall q \in Q.$$

Lemma 16 (Simulating a memory semiautomaton)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be the memory semiautomaton. Let $q_0 \in Q$, and $T \geq 1$. Then, there is a depth-1 Transformer which continuously simulates $\mathcal{A}_{T,q_0}$, with embedding dimension 4, width $4|Q|$, and ∞-weight norms at most $2T\log(|Q|T)$. It has $H = 1$ head with embedding dimension $k = 2$, and a 2-layer ReLU MLP.*

Proof. We start in state $q_0 \in Q$. Our goal is to identify the closest non-$\perp$ token and output the corresponding state. The attention construction is:

- Select $d = 4, k = 2, H = 1$.
- Select input symbol encodings

  $$E(\sigma) := (\mathbb{1}[\sigma = \perp]q_0 + \mathbb{1}[\sigma \neq \perp]\sigma)e_1 + \mathbb{1}[\sigma = \perp]e_2 + e_4 \in \mathbb{R}^d,$$

  where the first coordinate denotes the action that sets the state[20], the second coordinate denotes whether the input is the no-op action $\perp$, and the fourth coordinate is padding.
- We use positional encoding $P_{t,:} := (t/T) \cdot e_3$.
- $W_Q := \begin{bmatrix} -2e_4 & e_4 \end{bmatrix} \in \mathbb{R}^{4\times 2}$, $W_K := \begin{bmatrix} ce_2 & ce_3 \end{bmatrix} \in \mathbb{R}^{4\times 2}$ for $c = O(T\log(|Q|T))$ as explained below, $W_V := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 2}$, and $W_C^\top := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 2}$.

The unnormalized attention score computed for position i attending to j is $c(j/T - \mathbb{1}[\sigma_j = \perp])$. Note that the max attention value is achieved at the closest reset action: the unnormalized scored is non-negative if and only if $\sigma_j \neq \perp$, and j/T increases with j ensuring that the closest position is chosen.

Denote this max position as $j_{\max}$. In the setting of hard attention, the output for the i_{th} token after the attention module is $E(\sigma_{j_{\max}})^\top e_1$. In particular, this value is q_0 if and only if $\sigma_j = \perp, \forall j \leq i$, i.e. the semiautomaton never leaves the starting state. Otherwise, the value is the value of the nearest non-$\perp$ state (including the current state).

By Lemma 13 (with $\gamma = c/T$), we can approximate hard-attention by soft-attention weights $\alpha \in \mathbb{R}^T$, that is, $\|\alpha - e_{j_{\max}}\|_1 \leq 2T \cdot e^{-c/T}$. This implies, that the output of the attention layer, $\left|\sum_{j\leq i} \alpha_j E(\sigma_j)^\top e_1 - E(\sigma_{j_{\max}})^\top e_1\right| \leq 2T|Q| \cdot e^{-c/T}$. Then, the MLP can simply round the first coordinate, and we can invoke Lemma 10 with $\Delta = 8T|Q|\exp(-c/T) = 1/2$ (for $c = T\log(16|Q|T)$), $B_x = |Q|$, $B_y = |Q|$ to get weight norm bound $(4 + \log(|Q|T))T \leq 2\log(|Q|T)T$ and width $4|Q|$. $\square$

[20]Technically $\sigma = \perp$ does not reset the state. We will see that when q_0 is selected, it must be that the semiautomaton is always in state q_0.

3.4.4.3.2 Prime decompositions of groups and semigroups

The key idea behind the proof of Theorem 2 is that all semigroups (and thus, all transformation semigroups of semiautomata) admit a "prime factorization" into elementary components, which turn out to be *simple groups* and copies of the *flip-flop monoid*, which have both been discussed in Section 3.3.1. This is somewhat counterintuitive: the only constraint on the algebraic structure of a semigroup is associativity (and indeed, there are many more semigroups than groups), but all of these structures can be built using these two types of "atoms". These components, as well as the *cascade product* under which this notion of "factorization" is defined, are naturally and efficiently implementable by constant-depth self-attention networks.

The special case of groups. We begin by discussing the analogous decomposition for *groups*, which generalizes the fact that integers have unique prime factorizations. Let G be a finite group, and let

$$G = H_n \triangleright H_{n-1} \triangleright \cdots \triangleright H_1 \triangleright H_0 = 1$$

be a *composition series*: each H_i is a maximal proper normal subgroup of H_{i+1}; 1 denotes the trivial group with 1 element. Then the quotient group H_{i+1}/H_i is called a *composition factor*. The Jordan-Hölder theorem tells us that one can think about the set of composition factors as an invariant of G.

Theorem 14 (Jordan-Hölder)**.** *Any two composition series of G are equivalent: they have the same length n, and the sequences of compositions factors H_{i+1}/H_i are equivalent under permutation and isomorphism.*

When each H_{i+1}/H_i is abelian, G is called a *solvable* group. It turns out that each H_{i+1}/H_i is a simple group, so the composition factors of solvable groups can only be cyclic groups of prime order (because every finitely generated abelian group is a direct product of cyclic groups, and, of these, only those of prime order are simple). The smallest non-solvable group is A_5, realizable as the group of even permutations of 5 elements. As a part of Theorem 11, we will use the composition series to iteratively build neural networks which simulates solvable group operations, requiring intricate constructions to do this with depth independent of the sequence length T.

Adding memory to handle semigroups. Now, we move on to semigroups. When not all of the input symbols to a semiautomaton induce permutations, we no longer have the group axiom of invertibility (also, if there is no explicit identity symbol, we are not guaranteed to have the monoid axiom of an identity element either). Intuitively, this would seem to induce a much larger family of algebraic structures; an analogy, which is formalizable by representation theory, is that we are now considering a collection of general matrices under multiplication, instead of only invertible ones. The non-invertible transitions *collapse the rank* of the transformations, reducing the set of reachable transformations whenever they are included in an input sequence.

A landmark result of Krohn and Rhodes [1965] tames the seemingly vast and unorderly universe of general finite semigroups. It extends the Jordan-Hölder theorem to the case of semigroups, for a more sophisticated notion of decomposition. Since that work, many variations have arisen, in terms of its precise statement, construction of the decomposition, and proof of correctness. Out of these, an important development is the *holonomy decomposition* method [Zeiger, 1967, Eilenberg, 1974], which forms the basis of our results. We extract the definitions and theorems from Maler and Pnueli [1994], whose exposition emphasizes explicitly tracking the construction of the semiautomaton. We also refer to Maler [2010], Egri-Nagy and Nehaniv [2015], Zimmermann [2020] as recent expositions, containing historical context.

Definition 6 (Cascade product; cf. [Maler, 2010], Definition 11)**.** *Let n be a positive integer. For each $i \in [n]$, let $\mathcal{A}^{(i)} = (Q^{(i)}, \Sigma^{(i)}, \delta^{(i)})$ be a semiautomaton. For $i \in \{2, \dots, n\}$, let $\phi^{(i)} : Q^{(1)} \times \cdots \times Q^{(i-1)} \times \Sigma \to \Sigma^{(i)}$ denote a* dependency function*. This object $(\{\mathcal{A}^{(i)}\}; \{\phi^{(i)}\})$ is called a* transformation cascade*, and defines a* cascade product semiautomaton *$\mathcal{A} = (Q^{(1)} \times \cdots \times Q^{(n)}, \Sigma^{(1)}, \delta)$ by "feedforward simulation" under the dependency function. We define δ by the i-th component of its output (which we call $\delta^{(\leq i)} : Q^{(1)} \times \cdots \times Q^{(n)} \times \Sigma^{(1)} : Q^{(i)}$):*

$$\delta^{(\leq i)}((q^{(1)}, \dots, q^{(n)}), \sigma) := \delta^{(i)}(q^{(i)}, \sigma^{(i)}),$$

where

$$\sigma^{(i)} := \begin{cases} \sigma & \text{if } i = 1 \\ \phi^{(i)}(q^{(1)}, \dots, q^{(i-1)}, \sigma) & \text{otherwise} \end{cases}.$$

The corresponding transformation semigroup $\mathcal{T}(\mathcal{A})$ is known as a cascade product semigroup.

Intuitively, the cascade specifies a way to compose semiautomata hierarchically: the first layer $i = 1$ maps input sequences to its state sequence, and each internal layer receives an input which depends on the states of all of the preceding layers. Algebraically, the cascade product semigroup is a subsemigroup of the larger *wreath product* of semigroups (the straightforward analogue of the wreath product of groups, discussed in Section 3.3.1). Although this is useful from an algebraic point of view, we will not use this perspective; the cascade product is a smaller substructure of the wreath product which is sufficient for semiautomaton simulation.

Finally, it will be convenient to define *permutation-reset semiautomata*, which are a useful intermediate step in the Krohn-Rhodes decomposition. To obtain our final result, we will further break these semiautomata down into flip-flops and simple groups.

Definition 7 (Permutation-reset semiautomaton; cf. [Maler and Pnueli, 1994], Definition 12)**.** *A semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ is a* permutation-reset *semiautomaton if, for each $\sigma \in \Sigma$, the transition function $\delta(\cdot, \sigma) : Q \to Q$ is either a bijection (i.e. a permutation over the states of Q) or constant (i.e. maps every state to some $q(\sigma)$). Associated with each permutation-reset semiautomaton is its* permutation group*, generated by only the bijections.*

Now we can state the Krohn-Rhodes theorem, which decomposes *every* finite semiautomaton into a transformation cascade.

Theorem 15 (Krohn-Rhodes)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a semiautomaton. Then, there exists a transformation cascade $\{\mathcal{A}^{(1)}, \dots, \mathcal{A}^{(n)}; \phi^{(2)}, \dots, \phi^{(n)}\}$, defining a cascade product semiautomaton $\mathcal{A}'$, such that:*

(i) The input symbol space of $\mathcal{A}^{(1)}$ (and thus, that of $\mathcal{A}'$) is Σ, the same as that of $\mathcal{A}$.

(ii) Letting $Q^{(i)}$ denote the state space of $\mathcal{A}^{(i)}$, there exists a function $\mathcal{W} : Q^{(1)} \times \cdots \times Q^{(n)} \to Q$ such that $\mathcal{W} \circ \mathcal{A}'_{T,q_0}$ simulates $\mathcal{A}_{T,q_0}$ for all $T \geq 1, q_0 \in Q$. For each $i \in [n]$, the transformation semigroup $\mathcal{T}(\mathcal{A}^{(i)})$ is a permutation-reset semiautomaton with at most $|Q|$ states, whose permutation group is a (possibly trivial) subgroup of $\mathcal{T}(\mathcal{A})$ (Maler and Pnueli [1994], Theorem 4).

(iii) The number of semiautomata in the cascade is $n \leq 2^{|Q|}$. Furthermore, the cascade has at most $|Q|$ levels: the indices can be partitioned into at most $L \leq |Q|$ contiguous subsets $N^{(1)} = \{1, \ldots, n_1\}, N^{(2)}, \{n_1 + 1, \ldots, n_1 + n_2\}, \ldots, N^{(L)} = \{n - n_L + 1, \ldots, n\}$ such that $\phi^{(i)}$ only depends on input indices from previous partitions (Maler and Pnueli [1994], Claim 11 & Corollary 12).

With this decomposition, we are now able to define a *solvable* semiautomaton.

Definition 8 (Solvable semiautomaton)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a semiautomaton. We call $\mathcal{A}$* solvable *if all the permutation groups associated with all of the permutation-reset automata from Theorem 15 are solvable groups.*

The remainder of this section will build our construction from the bottom up:

- Appendix 3.4.4.3.3 will build up from the base case of cyclic groups (Lemma 15), using increasingly sophisticated notions of group products, culminating in a recursive construction which simulates all stages of the Jordan-Hölder composition series. The crucial step is a construction for simulating the semidirect product of groups, given networks which simulate the individual components; this allows us to handle the solvable non-abelian groups.
- Appendix 3.4.4.3.4 will build networks which simulate permutation-reset semiautomata. A new base case arises: the *memory unit* (Lemma 16), a semiautomaton whose transformation semigroup is a generalization of the flip-flop monoid. Combining the constructions for solvable groups and memory units, we obtain simulators for solvable permutation-reset semiautomata. Finally, the cascade product guaranteed by Krohn-Rhodes (Theorem 15) glues all of these pieces together, giving us the final result.

3.4.4.3.3 Simulating solvable groups

We begin by handling groups. Now, we are ready to specify the recursive constructions which "glue" these components together to form solvable groups. We will proceed in a "bottom-up" order:

(i) Define a canonical semiautomaton $\mathcal{A}^G$ corresponding to each group G (Definition 9), such that if a network can simulate $\mathcal{A}^G$, it can simulate any other semiautomaton whose transformation semigroup $\mathcal{T}(\mathcal{A}^G)$ is G. This lets us talk about simulating *groups*, rather than particular semiautomata. We will show how to turn simulators for groups N and H into simulators for extensions of N by H, for increasingly sophisticated extensions, until all cases have been captured.

(ii) Show how to build the *trivial extension*: given networks which simulate the groups N and H, simulate the direct product $G \cong N \times H$, by simply running the individual simulators in parallel (Lemma 17). Combined with Lemma 15, this immediately allows us to simulate arbitrary abelian groups with depth 1, since every abelian group is isomorphic to a direct product of cyclic groups.

(iii) Show how to build a *split extension*: given networks which simulate a normal subgroup N and quotient H, construct a network which simulates any semidirect product $G \cong N \rtimes H$ (Lemma 18). This is the first place where we will require a sequential cascade of layers. It will allow us to handle certain families of non-abelian groups (including S_3, D_{2n}, A_4, S_4)

(iv) Show how to build *arbitrary* extensions (any G which contains N as a normal subgroup, and for which the quotient group G/N is isomorphic to H), using the wreath product (Lemma 19), which contains all of the group extensions. The wreath product is itself the semidirect product between a $|H|$-way direct product and H, so this can be done in a constant number of layers, by the above. This finally lets us implement any step of a composition series. In particular, using cyclic groups as a simulable base case, this shows that we can simulate all solvable groups.

Step (i). It will be convenient to associate with each group G a canonical "complete" semiautomaton for the class of all semiautomata $\mathcal{A}$ for which $\mathcal{T}(\mathcal{A}) = G$. It is simply the one whose input symbol space Σ is every transformation reachable by some sequence of inputs (i.e. every element of $\mathcal{T}(\mathcal{A})$). (For a semigroup, we would also want to adjoin the identity element if it is missing, however, we will only find it useful to define this for groups.)

Definition 9 (Canonical group semiautomaton; simulating a group)**.** *Let G be a finite group. Then, we define the* canonical group semiautomaton *for G as the semiautomaton (Q, Σ, δ) defined by:*

- *$Q := G$, the set of elements of G. Note that if (for example) $G = S_n$, we are setting the state space to be the set of $n!$ permutations, not the ground set $[n]$.*
- *$\Sigma := G$. That is, we include* all *functions in the input symbol space.*
- *$\delta(g, h) := h \cdot g$, for all $\forall g \in Q, h \in \Sigma$. (In algebraic terms, we are embedding the G into its* left regular representation, *a.k.a.* left multiplication action.*) Thus, if we take q_0 to be the identity element, the sequence of states $q_1, q_2, \ldots, q_T$ corresponds to $q_t = \sigma_t\sigma_{t-1}\ldots\sigma_1$.*
- *When we simulate the canonical group semiautomaton, we will always choose q_0 to be the identity element e_G.*

A sequence-to-sequence network is said to continuously simulate *G at length T if it continuously simulates the canonical group semiautomaton of G at length T.*

Notation for composable implementations. Let us furthermore formalize an *implementation* of group simulation. For any finite group G, $T \geq 1$, $q_0 \in G$, let $\mathcal{A}^G = (Q, \Sigma, \delta)$ be the canonical semiautomaton for G. Then, we summarize a family of concrete implementations of networks which continuously simulate of $\mathcal{A}_{T,e_G}$. We write $\mathsf{sim} : (G, T) \mapsto (E : G \to \mathbb{R}^d, f_{\mathrm{tf}} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}, W : \mathbb{R}^d \to G)$, where the shape parameters of the output can depend on G, T.

To reduce notational clutter, we will access the shape attributes of an implementation via "object-oriented" notation, defining

$$\mathsf{sim}(G, T).\{\mathsf{depth}, \mathsf{dim}, \mathsf{heads}, \mathsf{headDim}, \mathsf{mlpWidth}, \mathsf{normBound}\}$$

to respectively denote the complexity-parameterizing quantities

$$\{L, d, H, k, \max_j \{d'_j\}, B\},$$

defined in Section 3.1. Also, we will let $\mathsf{sim}(G, T).\{E, \theta, W\}$ respectively denote the encoding layer E, network parameters θ_{nn}, and decoding layer W.

Canonical encodings of group elements. We also enforce that throughout our constructions of networks which simulate groups, we will maintain that all networks and their submodules manipulate encodings via *integer vectors* in a consecutive range $\{0, \ldots, n-1\}$. Furthermore, the identity element will always map to the zero vector. We will keep track of the dimensionality of these vectors $\mathsf{sim}(G, T).\mathsf{repDim} \le d$, and their maximum entries $\mathsf{sim}(G, T).\mathsf{repSize} - 1$. All encoders E and decoders W will map all group elements to and from this kind of representation, and we will choose $W = E^{-1}$. In all, the networks will keep a repDim-dimensional "workspace" of integer vectors, with entries bounded by $\mathsf{repSize} - 1$. When combining groups via the various products constructions, we will combine the components' individual workspaces to create a larger workspace for the product group's elements.

We make some additional remarks on implementations:

- Note that the canonical semiautomaton "forgets" about the semiautomaton abstraction, and never assumes that G is a permutation group on the original state space Q of the semiautomaton we would like to simulate. Indeed, when $N \triangleleft G$ are permutation groups on Q, there is no natural permutation group on Q associated with the quotient $H \cong G/N$; it turns out that will consider simulators for N and H.[21]
- To return to solving the simulation problem for some semiautomaton $\mathcal{A} = (Q, \Sigma, \delta)$ whose transformation semigroup is isomorphic to G (at length T and initial state q_0), let $\mu : G \to S_Q$ denote this isomorphism. We use $\mathcal{A}^G_{T,e_G}(\sigma_{1:T})$ as the network, with an encoding layer $E \circ \mu^{-1}$, and decoding layer $(\pi \mapsto \pi(q_0)) \circ \mu \circ W$, which can be memorized by an MLP of width $O(|G|)$ via Lemma 11.
- The modular counter semiautomaton, for which we constructed a simulator in Lemma 15, is the canonical group semiautomaton for the corresponding cyclic group C_n. Calling this construction sim_{C_n}, we can easily verify that it satisfies the canonical simulator's conditions, and:
    - $\mathsf{sim}_{C_n}.\mathsf{depth} = 1$.
    - $\mathsf{sim}_{C_n}.\mathsf{dim} = 3$.
    - $\mathsf{sim}_{C_n}.\mathsf{heads} = 1$.
    - $\mathsf{sim}_{C_n}.\mathsf{headDim} = 1$.
    - $\mathsf{sim}_{C_n}.\mathsf{mlpWidth} = 4|G| \cdot T$.
    - $\mathsf{sim}_{C_n}.\mathsf{normBound} \le 4|G| \cdot T + 2 \le 6|G| \cdot T$.
    - $\mathsf{sim}_{C_n}.\mathsf{repDim} = 1$.
    - $\mathsf{sim}_{C_n}.\mathsf{repSize} = |G|$.

Step (ii). As a precursor to the more sophisticated products, we formalize the obvious fact that two non-interacting parallel semiautomata can be simulated without increasing the depth. First, we define the direct product semiautomaton:

[21] There is nothing in general preventing quotient groups from being extremely large groups which are not realizable as smaller permutation groups. For concrete examples, see [Kovács and Praeger, 1989]. When we ultimately specialize to simulating the composition series of solvable groups, the largest groups we will handle will be the cyclic groups of prime order, so we will in the end be guaranteed that the groups we want to simulate are realizable with $\le |Q|$ states, but not directly or canonically.

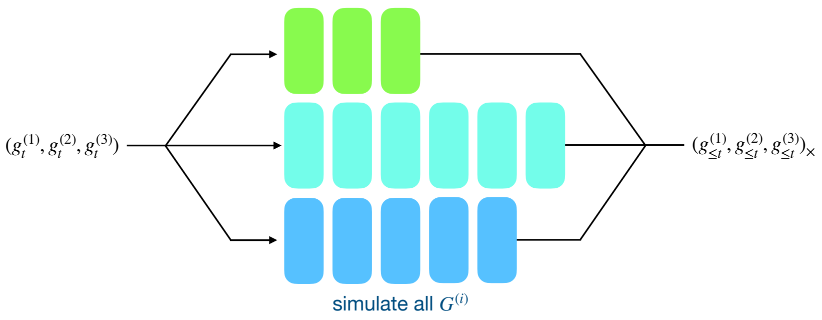

$(g_t^{(1)}, g_t^{(2)}, g_t^{(3)})$
$(g_{\le t}^{(1)}, g_{\le t}^{(2)}, g_{\le t}^{(3)})_\times$
simulate all $G^{(i)}$

Figure 3.10: Recursive construction for simulating a direct product of groups $G^{(1)} \times \cdots \times G^{(n)}$. Any number of groups can be simulated in parallel without increasing the depth.

Definition 10 (Direct product of semiautomata). *Let $\mathcal{A} = (Q, \Sigma, \delta), \mathcal{A}' = (Q', \Sigma', \delta')$ be two semiautomata. Then, $\mathcal{A} \times \mathcal{A}' = (Q \times Q', \Sigma \cup \{e\} \times \Sigma' \cup \{e\}, \delta \times \delta')$ denotes the natural direct product semiautomaton. Its states are ordered pairs $(q \in Q, q' \in Q')$. Its input symbols are defined similarly, adjoining identity inputs (so that $\delta(q, e) = q, \delta'(q', e) = q'$). The transitions $\delta \times \delta'$ are defined such that*

$$(\delta \times \delta')((q, q'), (\sigma, \sigma')) := (\delta(q, \sigma), \delta'(q', \sigma')).$$

Note that under this definition, we have $\mathcal{T}(\mathcal{A} \times \mathcal{A}') = \mathcal{T}(\mathcal{A}) \times \mathcal{T}(\mathcal{A}')$. In particular, for two groups G, H, we have $G \times H = \mathcal{T}(\mathcal{A}^G) \times \mathcal{T}(\mathcal{A}^H) = \mathcal{T}(\mathcal{A}^{G \times H}) = G \times H$.

Lemma 17 (Direct product via parallel simulation). *Let $G^{(1)}, \ldots, G^{(n)}$ be a collection of finite groups, and let $T \geq 1$. Suppose each group admits a simulation $\mathsf{sim}_i := \mathsf{sim}(G^{(i)}, T)$. Then, there is a simulation of the direct product group $\mathsf{sim}_\times := \mathsf{sim}(G^{(1)} \times \ldots \times G^{(n)}, T)$, whose sizes satisfy:*

- $\mathsf{sim}_\times.\mathsf{depth} = \max_i\{\mathsf{sim}_i.\mathsf{depth}\}$.
- $\mathsf{sim}_\times.\mathsf{dim} = \sum_i\{\mathsf{sim}_i.\mathsf{dim}\}$.
- $\mathsf{sim}_\times.\mathsf{heads} = \sum_i\{\mathsf{sim}_i.\mathsf{heads}\}$.
- $\mathsf{sim}_\times.\mathsf{headDim} = \max_i\{\mathsf{sim}_i.\mathsf{headDim}\}$.
- $\mathsf{sim}_\times.\mathsf{mlpWidth} = \sum_i\{\mathsf{sim}_i.\mathsf{mlpWidth}\}$.
- $\mathsf{sim}_\times.\mathsf{normBound} \leq \max_i\{\mathsf{sim}_i.\mathsf{normBound}\}$.
- $\mathsf{sim}_\times.\mathsf{repDim} = \sum_i\{\mathsf{sim}_i.\mathsf{repDim}\}$.
- $\mathsf{sim}_\times.\mathsf{repSize} = \max_i\{\mathsf{sim}_i.\mathsf{repSize}\}$.

Proof. First, we pad all of the individual sim_i with layers implementing identity (add residual connections, and set attention W_V and all MLP weight matrices to 0), so that all of them have depth $\max_i\{\mathsf{sim}_i.\mathsf{depth}\}$.

Then, the intuition is to construct the direct product semiautomaton by concatenating the "workspaces" of each $G^{(i)}$. In other words, we set the canonical encoding $\mathsf{sim}_\times.E$ of $(g^{(1)}, \ldots, g^{(n)})$ to be the concatenation of each sim_i's encodings.

The direct product simply lets each sim_i take inputs and outputs in its individual workspace. To enable this, we need enough parallel dimensions. We set an embedding space of dimension $\sum_i\{\mathsf{sim}_i.\mathsf{dim}\}$

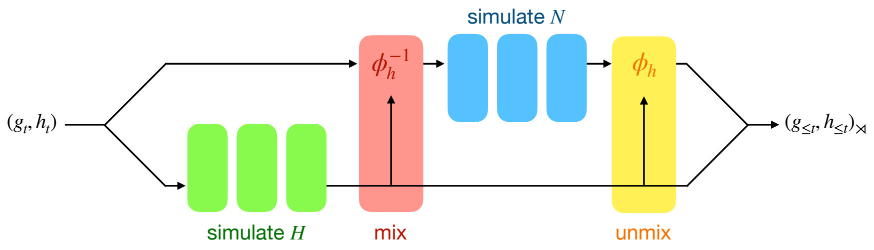


Figure 3.11: Recursive construction for simulating the semidirect product $N \rtimes H$. The quotient group H is simulated first; these outputs are used to "re-map" the inputs into the simulator for N.

(and similarly within the heads and MLPs), partitioning the coordinates such that in the product construction, each $\mathsf{sim}_i.E$ and $\mathsf{sim}_i.\theta$ only reads and writes to its own dimensions.

This clearly simulates the direct product group. Figure 3.10 provides a sketch of this construction. □

Note that the direct product construction already allows us to simulate all finite abelian groups in constant depth, since each such group is isomorphic to the direct product of a collection of abelian groups of prime power order.

Step (iii). Now, as a harder (and conceptually crucial) case, we show how to simulate a group which is a semidirect product of two groups we already know how to simulate. This encompasses the direct product as a special case, but can now handle some non-abelian groups which admit such decompositions (like the dihedral group D_{2n}). The catch is that we will have to simulate these groups using a *sequential* cascade of the individual simulators. This is the key lemma which lets us simulate non-abelian groups:

Lemma 18 (Semidirect product via 4-stage cascade)**.** *Let G be a finite group which is isomorphic to a semidirect product: $G \cong N \rtimes H$, where N is a normal subgroup of G. Let $T \ge 1$. Suppose N, H admit simulations $\mathsf{sim}_N := \mathsf{sim}(N,T), \mathsf{sim}_H := \mathsf{sim}(H,T)$. Then, there is a simulation of G, $\mathsf{sim}_\rtimes := \mathsf{sim}(G,T)$, whose sizes satisfy:*

- $\mathsf{sim}_\rtimes.\mathsf{depth} = \mathsf{sim}_N.\mathsf{depth} + \mathsf{sim}_H.\mathsf{depth} + 2$.
- $\mathsf{sim}_\rtimes.\mathsf{dim} = \mathsf{sim}_N.\mathsf{dim} + \mathsf{sim}_H.\mathsf{dim}$.
- $\mathsf{sim}_\rtimes.\mathsf{heads} = \max\{\mathsf{sim}_N.\mathsf{heads}, \mathsf{sim}_H.\mathsf{heads}\}$.
- $\mathsf{sim}_\rtimes.\mathsf{headDim} = \max\{\mathsf{sim}_N.\mathsf{headDim}, \mathsf{sim}_H.\mathsf{headDim}\}$.
- $\mathsf{sim}_\rtimes.\mathsf{mlpWidth} = \max\{\mathsf{sim}_{\{N,H\}}.\mathsf{mlpWidth}, 4|G|\}$.
- $\mathsf{sim}_\rtimes.\mathsf{normBound} \le \max\{\mathsf{sim}_{\{N,H\}}.\mathsf{normBound}, 6\,\mathsf{sim}_{\{N,H\}}.\mathsf{repSize}, \mathsf{sim}_N.\mathsf{repDim} + \mathsf{sim}_H.\mathsf{repDim}\}$.
- $\mathsf{sim}_\rtimes.\mathsf{repDim} = \mathsf{sim}_N.\mathsf{repDim} + \mathsf{sim}_H.\mathsf{repDim}$.
- $\mathsf{sim}_\rtimes.\mathsf{repSize} = \max\{\mathsf{sim}_N.\mathsf{repSize}, \mathsf{sim}_H.\mathsf{repSize}\}$.

Proof. The intuition is as follows, using the dihedral group $D_{2n} \cong C_n \rtimes C_2$ as an example:

- For simplicity, let us think of the "reversible car on a circular world" semiautomaton, whose transformation semigroup is D_{2n}. Its state consists of a direction $\in \{+1, -1\}$, and a position $\in \{0, 1, \ldots, n-1\}$. It has two types of inputs: "advance by i" (increment the position by i in the current direction, modulo n), and "reverse" (flip the sign of the direction). Our simulation task is to track the car's state sequence, given a sequence of inputs (in constant depth, of course).
- It is intuitively clear that we can (and should) compute the sequence corresponding to "direction at time t", which is equivalent to simulating the parity semiautomaton.
- We will convert the "advance" moves via a "basis transformation": whenever the current direction is -1, an "advance by i" should be converted into $-i$. Then, we have reduced the problem to the prefix sum.

Algorithm. This intuition essentially shows us how to implement arbitrary semidirect products; we derive the basis change from ϕ. Before implementing it with Transformer operations, we formalize this "basis transformation". Recall that by the definition of a semidirect product, the elements of $N \rtimes H$ can be written as pairs $(g \in N, h \in H)$, equipped with a homomorphism $\phi : h \to \mathrm{Aut}(N)$ which specifies a multiplication rule:

$$(g, h) \cdot (g', h') := (g\phi_h(g'), hh').$$

Let us write down the properties of ϕ:

- ϕ is a homomorphism. That is, $\phi_{h \cdot h'} = \phi_h(\phi_{h'}(\cdot)) = \phi_h \circ \phi_{h'}$ as permutations on N.
- The output of that homomorphism, ϕ_h, is *also* a homomorphism. That is, $\phi_h(gg') = \phi_h(g) \cdot \phi_h(g')$.

Let us roll out the definition of the semidirect product, given a sequence of inputs (g_t, h_t):

$$(g_2, h_2) \cdot (g_1, h_1) = (g_2 \cdot \phi_{h_2}(g_1), h_2 h_1),$$

$$(g_3, h_3) \cdot (g_2, h_2) \cdot (g_1, h_1) = (g_3 \cdot \phi_{h_3}(g_2 \cdot \phi_{h_2}(g_1)), h_3 h_2 h_1),$$

$$(g_4, h_4) \cdots (g_1, h_1) = (g_4 \cdot \phi_{h_4}(g_3 \cdot \phi_{h_3}(g_2 \cdot \phi_{h_2}(g_1))), h_4 h_3 h_2 h_1).$$

In general, by induction, letting $(g_{\le t}, h_{\le t})$ denote $(g_t, h_t) \cdots (g_1, h_1)$, we have

$$g_{\le t} = g_t \cdot \phi_{h_t}(g_{t-1}) \cdot \phi_{h_t h_{t-1}}(g_{t-2}) \cdots \phi_{h_t \ldots h_3}(g_2) \cdot \phi_{h_t \ldots h_2}(g_1).$$

Applying $\phi^{-1}_{h_{\le t}}$ on both sides, we notice that

$$\phi^{-1}_{h_{\le t}}(g_{\le t}) = \phi^{-1}_{h_{\le t}}(g_t) \cdot \phi^{-1}_{h_{\le t-1}}(g_{t-1}) \cdots \phi^{-1}_{h_{\le 2}}(g_2) \cdot \phi^{-1}_{h_{\le 1}}(g_1).$$

Thus, it suffices to compute each $h_{\le t} = h_t h_{t-1} \ldots h_1$, map each $g_t \mapsto \phi^{-1}_{h_{\le t}}(g_t)$, compute the prefix products in these "coordinates", then invert the mapping to get back $g_{\le t}$.

Implementation. Like before, we partition the embedding dimension in our construction $\mathsf{sim}_{\rtimes}$ into blocks, one for each component simulator. Let us index the dimensions by the $d_N := \mathsf{sim}_N.\mathsf{dim}$ indices in the "N channel" and analogously for the d_H-dimensional "H channel". We choose the canonical encoding E to map elements to their individual channels:

$$E(g, h) = \mathsf{sim}_N.E(g) \ \ \textit{(in the N channel)} + \mathsf{sim}_H.E(h) \ \ \textit{(in the H channel)}.$$

We proceed to specify the construction layer-by-layer. Let $L_{\{N,H\}}$ denote $\mathsf{sim}_{\{N,H\}}.\mathsf{depth}$.

Layers 1 **through** L_H**: quotient group simulation.** As suggested by the intuitive sketch, we begin with L_H Transformer layers, which are just a copy of $\mathsf{sim}_H.\theta$, reading and writing in the H channel, with a parallel residual layer in the N channel. So far, after these L_H layers, the output at each position t is an integer vector, whose H channel contains $h_{\leq t}$, and whose N channel contains $\mathsf{sim}_N.E(g_t)$.

Layer $L_H + 1$**: basis change.** Now, let us add one more "mixing" Transformer layer, whose attention block is identity[22]; we only need a 3-layer MLP block, which represents the function

$$(g \in N, h \in H) \mapsto \phi_h^{-1}(g).$$

To do this, we invoke Lemma 11 (choosing the output to be in the same representation as that used by $\mathsf{sim}_N.E$, in the N channel), with

$$\Delta = 1, d_{\text{in}} = \mathsf{sim}_N.\mathsf{repDim} + \mathsf{sim}_H.\mathsf{repDim},$$

$$B_x = \max\{\mathsf{sim}_N.\mathsf{repSize}, \mathsf{sim}_H.\mathsf{repSize}\}, B_y = \mathsf{sim}_N.\mathsf{repSize},$$

giving us a construction with

$$d_1 \leq 4(|N| + |H|), \quad d_2 \leq |N| \cdot |H|,$$

$$\|W_1\|_\infty \leq 4, \quad \|b_1\|_\infty \leq 6 \max\{\mathsf{sim}_N.\mathsf{repSize}, \mathsf{sim}_H.\mathsf{repSize}\},$$

$$\|W_2\|_\infty \leq 1, \quad \|b_2\|_\infty \leq \mathsf{sim}_N.\mathsf{repDim} + \mathsf{sim}_H.\mathsf{repDim}, \quad \|W_3\|_\infty \leq \mathsf{sim}_N.\mathsf{repSize}.$$

We also add residual connections in the H channel. In summary, after this layer, the output at each position t is an integer vector, whose H channel contains $h_{\leq t}$, and whose N channel contains $\phi_{h_{\leq t}}^{-1}(g_t)$.

Layers $L_H + 1$ **through** $L_H + L_N + 1$**: normal group simulation.** The next L_N layers are a copy of $\mathsf{sim}_H.\theta$, with residual connections in the H channel. After these layers, the output at each position t is an integer vector, whose H channel contains $h_{\leq t}$, and whose N channel contains $\phi_{h_{\leq t}}^{-1}(g_{\leq t})$.

[22]Even when an attention block simply implements identity, we choose to include it, rather than combining the preceding and subsequent MLPs into a single MLP. This is to ensure that if we compose a number of Transformer layers that depends on $|Q|$, the depth of each MLP is bounded by an absolute constant.

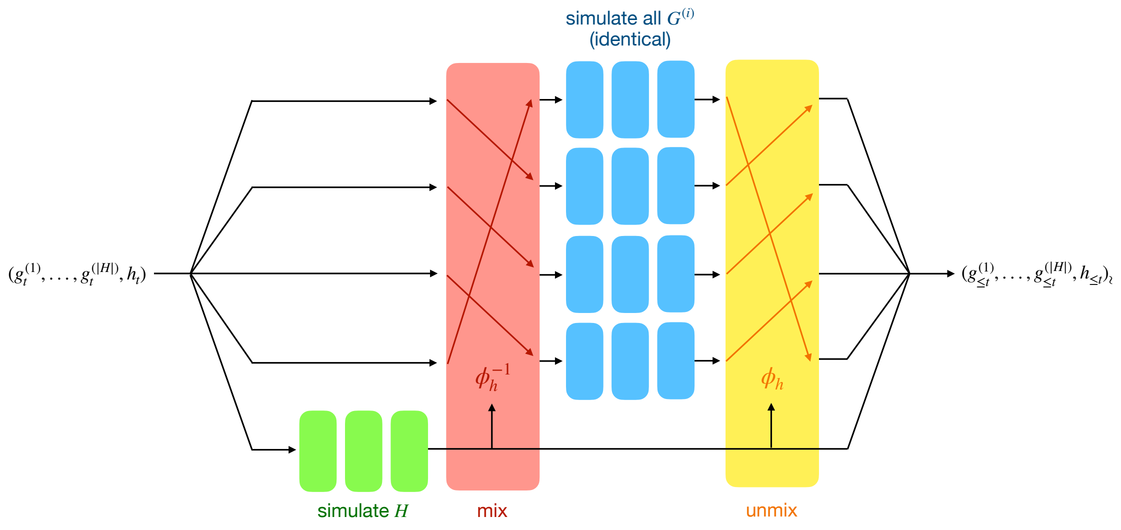

simulate all $G^{(i)}$ (identical)
$(g_t^{(1)}, \dots, g_t^{(|H|)}, h_t)$
ϕ_h^{-1}
ϕ_h
$(g_{\le t}^{(1)}, \dots, g_{\le t}^{(|H|)}, h_{\le t})_\wr$
simulate H
mix
unmix

Figure 3.12: Recursive construction for simulating the wreath product $N \wr H$. One independent copy of N is instantiated for each element of H, while the simulator for H permutes them.

Layer $L_H + L_N + 2$: undoing the basis change. Now, we add one more Transformer layer, whose attention block is identity; we will again use a 3-layer MLP block to represent the inverse of our mapping function

$$(g \in N, h \in H) \mapsto \phi_h(g).$$

This uses Lemma 11, with exactly the same bounds.

At the end of this final "unmixing" layer, the output at each position t is an integer vector, whose H channel contains $h_{\le t}$, and whose N channel contains $g_{\le t}$; thus, this is a valid simulation of the semidirect product.

This construction is sketched in Figure 3.11. □

Step (iv). Note that $H \cong G/N$ does *not* imply that G is a semidirect product of N and H. Thus, although simulating semidirect products allows us to handle some families of non-abelian groups, this does not yet allow us to handle general solvable groups (i.e. general steps of a composition series, even with a cyclic quotient group). The smallest example is the non-abelian *quaternion group* Q_8, the group of unit quaternions under multiplication, which cannot be realized as a semidirect product of subgroups. Instead, we need to appeal to the Krasner–Kaloujnine *universal embedding theorem* [Krasner and Kaloujnine, 1951]: a characterization of all of the groups G which are extensions of N by H, as subgroups of the wreath product $N \wr H$.

Lemma 19 (Wreath product via direct and semidirect products)**.** *Let G be a finite group which is isomorphic to a wreath product: $G \cong N \wr H$. Let $T \ge 1$. Suppose N, H admit simulations $\mathsf{sim}_N := \mathsf{sim}(N,T), \mathsf{sim}_H := \mathsf{sim}(H,T)$. Then, there is a simulation of G, $\mathsf{sim}_\wr := \mathsf{sim}(G,T)$. In the case where $\mathsf{sim}_\wr.\mathsf{repDim} = 1$, the sizes satisfy:*

- $\mathsf{sim}_\wr.\mathsf{depth} = \mathsf{sim}_N.\mathsf{depth} + \mathsf{sim}_H.\mathsf{depth} + 2.$
- $\mathsf{sim}_\wr.\mathsf{dim} = |H| \cdot \mathsf{sim}_N.\mathsf{dim} + \mathsf{sim}_H.\mathsf{dim}.$

- $\mathsf{sim}_\wr.\mathsf{heads} = \max\{|H| \cdot \mathsf{sim}_N.\mathsf{heads}, \mathsf{sim}_H.\mathsf{heads}\}$.
- $\mathsf{sim}_\wr.\mathsf{headDim} = \max\{\mathsf{sim}_N.\mathsf{headDim}, \mathsf{sim}_H.\mathsf{headDim}\}$.
- $\mathsf{sim}_\wr.\mathsf{mlpWidth} = \max\{|H| \cdot \mathsf{sim}_N.\mathsf{mlpWidth}, \mathsf{sim}_H.\mathsf{mlpWidth}, 5|H|^2|N|\}$.
- $\mathsf{sim}_\wr.\mathsf{normBound} \le \max\{\mathsf{sim}_{\{N,H\}}.\mathsf{normBound}, 6\,|H|\}$.
- $\mathsf{sim}_\wr.\mathsf{repDim} = |H| \cdot \mathsf{sim}_N.\mathsf{repDim} + 1$.
- $\mathsf{sim}_\wr.\mathsf{repSize} = \max\{\mathsf{sim}_N.\mathsf{repSize}, \mathsf{sim}_H.\mathsf{repSize}\}$.

Proof. Even though the wreath product's algebraic structure can be very complex, the construction just requires us to implement its relatively simple description. Applying Lemma 17, we have a network $\mathsf{sim}_\times$ which simulates $N \times \ldots \times N$. Then, we simply apply Lemma 18, using sim_H to "re-map" inputs to $\mathsf{sim}_\times$ for the normal subgroup. This construction is sketched in Figure 3.12.

Concise implementation of reindexing. We can make one interesting improvement over a generic application of Lemmas 17 and 18: the structure of the mixing function ϕ, which specifies the semidirect product, is extremely regular. Very fortunately, the structure of ϕ allows us to avoid any dependence on the size of the wreath product group ($|N|^{|H|} \cdot |H|$) in the size measures of the implementation. A general automorphism on $N \times \cdots \times N$ is specified by its $|N|^{|H|}$ values. However, in this case, ϕ is just a permutation, specified by how each of the $|H|$ channels should switch places. Thus, much like the function composition gadget in Theorem 10, we can construct a simpler MLP than the generic one from Lemma 11.

Specifically, we would like to approximate the function $\phi : H \times (N \times \cdots \times N) \to (N \times \cdots \times N)$, which simply applies π_h to the indices:

$$\phi_h(g^{(1)}, \ldots, g^{(|H|)}) := (g^{(\pi_h(1))}, \ldots, g^{(\pi_h(|H|))}).$$

In the component neural networks' representation space, we need the MLP to implement

$$\left(\mathsf{sim}_N.E(g^{(1)}), \ldots, \mathsf{sim}_N.E(g^{(|H|)}), \mathsf{sim}_H.E(h)\right) \mapsto \left(\mathsf{sim}_N.E(g^{(\pi_h(1))}), \ldots, \mathsf{sim}_N.E(g^{(\pi_h(|H|))})\right),$$

recalling that the elements of g, h are represented by integer vectors with ∞-norm at most $\mathsf{sim}_{\{N,H\}}.\mathsf{repBound}$. Notice that when the representation of $|H|$ is a single integer, restricting to any particular coordinate in the representation of an element g, this is the same composition problem of function transition maps solved by Lemma 14 in the proof of Theorem 10, which uses its left inputs to permute its right inputs (modulo converting the representations from $\{0, \ldots, |H| - 1\}$ to $\{1, \ldots, |H|\}$, which we can do by shifting the indicators at the input and final-layer output weights). Thus, $|N| \cdot \mathsf{sim}_N.\mathsf{dim}$ parallel copies of the 3-layer function composition MLP suffice, yielding

$$d_1 = 4|H|^2|N| + |H| \cdot |N| < 5|H|^2|N|, \quad d_2 = |H|^2|N|,$$

$$\|W_1\|_\infty \le 4|H|, \quad \|b_1\|_\infty \le 6|H|, \quad \|W_2''W_2'\|_\infty \le 4|H|, \quad \|b_2''\|_\infty = |H|, \quad \|W_3\|_\infty = 1.$$

When the information about group elements in H is encoded by multiple integers, it is straightforward to extend this construction, by replacing the one-dimensional indicator with the multidimensional indicator from Lemma 11. We will skip the details of this case, since our final results are only about solvable groups; when we want to simulate a general group extension, it will always come from the composition series, so that H is always a cyclic group of prime order. □

Thus, for general group extensions G, we can construct $\mathsf{sim}_\wr$, the wreath product simulator for $N \wr H$, and combine the individual simulators. Note that we can throw away the excess group elements from the simulator: only include in $\mathsf{sim}_\wr.E, \mathsf{sim}_\wr.W$ the group elements which correspond to the subgroup isomorphic to G. Then, no part of this construction needs to maintain a width or matrix entry scaling with $|N \wr H|$.

Putting all of this together, we state an intermediate theorem, which is our most general result for groups:

Theorem 16 (Simulation of solvable groups)**.** *Let G be a solvable group which is isomorphic to a permutation group on n elements. Let $T \geq 1$. Then, there is a Transformer network $\mathsf{sim} := \mathsf{sim}(G, T)$ which simulates G at length T, for which we have the following size bounds:*

- $\mathsf{sim.depth} \leq 3 \log_2 |G|$.
- $\mathsf{sim.dim} \leq 2|G|$.
- $\mathsf{sim.heads} \leq 2|G|$.
- $\mathsf{sim.headDim} = 1$.
- $\mathsf{sim.mlpWidth} \leq 20nT|G|$.
- $\mathsf{sim.normBound} \leq 6nT$.
- $\mathsf{sim.repDim} \leq 2|G|$.
- $\mathsf{sim.repSize} \leq n$.

Proof. Let

$$G = H_\ell \triangleright H_{\ell-1} \triangleright \cdots \triangleright H_1 \triangleright H_0 = 1$$

denote the composition series. Then, because G is solvable, all of the quotient groups $K_i := H_{i+1}/H_i$ are abelian, thus cyclic groups of prime order. Since G is assumed to be a subgroup of S_n, none of these primes can be greater than n. Thus, every quotient group K_i in the chain satisfies $2 \leq |K_i| \leq n$. Also, note that the length of the composition series ℓ is at most $\log_2(|G|)$ (since each inclusion halves the size of the group).

We start with a simulation of H_1, which must be a cyclic group, and build the sequence of group extensions recursively until we obtain G. In the worst case (in the sense that the implementation size bounds from Lemma 19 are maximized), each step in the composition series must be manifested by a wreath products with $K := C_n$. Recall that we have:

- $\mathsf{sim}_K.\mathsf{depth} = 1$.
- $\mathsf{sim}_K.\mathsf{dim} = 3$.
- $\mathsf{sim}_K.\mathsf{heads} = 1$.
- $\mathsf{sim}_K.\mathsf{headDim} = 1$.
- $\mathsf{sim}_K.\mathsf{mlpWidth} = 4nT$.
- $\mathsf{sim}_K.\mathsf{normBound} \leq 6nT$.
- $\mathsf{sim}_K.\mathsf{repDim} = 1$.
- $\mathsf{sim}_K.\mathsf{repSize} \leq n$.

At each step $i = 0, \ldots, \ell - 1$, Lemma 19, with $H := K_i, N := H_i$, implies:

- $\mathsf{sim}_{H_{i+1}}.\mathsf{depth} \le \mathsf{sim}_{K_i}.\mathsf{depth} + 3$ (1 more layer to simulate the cyclic group K_i, and 2 from the wreath product's mixing operations).
- $\mathsf{sim}_{H_{i+1}}.\mathsf{dim} \le |K_i| \cdot \mathsf{sim}_{H_i}.\mathsf{dim} + 1$ (noting that all of the components can reuse the same $\perp$ and positional encoding dimensions).
- $\mathsf{sim}_{H_{i+1}}.\mathsf{heads} \le |K_i| \cdot \mathsf{sim}_{H_i}.\mathsf{heads} + 1$.
- $\mathsf{sim}_{H_{i+1}}.\mathsf{headDim} \le \max\{1, 1, \ldots, 1\} = 1$.
- $\mathsf{sim}_{H_{i+1}}.\mathsf{mlpWidth} \le \max\{|K_i| \cdot \mathsf{sim}_{H_i}.\mathsf{mlpWidth}, 4nT, 5|K_i|^2 \cdot |H_i|\}$.
- $\mathsf{sim}_{H_{i+1}}.\mathsf{normBound} \le \max\{6nT, 6\,|K_i|\}$.
- $\mathsf{sim}_{H_{i+1}}.\mathsf{repDim} = |K_i| \cdot \mathsf{sim}_{H_i}.\mathsf{repDim} + 1$.
- $\mathsf{sim}_{H_{i+1}}.\mathsf{repSize} \le n$.

Iterating these recursive inequalities $\ell \le \lfloor \log_2 T \rfloor$ times gives us the desired bounds. Note that we are using Lagrange's theorem ($\prod_i |K_i| = |G|$), as well as the fact that for positive integers $m_1, \ldots, m_\ell \ge 2$, we have a bound on the series of prefix products: $\sum_i \prod_{j \le i} m_i \le 2 \prod_{j \le \ell} m_i$.

$\square$

3.4.4.3.4 Simulating semigroups

Now, using this construction and the results developed in the previous section for groups, we complete the construction for semigroups:

- We combine the memory gate construction (Lemma 16) and any network simulating a group to implement the corresponding permutation-reset semiautomaton (Definition 7), the elements of the cascade in Theorem 15.
- To finish, we implement the cascade product (Definition 6) of these permutation-reset semiautomata, guaranteed to exist by Theorem 15. This gives the full result.

First, we summarize the findings of Lemma 16, naming this neural network $\mathsf{sim}_{\mathrm{M}}$ in our "object-oriented" notation. Note that since we are no longer simulating canonical group semiautomata past this point, $\mathsf{repDim}, \mathsf{repSize}$ are no longer well-defined.

- $\mathsf{sim}_M.\mathsf{depth} = 1$.
- $\mathsf{sim}_M.\mathsf{dim} = 4$.
- $\mathsf{sim}_M.\mathsf{heads} = 1$.
- $\mathsf{sim}_M.\mathsf{headDim} = 2$.
- $\mathsf{sim}_M.\mathsf{mlpWidth} = 4|Q|$.
- $\mathsf{sim}_M.\mathsf{normBound} \le 2T \log(|Q|\, T)$.

Lemma 20 (Simulating a permutation-reset semiautomaton)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a permutation-reset semiautomaton (see Definition 7), and let G denote its permutation group. Let $T \geq 1, q_0 \in Q$. Let $\mathsf{sim}_G := \mathsf{sim}(G, T)$ be a Transformer network which continuously simulates G at length T. Then, there is a Transformer network sim'_G which continuously simulates $\mathcal{A}_{T,q_0}$, with size bounds:*

- $\mathsf{sim}'_G.\mathsf{depth} = \mathsf{sim}_G.\mathsf{depth} + \mathsf{sim}_M.\mathsf{depth} + 1 \leq 3 \log_2 |G| + 2.$
- $\mathsf{sim}'_G.\mathsf{dim} = \mathsf{sim}_G.\mathsf{dim} + \mathsf{sim}_G.\mathsf{repDim} + \mathsf{sim}_M.\mathsf{dim} \leq |G| + |Q| + 4.$
- $\mathsf{sim}'_G.\mathsf{heads} = \mathsf{sim}_G.\mathsf{heads} + \mathsf{sim}_M.\mathsf{heads} \leq 2|G| + 1.$
- $\mathsf{sim}'_G.\mathsf{headDim} = \mathsf{sim}_G.\mathsf{headDim} + \mathsf{sim}_M.\mathsf{headDim} + \mathsf{sim}_G.\mathsf{repDim} \leq |Q| + 3.$
- $\mathsf{sim}'_G.\mathsf{mlpWidth} = \mathsf{sim}_G.\mathsf{mlpWidth} + \mathsf{sim}_M.\mathsf{mlpWidth} + |G|^2|Q| \leq 20nT|G| + 4|Q| + |G|^2|Q|.$
- $\mathsf{sim}'_G.\mathsf{normBound} \leq \max\{\mathsf{sim}_G.\mathsf{normBound}, \mathsf{sim}_M.\mathsf{normBound}, 6|Q|\} \leq 6|Q|\, T \log T.$

Proof. Without loss of generality, we will let $Q := [|Q|]$.

We split the embedding space in our construction into two channels: the $\mathsf{sim}_G.\mathsf{dim}$ dimensions used by G, and a channel consisting of 4 additional dimensions, to be used by a copy of the memory semiautomaton, whose symbol set is Q. Let us call these the G and M channels. For the reset symbols, let $E_M(\sigma)$ denote the 4-dimensional encoding of σ from the memory semiautomaton.

Since we defined G to be isomorphic to the permutation group associated with $\mathcal{A}$, there is a bijection $\Phi : G \to S_Q$ between group elements and permutations on Q. We choose the embedding E as follows:

$$E(\sigma) := \begin{cases} \mathsf{sim}.E(\Phi^{-1}(\delta(\cdot, \sigma))\ (G \text{ channel})\ + E_M(\perp)\ (M \text{ channel}), & \text{bijections } \delta(\cdot, \sigma) \\ \mathsf{sim}.E(e_G)\ (G \text{ channel})\ + E_M(q_\sigma)\ (M \text{ channel}), & \text{resets } \delta(\cdot, \sigma) = q_\sigma \end{cases}.$$

Let L_G denote $\mathsf{sim}_G.\mathsf{depth}$.

Layers 1 through L_G: group simulation. The first L_G layers are chosen to be a copy of $\mathsf{sim}_G.\theta$ in the G channel, and only residual connections in the M channel. At the end of this, given any inputs $\sigma_{1:T}$ which map via Φ^{-1} to g_t (letting the group operation be identity when σ_t is a reset symbol), the outputs in the G channel will be $d_G := \mathsf{sim}_G.\mathsf{repDim}$-dimensional encodings of the prefix group products $g_{\leq t} = g_t g_{t-1} \cdots g_1$. Now, letting $r(t)$ denote the most recent reset ($\tau \leq t$ such that σ_τ is a reset token), we notice that the state we want can be derived from this sequence:

$$q_t = \Phi(g_t g_{t-1} \cdots g_{r(t)}) q_{\sigma_{r(t)}} = \Phi(g_{\leq t} \cdot g_{\leq r(t)}^{-1})(q_{\sigma_{r(t)}}). \tag{3.2}$$

Here, if there have been no resets up to time t, we define $r(t)$ to be 0. We treat q_0 like a reset symbol at the beginning of the sequence. Also, note that our canonical group semiautomaton simulator always uses $g_0 = e_G$ as its initial state.

Layer $L_G + 1$: memory lookup and copy. To implement the above, at layer $L_G + 1$, we put a copy of the memory semiautomaton in channel M, setting its initial state to q_0. We will modify this construction slightly, extending W_V with the identity matrix on the d_G group element encoding dimensions of channel G. Intuitively, when the memory unit "fetches" the last non-$\perp$ token, we would like it to copy the corresponding $g_{\leq t}$. Note that sim_G.repSize, the ∞-norm bound on the group element encodings, is at most $|Q|$ by Theorem 16, so we do not need to modify the W_Q, W_K norms to increase the attention head's precision. The final modification is that we append to W_C an identity matrix copying the d_G embedding dimensions to a new d_G dimensional channel, which we will call the I (for "invert") channel (set to 0 in the embedding all preceding layers). Then, by Lemma 16, at the end of this layer, at each position t, the M channel will contain $q_{\sigma_{r(t)}}$ in dimension 1, and $g_{\leq r(t)}$ in channel I. Finally, in channel G, we use only residual connections, preserving $g_{\leq t}$ in channel G.

Layer $L_G + 2$: applying $\Phi(gh^{-1})$ pointwise. This finally allows us to execute Equation 3.2 at each position t. We use one more Transformer layer, with attention block implementing identity. The MLP memorizes the function $(g, h, q) \mapsto \Phi(gh^{-1})(q) \cdot e_1$ (the coordinate is selected arbitrarily), with the concatenated ($d_{\text{inv}} := 2 \cdot \mathsf{sim}_G$.repSize $+ 1$)-dimensional encodings on the (G, I, M) channels, whose activations have ∞-norms bounded by $|Q|$. We invoke Lemma 11, with parameters

$$\Delta = 1, d_{\text{in}} = d_{\text{inv}}, B_x = |Q|, B_y = |Q|,$$

giving us a construction with

$$d_1 \leq 4d_{\text{inv}}(|Q| + 1), \quad d_2 \leq |G|^2|Q|,$$

$$\|W_1\|_\infty \leq 4, \quad \|b_1\|_\infty \leq 6|Q|, \quad \|W_2\|_\infty \leq 1, \quad \|b_2\|_\infty \leq d_{\text{inv}}, \quad \|W_3\|_\infty \leq |Q|.$$

From this output, W simply decodes the correct q_t from dimension 1. □

Lemma 21 (Implementing the transformation cascade)**.** *Let $\mathcal{A} = (Q, \Sigma, \delta)$ be a semiautomaton, and let $T \geq 1$. Let $\{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(n)}; \phi^{(2)}, \ldots, \phi^{(n)}\}$ be the transformation cascade (Definition 6) which simulates $\mathcal{A}$, as guaranteed by Theorem 15. For each i, let sim_i be a Transformer network which continuously simulates the permutation-reset semiautomaton $\mathcal{A}^{(i)}$ at length T. Then, there is a Transformer network $\mathsf{sim}_{\mathcal{A}}$ which simulates $\mathcal{A}$ at length T. Its size bounds are:*

- $\mathsf{sim}_{\mathcal{A}}.\mathsf{depth} = |Q| \cdot (\max_i\{\mathsf{sim}_i.\mathsf{depth}\} + 1) - 1 \leq 3|Q|^2 \log |Q| + 7|Q|$.
- $\mathsf{sim}_{\mathcal{A}}.\mathsf{dim} = \sum_{i=1}^n \mathsf{sim}_i.\mathsf{dim} + 1 \leq 2^{|Q|}(|\mathcal{T}(\mathcal{A})| + |Q| + 4) + 1$.
- $\mathsf{sim}_{\mathcal{A}}.\mathsf{heads} = \sum_{i=1}^n \mathsf{sim}_i.\mathsf{heads} \leq 2^{|Q|+1}(|\mathcal{T}(\mathcal{A})| + 1)$.
- $\mathsf{sim}_{\mathcal{A}}.\mathsf{headDim} = \max_{i=1}^n\{\mathsf{sim}_i.\mathsf{headDim}\} \leq |Q| + 3$.
- $\mathsf{sim}_{\mathcal{A}}.\mathsf{mlpWidth} = \sum_{i=1}^n \mathsf{sim}_i.\mathsf{mlpWidth} + 2^{|Q|}\, |Q|^{2^{|Q|}}\, |\Sigma| \leq 2^{|Q|}(20|Q|\, |\mathcal{T}(\mathcal{A})|\, T + 4|Q| + |\mathcal{T}(\mathcal{A})|^2|Q| + |Q|^{2^{|Q|}}\, |\Sigma|)$.
- $\mathsf{sim}_{\mathcal{A}}.\mathsf{normBound} \leq \max_{i=1}^n\{\mathsf{sim}_i.\mathsf{normBound}\} \cup \{2^{|Q|}(|\mathcal{T}(\mathcal{A})| + 5|Q|)\} + 6\max\{|Q|, |\Sigma|\}$
  $\leq \max\{6|Q|\, T \log T, 2^{|Q|}(|\mathcal{T}(\mathcal{A})| + 5|Q|)\} + 6\max\{|Q|, |\Sigma|\}$.

Proof. At this point, most of the work has been done for us.

We create a separate channel i for each component permutation-reset semiautomaton $\mathcal{A}^{(i)}$. This requires a total of $\sum_{i=1}^n \mathsf{sim}_i.\mathsf{dim}$ embedding dimensions. In addition to these channels, we keep one dimension (with residual connections throughout the network) to represent the input σ_t. Let e_Σ denotes the unit vector along this coordinate. Choosing an arbitrary enumeration to identify Σ with $[\Sigma]$, we select the embeddings to be $E(\sigma) := \sigma \cdot e_\Sigma$.

The L layers of $\mathsf{sim}_{\mathcal{A}}$ are divided into $|Q|$ *subnetworks* (which are just Transformer networks), which we will concatenate sequentially at the end. Let these subnetworks be indexed by $\widetilde{\ell} \in \{1, \dots, \widetilde{L}\}$. Each subnetwork starts with a parallel simulation (as in the direct product construction of Lemma 17, padding with layers implementing identity if their depths do not match), combining all of the $\mathsf{sim}_i.\theta$ in the ℓ-th level of the Krohn-Rhodes decomposition, as defined by Theorem 15. Each $\mathsf{sim}_i.\theta$ is chosen to operate in its own channel i. We add residual connections on all of the input/output dimensions in each channel. Then, at the end of each subnetwork except the final one ($1 \le \widetilde{\ell} \le \widetilde{L} - 1$), we append one more Transformer layer with identity attention block, whose MLP implements the "wiring" specified by $\phi^{(i)}$ from the next level of the decomposition.

Namely, we invoke Lemma 11 with $\Delta = 1, B_x = \max\{|Q|, |\Sigma|\}, B_y = |\Sigma|$, giving us for each pre-final-layer i an MLP which represents the function

$$(\mathsf{sim}_1.W^{-1}(q^{(1)}), \dots, \mathsf{sim}_{i-1}.W^{-1}(q^{(i-1)}), E(\sigma)) \mapsto \mathsf{sim}_i.E(\phi(q^{(1)}, \dots, q^{(i-1)}, \sigma)),$$

where the inputs are stored in the respective $i' < i$ and Σ channels, and the output is written to the i channel. Here, the number of input dimensions is

$$d_{\text{in}} = \sum_{i' < i} \mathsf{sim}_{i'}.\mathsf{dim} + 1 \le 2^{|Q|}(|\mathcal{T}(\mathcal{A})| + |Q| + 4) + 1 \le 2^{|Q|}(|\mathcal{T}(\mathcal{A})| + 5|Q|).$$

Since the state encodings for each predecessor semiautomaton $i' < i$ are $|Q|$-bounded integer vectors and Σ has been assumed to be a $|\Sigma|$-bounded positive integer, it suffices to use $d_1 = 4 d_{\text{in}} \max\{|Q|, |\Sigma|\}$. The second hidden layer's width d_2 is the number of possible inputs $|\mathcal{X}|$, which is bounded by $|Q|^{2^{|Q|}} \cdot |\Sigma| > d_1$. The weights satisfy

$$\|W_1\|_\infty \le 4, \quad \|b_1\|_\infty \le 6 \max\{|Q|, |\Sigma|\},$$

$$\|W_2\|_\infty \le 1, \quad \|b_2\|_\infty \le d_{\text{in}}, \quad \|W_3\|_\infty \le |\Sigma|, \quad b_3 = 0.$$

Between i in the same layer, these routing constructions need to be executed in parallel, so this incurs another multiplicative factor in the width, bounded conservatively by $2^{|Q|}$.

The final construction concatenates these blocks, so that at the output of the last layer, every channel i contains a representation of its corresponding component's semiautomaton Q_i. The $\mathcal{W}$ guaranteed by Theorem 15 suffices for the overall choice of W. □

3.4.4.4 Proof of Theorem 12: Even shorter shortcuts for gridworld

Recall the gridworld semiautomaton in Example 3, where the state ($Q = \{0, 1, \dots, S\}$) either move to the adjacent state based upon seeing input token L or R (modulo boundary effects), or stay unmoved upon seeing $\perp$. More formally, the transition function is defined as:

$$\begin{aligned} \delta(q, L) &= \max(q - 1, 0) \\ \delta(q, R) &= \min(q + 1, S) \\ \delta(q, \perp) &= q. \end{aligned}$$

In this section, we will show how to implement gridworld simulation using only 2 Transformer layers. Here we restate the theorem in full generality:

Theorem 12 (Even shallower shortcuts for gridworld)**.** *For each positive integer T, Transformers can simulate the $(S+1)$-state gridworld semiautomaton with 2 attention layers, where the MLP has either (i) depth $O(\log S)$, width $O(T+S)$, or (ii) depth $O(1)$, width $O(T)+2^{O(S)}$. The weight norms are bounded by* $\operatorname{poly}(T)$.

The depth in (i) can be reduced to $O(S)$ if we allow max pooling, and the dependence on T in the width can be removed with sinusoidal activation. We discuss this in detail after the proof along with generalization to the k-dimensional gridworld case.

Note that, in order to find the current state, we need to only know the most recent time at which the semiautomata was at a boundary. It is not immediately obvious how to compute the most recent boundary, if one is not allowed to use the trivial sequential simulation algorithm. Our key insight is that this *boundary detector* can be computed without needing to parse the entire sequence, using the most recent $S+1$ distinct values of the prefix sums in the sequence.

This algorithm is especially well-suited to the Transformer architecture since: (i) the prefix sum can be computed using one attention layer as in Lemma 15, and (ii) the identification of distinct values can be implemented by a sparse *value-dependent* lookup similar to the memory lookup in Lemma 16 with the help of the *self*-attention (context-dependent retrieval, as opposed to a static lookup), and (iii) the positional weight sharing and causal masking enable all of these computations to be performed in parallel. Overall, Theorem 12 consists of a concise implementation which executes all of these most-recent-boundary detectors in parallel.

In what follows, we first describe the algorithm (Algorithm 1) for computing the state of the semiautomata using the $S+1$ distinct prefix sum values, and give a proof of its correctness. Subsequently, we formalize the Transformer construction that implements the algorithm. A consolidated list of notations used in the algorithm as well as the proofs is provided in Table 3.1 for the reader.

3.4.4.4.1 The algorithm solving 1D gridworld

To convey the essence of the full construction, we first provide pseudocode (rather than Transformer weights) for computing the *final* state q_T (rather than the entire state sequence).

We map actions $\sigma \in \{L, R, \perp\}$ to $\widetilde{\sigma} \in \{-1, 1, 0\}$, i.e. $L \mapsto -1$, $R \mapsto 1$, and $\perp \mapsto 0$. Let $\widetilde{\sigma}^{(:)}$ denote the sequence of mapped actions, and let 0 be the initial state. The algorithm (Algorithm 1) has two steps: first, we identify the last time the agent is at a boundary (wall) and the type of the boundary (i.e. state 0 or state S). The final state is then simply the sum of all actions in the sub-sequence, shifted by the last boundary, which is easily computable with 1 attention layer (Lemma 15). Our key insight is that we can identify the boundary using $O(S)$ attention heads in two attention layers, and therefore do not require a recursive computation from the start state (with depth T).

To show the correctness of Algorithm 1, it suffices to show that the boundary state is detected correctly, since after that there is no more boundaries and the only step remaining is to calculate the sum of the actions. Let $t_{\text{uniq}}, t_{\min}, t_{\max}$ be as defined in Algorithm 1. Then:

Lemma 22. *If $t_{\min} > t_{\max}$, then state at $t_{\min}$ is 0, otherwise state at $t_{\max}$ is S.*

Proof. The proof follows from two observations:

Notation	Definitions
σ	An input token; $\sigma \in \{L, R, \perp\}$.
$\widetilde{\sigma}$	A mapped input token; $\widetilde{\sigma} = -1$ if $\sigma = L$, $\widetilde{\sigma} = 1$ if $\sigma = R$, and $\widetilde{\sigma} = 0$ if $\sigma = \perp$.
Used in Algorithm 1:	
q_t	The state at position $t \in [T]$; $q_t \in \{0, 1, \cdots, S\}$.
$\boldsymbol{z} \in \mathbb{Z}^T$	Prefix sums at all T positions.
t_{uniq}	The most recent position for which the prefix sums $\boldsymbol{z}_{t_{\text{uniq}}:T}$ contain $S+1$ unique values.
$t_{\max}, t_{\min}$	The positions corresponding to the max/min prefix sum among positions $t_{\text{uniq}}, \ldots, T$.
t_{final}	The position of the last boundary state, defined as $t_{\text{final}} := \max\{t_{\max}, t_{\min}\}$.
Used in the Transformer construction:	
γ_t	The positional encoding for position $t \in [T]$, defined as $\gamma_t := \log(2T - t)$.
$x^{(1)}_{\text{attn}}[t]$	The output of the first layer attention at position $t \in [T]$, defined as $x^{(1)}_{\text{attn}}[t] := \frac{1}{2T} \sum_{i \in [t]} s_i$.
$x^{(1)}_{\text{mlp}}[t]$	The output of the first layer MLP at position $t \in [T]$, defined as $x^{(1)}_{\text{mlp}}[t] := [x^{(1)}_{\text{attn}}[t], \gamma_t, 1, \cos(x^{(1)}_{\text{attn}}[t]\pi), \sin(x^{(1)}_{\text{attn}}[t]\pi)]$.
$j^{(s)}_{\max}$	The position which achieves the max attention score for the s^{th} head at time $t \in [T]$ (t is omitted for notational convenience), for $s \in [0, 1, \cdots, 2S]$.
$x^{(2)}_{\text{attn}}[t]$	The output of the second layer attention at position $t \in [T]$, defined as $x^{(2)}_{\text{attn}} := [\gamma_{j^{(0)}_{\max}}, \gamma_{j^{(1)}_{\max}}, \cdots, \gamma_{j^{(2S)}_{\max}}]$.
$x^{(2)}_{\text{mlp}}[t]$	The output of the second layer MLP at position $t \in [T]$, which gives the state at t.

Table 3.1: Notations for the proof of Theorem 12.

1. $\min\{t_{\min}, t_{\max}\} = t_{\text{uniq}}$,
2. Suppose $t_{\min} = t_{\text{uniq}}$ (the argument is symmetric for $t_{\max} = t_{\text{uniq}}$). Then $q_{t_{\max}} = S$.

First note that $\widetilde{\sigma} \in \{\pm 1\}$ which implies that the prefix sums increment or decrement by 1 at each index. Therefore, $z_{t_{\max}} - z_{t_{\min}} = S + 1$. This also applies that between (and including) $t_{\max}$ and $t_{\min}$, there must be indices such that they traverse the $S+1$ distinct values. Since we take the shortest suffix satisfying this, $t_{\text{uniq}} \geq \min\{t_{\max}, t_{\min}\}$. This proves Observation 1.

Assume $t_{\min} = t_{\text{uniq}}$. We can break the analysis into the following 2 cases:

(a) *The $S+1$ distinct values correspond to $S+1$ distinct states (covering both boundaries).* This implies that the minimum and maximum out of these distinct prefix sums must correspond to the boundaries, that is, $q_{t_{\max}} = S$ and $q_{t_{\min}} = 0$.

```
Algorithm 1: 1D gridworld: computing the final state
    Data: $\widetilde{\sigma} \in \{\pm 1\}^T, S \in \mathbb{Z}^+$
    Result: Final state $y_T \in \{0, 1, \cdots, S\}$
    // Pad tokens so there are at least $S+1$ distinct values
    $\widetilde{\sigma} \leftarrow [\underbrace{-1, -1, \cdots, -1}_{S+1}, \widetilde{\sigma}]$
    // Calculate the prefix sum for each index
    $\boldsymbol{z} \leftarrow \mathsf{prefix\_sum}(\widetilde{\sigma})$    (i.e. $\boldsymbol{z}_t \leftarrow \sum_{\tau=1}^{t} \widetilde{\sigma}_t$)
    // Find a substring containing $S+1$ unique values
    $t_{\text{uniq}} \leftarrow \max\{t : |\mathsf{set}(\boldsymbol{z}_{t:})| = S+1\}$
    // Find positions of the last max and min values
    $t_{\min} \leftarrow \max\{t : z_t = \min_{\tau \geq t_{\text{uniq}}} z_\tau\}$
    $t_{\max} \leftarrow \max\{t : z_t = \max_{\tau \geq t_{\text{uniq}}} z_\tau\}$
    // Identify the type of boundary
    if $t_{\min} > t_{\max}$ then
        $\mathsf{boundary} \leftarrow 0$
    else
        $\mathsf{boundary} \leftarrow S$
    end
    // The final state is the sum of the substring after the last boundary
    $t_{\text{final}} \leftarrow \max\{t_{\min}, t_{\max}\}$
    $y_T = z_T - z_{t_{\text{final}}} + \mathsf{boundary}$
```

(b) *The $S+1$ distinct values correspond to fewer than $S+1$ distinct states.* This implies that only one of the two boundaries is visited in the sequence starting from t_{uniq}. In order to get $S+1$ distinct values, it must be that this boundary wall is hit, i.e., the sequence tries to make a move that the boundary blocks. If the sequence does not hit a boundary, then at every time the same state is revisited, the prefix sum must be the same, and we will not be able to get $S+1$ distinct values. Since $t_{\min} = t_{\text{uniq}}$, we claim that the visited boundary must be S. Suppose this is not true, then the boundary visited is 0. This implies that $q_{t_{\min}} = 0$. Since the sequence does not hit S, at any position it is at state 0 before hitting the wall, the value will be $z_{t_{\min}}$. Thus, when the sequence first hits the wall at 0 (say index τ), then $z_\tau = z_{t_{\min}} - 1$ which is not possible by definition of $t_{\min}$. Thus, the boundary must be S.

□

Given the above, our algorithm identifies the boundary correctly and then can just use the prefix sum to evaluate the current state.

3.4.4.4.2 Transformer construction for Algorithm 1

In this section we will show how to simulate Algorithm 1 using a 2-layer Transformer with $2S$ attention heads.

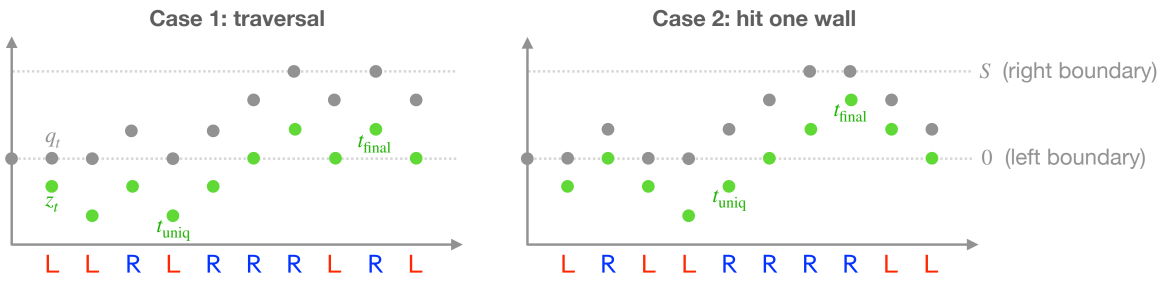


Figure 3.13: Illustrated examples for 4-state gridworld ($Q = \{0, 1, 2, 3\}$). The algorithms compute the prefix sums z_t (as if there were no boundaries; shown as green dots); intuitively, it might seem like they have no direct relationship with the state sequence q_t (gray dots). However, defining t_{uniq} as the start of the shortest suffix containing $S + 1$ distinct prefix sums, and t_{final} as the most recent minimum or maximum point within this suffix, our case analysis shows that $q_{t_{\text{final}}}$ is always a boundary, resulting in a parallel simulation algorithm.

Proof of Theorem 12. In our construction, the first attention layer will compute the prefix sums. This can be mapped to a cyclic group from Lemma 15, however for completeness, we will restate the main construction. The MLP in the first layer will map this prefix sum to a circular embedding (see Proposition 2). The second layer attention will use the circular embedding structure to find $S + 1$ closest distinct values to the current value z_t (suppose we are considering position $t \in [T]$) by identifying the positions for closest values in the set $\{z_t - S, z_t - S + 1, \ldots, z_t - 1, z_t + 1, z_t + 2, \ldots z_t + S\}$, i.e. S closest distinct values smaller than z_t, and S values larger than z_t. This closest distinct value construction can be viewed as a position dependent flip-flop monoid construction, where we need to identify the closest position with a particular action. Note that this set of values would contain the distinct $S + 1$ values needed by the Algorithm 1, hence the second layer MLP can implement the state computation using these values.

Input representation: We select input symbol embedding $E(\sigma) = \tilde{\sigma} \cdot e_1 \in \mathbb{R}^d$ where $\tilde{\sigma}$ is the action corresponding to σ, that is, $s = \begin{cases} -1 & \text{if } \sigma = L \\ 1 & \text{if } \sigma = R \\ 0 & \text{otherwise} \end{cases}$. We will use positional encoding $P_{t,:} := \gamma_t e_3$, where $\gamma_t := \log(2T - t)$ is such that $\frac{1}{e^{\gamma_t} + t} = \frac{1}{2T}$. We will include an extra position $\perp$, with embedding $E(\perp) := e_2$ and position encoding $P_{\perp,:} := 0$. Think of this as padding at position 0; it is not masked by the causal attention mask.

Prefix sum (Layer 1 attention): The attention construction for the first layer, in full detail:

- We select $d := 4, k := 1, H := 1$.
- Select $W_Q := e_3, W_K := e_2, W_V := e_1, W_C^\top := e_4$.

With this attention module, the 4th channel of the output at position t is $x_{\text{attn}}^{(1)}[t] = \frac{1}{2T} \sum_{i \in [t]} s_i$, which is the prefix sum scaled down by $1/2T$.

Circular embedding (MLP 1): The first MLP maps $x_{\text{attn}}^{(1)}[i] \mapsto [\cos(x_{\text{attn}}^{(1)}[i]\pi), \sin(x_{\text{attn}}^{(1)}[i]\pi)]$, where cos, sin are calculated up to $O(\log T)$ precision using the construction in Lemma 10 with width $4(2T+1)$. [23] and weight norms at most $8T$. Together with the input using the skip connection (for γ_i) we get $x_{\text{mlp}}^{(1)}[t] := \left[x_{\text{attn}}^{(1)}[t], \gamma_t, 1, \cos(x_{\text{attn}}^{(1)}[t]\pi), \sin(x_{\text{attn}}^{(1)}[t]\pi)\right]$ as the embedding to be input to the second attention layer.

Finding closest $S+1$ values (Layer 2 attention): Our goal is to find the shortest subsequence (looking back from the current position t) that contains $S+1$ distinct values for $x_{\text{attn}}^{(1)}$; that is, we want to find the max $\tau \leq t-S$ such that $\left|\{x_{\text{attn}}^{(1)}[i]\}_{i=\tau}^{t}\right| = S+1$. We will do this by using $2S+1$ heads such that $\forall s \in \{0, 1, \cdots, 2S\}$, the attention score for the s_{th} head on position $i \in [T]$ satisfies

$$\tilde{\alpha}_{t,i}^{(s)} := \left\langle (W_Q^{(s)})^\top x_{\text{mlp}}^{(1)}[t], (W_K^{(s)})^\top x_{\text{mlp}}^{(1)}[i] \right\rangle$$
$$\begin{cases} = 1 - c\log(2T-i), & \text{if } x_{\text{attn}}^{(1)}[i] = x_{\text{attn}}^{(1)}[t] + \frac{s-S}{2T}, \\ \leq 1 - c\log(2T-i) - \frac{\pi^2}{8T^2}, & \text{otherwise,} \end{cases}$$

where $c = \frac{\pi^2}{(16\log 2)T^2}$. That is, for any i, j s.t. $x_{\text{attn}}^{(1)}[i] = x_{\text{attn}}^{(1)}[t] + \frac{s-S}{2T}$ (matched) and $x_{\text{attn}}^{(1)}[j] \neq x_{\text{attn}}^{(1)}[t] + \frac{s-S}{2T}$ (unmatched), the difference in the unnormalized attention weights is lower bounded by $\tilde{\alpha}_{t,i}^{(s)} - \tilde{\alpha}_{t,j}^{(s)} \geq \frac{\pi^2}{16T^2}$. This can be achieved by letting $W_Q^{(s)} := \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -c \\ 0 & 0 & 0 \end{bmatrix} & 0 \\ 0 & \rho_{\theta^{(s)}} \end{bmatrix} \in \mathbb{R}^{5\times 5}$ where $\rho_{\theta^{(s)}}$ the rotation matrix of angle $\theta^{(s)} := \frac{(s-S)\pi}{2T}$, such that $(W_Q^{(s)})^\top x_{\text{mlp}}^{(1)}[t] = \Big[0, -c, 0, \cos\left(\left(x_{\text{attn}}^{(1)}[t] + \frac{s-S}{2T}\right)\pi\right), \sin\left(\left(x_{\text{attn}}^{(1)}[t] + \frac{s-S}{2T}\right)$
$W_K^{(s)}, W_C^{(s)}$ are simply the 5×5 identity matrix, and $W_V^{(s)} = e_1 e_1^\top + e_2 e_2^\top$.

Let $j_{\max}^{(s)}$ denote the position that achieves the max attention score for the s^{th} head, then the output of the s^{th} head [24] is $[x_{\text{attn}}^{(1)}[j_{\max}^{(s)}], \gamma_{j_{\max}^{(s)}}, 0, 0, 0]$. We can ignore the last three coordinates (which are 0) as well as $x_{\text{attn}}^{(1)}[j_{\max}^{(s)}]$, since we will only need the difference $x_{\text{attn}}^{(1)}[t] - x_{\text{attn}}^{(1)}[j_{\max}^{(s)}]$ which is $\frac{s-S}{2T}$ by definition. We then concatenate the outputs from all $(2S+1)$ heads in a $(2S+1)$-dimensional vector $x_{\text{attn}}^{(2)} = [\gamma_{j_{\max}^{(0)}}, \gamma_{j_{\max}^{(1)}}, \cdots, \gamma_{j_{\max}^{(2S)}}]$ as the input to the second layer MLP.

Intuitively, each head in the second attention layer is trying to identify the set of positions for which the prefix sums match a particular value specified by the head. Each head selects the last matching position if such positions exist, and selects t if not. The following observation will be helpful for our subsequent MLP construction: the values of coordinates of $x_{\text{attn}}^{(2)}$ increase on both sides of the S^{th} heads; that is, $x_{\text{attn}}^{(2)}$ satisfies the following:

Lemma 23. *There exist $a < b \in \{0, 1, \ldots, 2S\}$ such that*

$$x_{\text{attn}}^{(2)}[a] > x_{\text{attn}}^{(2)}[a+1] > \ldots > x_{\text{attn}}^{(2)}[S] < x_{\text{attn}}^{(2)}[S+1] < \ldots < x_{\text{attn}}^{(2)}[b]$$

and all $s \in \{0, 1, \ldots, 2S\} \setminus \{a, a+1, \ldots, b\}$ we have $x_{\text{attn}}^{(2)}[s] = \log(2T-t)$.

[23]The width is 1 if we allow sinusoidal activation instead of relu; see the discussion after the proof.

[24]We assume hard attention here for ease of exposition of the proof; soft attention can be handled with Lemma 13 and Lemma 10 as in our previous constructions. This requires norm poly(T) at max.

Proof. Note that $x_{\text{attn}}^{(2)}[s] := \log(2T - \gamma_{j_{\max}^{(s)}})$ which makes the ordering inverse of the position. Observe that the unmatched indices correspond to values that have not been reached. This can only happen for values on either the leftmost or the rightmost coordinates, since the prefix sums are continuous on integers. Now let's prove that the value will be decreasing moving away from index S in both directions. Suppose this was not true, there indeed was $s \le S$ such that $x_{\text{attn}}^{(2)}[s] \ge x_{\text{attn}}^{(2)}[s-1]$ ($s \ge S$ case is identical), then it implies that the closest index that achieved relative value $S - s$ is further away from t than $S - s + 1$. However, since the moves can update the prefix sum by magnitude at most 1, then to get to relative value 0 from relative value $S - s + 1$, we would need to have crossed relative value $S - s$. This implies that there is another position closer to t with this value, contradicting our assumption. This proves the result. □

Computing state (MLP 2): To compute state from the positional information given by $x_{\text{attn}}^{(2)}$, we need to do the following computations:

- *Step 1*: consider $S+1$ windows of size-$(S+1)$, each containing the s^{th} to $(s+S)^{th}$ heads for $s \in \{0, 1, \dots, S\}$, and identify the window that contains positions closest to the end (this would correspond to the closest $S+1$ distinct values to $x_{\text{att}}^{(1)}[t]$);
- *Step 2*: identify the boundary state in the selected window by comparing the indices of the endpoints of the window;
- *Step 3*: output the final state based on the position of the boundary states and its value relative to the current position t.

We will show two constructions for implementing this, one of which will use $O(1)$ depth and $2^{O(S)}$ width, and the other will use $O(\log S)$ depth and $O(S)$ width. The trade-off essentially lies in how a min function is implemented and can be resolved if we allow a min-pooling layer, which we will discuss after the proof.

1. $O(1)$-*depth construction:* The idea is that we can first use $O(1)$ layers to construct "features" that contain all the information needed to determine the state, then a 3-layer MLP with $2^{O(S)}$ width can compute the state as a function of these features by Lemma 11. The features we need are the following (the labels underneath are to be consistent with Figure 3.14, *left*):

$$\underbrace{\{\mathbb{1}[x_{\text{attn}}^{(2)}[s] > x_{\text{attn}}^{(2)}[s+S]]\}_{s=0}^{S}}_{>}, \tag{3.3}$$

$$\underbrace{\{\mathbb{1}[x_{\text{attn}}^{(2)}[s-1] > x_{\text{attn}}^{(2)}[s+S]]\}_{s=0}^{S}}_{>_L}, \quad \underbrace{\{\mathbb{1}[x_{\text{attn}}^{(2)}[s] > x_{\text{attn}}^{(2)}[s+S+1]]\}_{s=0}^{S}}_{>_R}, \tag{3.4}$$

$$\underbrace{\{\mathbb{1}[x_{\text{attn}}^{(2)}[s] = \log(2T-t)]\}_{s=0}^{2S}}_{=}\}. \tag{3.5}$$

Here the feature in 3.3 compares the end points of the $S+1$ windows, the two features in 3.4 compare the window with its adjacent windows on each side, and the last feature in 3.5 will be used to eliminate the irrelevant window. Features in 3.3 and 3.4 can each be computed as a threshold function (at 0) on the difference between the two elements to be compared, which can be implemented using 2 layers by Lemma 12.

(a) First, we show that to compute *Step 1* we only need to compare between adjacent windows whose $S+1$ heads are all matched, which can be computed using the features above. Consider any window starting at $s \in [0,1,\cdots,S]$. On either side, if this window is closer to t than its adjacent window on this side, then it is closer to the end of the boundary than all the windows on this same side, which would imply that we can ignore these non-adjacent windows. To prove this, let's consider the left side (the right side is analogous) and we want to show: For any $s \in [S]$, if $\max\{x^{(2)}_{\text{attn}}[s], x^{(2)}_{\text{attn}}[s+S]\} < \max\{x^{(2)}_{\text{attn}}[s-1], x^{(2)}_{\text{attn}}[s+S-1]\}$, then $\max\{x^{(2)}_{\text{attn}}[s], x^{(2)}_{\text{attn}}[s+S]\} < \max\{x^{(2)}_{\text{attn}}[s-i], x^{(2)}_{\text{attn}}[s+S-i]\}$ for $i > 1$: the *if* condition gives us $x^{(2)}_{\text{attn}}[s-1] > x^{(2)}_{\text{attn}}[s+S]$, and we know from Lemma 23 that

$$x^{(2)}_{\text{attn}}[s-i] > x^{(2)}_{\text{attn}}[s-1] > x^{(2)}_{\text{attn}}[s],$$
$$x^{(2)}_{\text{attn}}[s+S] > x^{(2)}_{\text{attn}}[s+S-1] > x^{(2)}_{\text{attn}}[s+S-i].$$

Combining these inequalities together concludes the proof.

(b) Given the optimal window, we can use feature 3.3 for the relevant window to identify the boundary, since the closer-to-t index gives us the last boundary state (see Algorithm 1 for why this suffices).

(c) Now that we have identified the boundary state (suppose it is at position i), the final state can be computed as the last boundary state (0 or S) plus the difference $x^{(1)}_{\text{attn}}[t] - x^{(1)}_{\text{attn}}[i]$. The difference is built in to the ordering of the heads, hence we have all the information to compute the final state and we are done.

Therefore, we can compute this function using $4S+3$ features each taking value in $\{0,1\}$ and the output having $S+1$ values. These features themselves can be constructed using Lemma 12 with $\Delta = 1/4T$ since the indices are separated by at least this gap. For the indicator index, we can compose two such constructions similar to Lemma 10. This gives us the first layer of MLP with width $O(S)$ and norms $O(T)$. After this, the rest of the function can be constructed using a 3-layer ReLU network with width $2^{O(S)}$ and norms bounded by $O(S)$ using Lemma 11.

2. *$O(\log S)$-depth construction:* An alternative solution to the above is to pay $O(\log(S))$ depth, but reduce the width to be $O(S)$. We will borrow features in equation 3.3-3.5, but construct the MLP explicitly rather than calling Lemma 11 as a black box: the width and depth trade-off essentially correspond to two ways of implementing the min of S numbers. We describe the corresponding MLP by components (Fig 3.14, *right*):

(a) Ignore the unmatched heads: as a preprocessing step for the cleanness of the proof, we use a 1-layer MLP to map $x^{(2)}_{\text{attn}}[s]$ for heads where $x^{(2)}_{\text{attn}}[s] = \log(2T-t)$ (i.e. $j_{\max} = t$) to $\log(2T)$ such that these unmatched heads can be ignored in the following steps. This can be done by multiplying $\log(2T)$ to the threshold function given by Lemma 12 (with $\Delta = \frac{1}{4T}$), where the network has 1 hidden layer with width $2S+1$ and ∞-norm $4T$. Note that this map also changes $x^{(2)}_{\text{attn}}[S]$ (i.e. the center head that corresponds to position t) but this will not affect the correctness of the proof.

(b) Compute the function f_1 in Figure 3.14 (*right*), which computes $f_1(a,b) := (\max\{a,b\}, \mathbb{1}[a > b])$ for $a = x^{(2)}_{\text{attn}}[s], b = x^{(2)}_{\text{attn}}[s+S]\}, \forall s \in \{0,1,\cdots,S\}$. The first coordinate $\max\{a,b\}$ can be implemented using 1 hidden layer with width 1, and $\mathbb{1}[a > b]$ is the same as feature 3.3 and can be implemented with 1 hidden layer by Lemma 12. There are $S+1$ choices of s, hence the

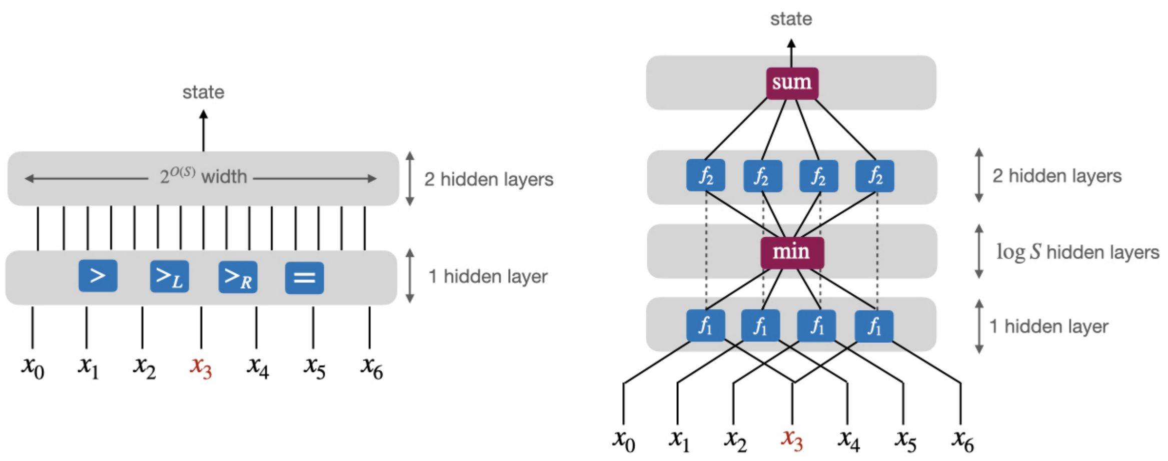


Figure 3.14: Illustration of the two constructions for second-layer MLP in Theorem , with $S = 3$. For ease of readability, we replace $x^{(2)}_{\text{attn}}$ with x. *Left:* $O(1)$-depth solution where the first block compares the comparison and equality features (see equation 3.3–3.5), and the second block computes the state from these features. *Right:* $O(\log S)$-depth solution where first block implements $f_1(a,b) = (\max\{a,b\}, \mathbb{1}[a > b])$, second block does a min-pooling operation, the third block implements f_2 which takes f_1 via a residual connection and output of min-pool to compute the final state.

overall width is $O(S)$. For notational convenience, let's denote $f_1(s) := f_1(x^{(2)}_{\text{attn}}[s], x^{(2)}_{\text{attn}}[s+S])$ (with a slight abuse of notation).

(c) Find the min value of $f_1(s)[1]$, denoted as $f_{1,\min} := \min_s f_1(s)[1]$: This can be achieved using 1 min-pooling layer. If we allow ReLU only, then this can be implemented with pairwise comparison using a network with $\lceil \log S + 1 \rceil$ depth, $3S$ width and and constant weight norm. [25]

(d) Compute the function $f_2^{(s)}$ in Figure 3.14 (*right*): $f_2^{(s)}$ takes two inputs: 1) the second output of f_1, which we denote as $B_s := f_1(s)[2]$, and 2) $M_s := \mathbb{1}[f_{1,\min} \leq f_1(s)[1]]$, which indicates whether the s^{th} window is the closest-to-t window or not and can be computed using a 1-hidden-layer network with width 2. As in the previous construction, the difference $f_{1,\min} - f_1(s)[1]$ is built-in in the ordering of the head and hence does not need to be passed in explicitly. Then by Lemma 11, a 2-hidden-layer network of width $O(1)$ can take B_s, M_s as input and compute $M_s \cdot \left[B(S-s) + (1-B)(2S+1-s)\right]$. The overall width is $O(S)$ for $S+1$ choices of such $f_2^{(s)}$.

(e) Finally, the state is computed as $\sum_s f_2^{(s)}(B_s, M_s)$, which can be implemented with 1 layer of width 1.

$\square$

Improving the construction to remove T width and $\log(S)$ depth. Using standard architectural tools, such as max-pooling, we can improve our construction to get $O(1)$-depth and $O(S)$-width for the MLP.

[25]The log depth is conjectured to be unimprovable; see discussion after the proof.

- *Avoiding width T in the MLP 1 using periodic activations.* As in the modular addition (Lemma 15) construction, we can use sin activations in the MLP to directly compute the circular embeddings that are used as input to the second attention layer. This would require only two hidden nodes in the MLP. Note that we do not need precision greater than $O(\log T)$ for these activations since we are embedding values only as close as $1/\mathsf{poly}(T)$.
- *Avoiding* $\log(S)$ *depth in the MLP 2 using max-pooling.* The $O(\log S)$-depth in MLP 2 is incurred by calculating the min of S numbers and is conjectured to be necessary for ReLU networks [Goel et al., 2017, Mukherjee and Basu, 2017, Hertrich et al., 2021]. However, the depth can be reduced to 1 if we allow max-pooling layers, which are commonly used in both theory and practice [Zhang et al., 2021b, He et al., 2016, Vaswani et al., 2021].

Remark: Yao et al. [2021b] use layer-norm to compute cos and sin embedding with non-uniform angles. This could potentially alleviate the width T concern; we leave this exploration to future work.

Extending beyond 1 dimension. Since a 2-dimensional gridworld is just the direct product of 1-dimensional gridworlds (by the construction in Lemma 17), we can implement both dimensions in parallel by concatenating the network for each dimension. This can be done by doubling the dimensions, parallel attention heads, and parallel hidden units in the MLP. The attention head parameters for each dimension can be chosen to only focus on the relevant dimension and similarly the MLP can zero out dependence on the other dimension. We can extend this to higher dimension with a multiplicative increase in the size of the parameters.

3.4.5 Experimental details and additional results

3.4.5.1 Section 3.4.2: SGD finds the shortcuts, under ideal supervision

This section contains a full description and discussion of the in-distribution simulation experiments from Section 3.4.2.

3.4.5.1.1 Shallow Transformers simulate small groups and semigroups

The main experiments in this paper investigate whether gradient-based training of Transformers finds low-depth solutions to the problem of simulating semiautomata. In these experiments, we consider a wide variety of semiautomata $\mathcal{A}$, corresponding to various groups and semigroups, and construct a distribution $\mathcal{D}_{\mathcal{A}}$ over input sequences $(\sigma_1, \ldots \sigma_T)$ and their corresponding state sequences $(q_1, \ldots, q_T) = \mathcal{A}_{T,q_0}(\sigma_{1:T})$. In each setting, the σ_t are chosen uniformly at random from the set of valid tokens in Σ. [26] Given this distribution $\mathcal{D}_{\mathcal{A}}$, and a sequence-to-sequence neural network (with a token embedding and a linear classification head) which maps Σ^T to token predictions $Y \in \mathbb{R}^{T\times|Q|}$ (such that $Y_{t,q} := \widehat{\mathbf{Pr}}_\theta(q_t = q|\sigma_{1:t})$), we establish the task of minimizing the cross-entropy loss

$$L(\theta) := \frac{1}{T}\sum_{t=1}^{T}\log(1/Y_{t,q_t}).$$

[26]Take for instance the Dyck language, if the current stack is empty, then σ_t is chosen uniformly from the choices of open parentheses but not the closing parentheses. This is in accordance with Yao et al. [2021b].

This defines a supervised learning problem over sequences.

Note that without intermediate states in the input these problems exhibit *long-range dependencies*: for example, in the parity semiautomaton (and for any semiautomaton whose transformation semigroup is a group), every q_t depends on *every* preceding input $\{\sigma_{t'} : t' < t\}$. Indeed, this is why previous studies have used group operations as a benchmark for reasoning [Anil et al., 2022b, Zhang et al., 2022].

Settings. We proceed to enumerate the semiautomata considered in these simulation experiments.

- Cyclic groups $C_2, C_3, \dots, C_8$. For each cyclic group C_n (realized as $Q := \{0, 1, \dots, n-1\}$ under mod-n addition), we choose the generator set Σ to be the full set of group elements $\{0, \dots, n-1\}$. An alternative could be to let Σ be a minimal[27] set $\{0, 1\}$, which we do not use in the experiments.
- Direct products of cyclic groups $C_2 \times C_2, C_2 \times C_2 \times C_2$, realized as concatenated copies of the component semiautomata. Note that C_6 (which is isomorphic to $C_2 \times C_3$), included in the above set, is another example.
- Dihedral groups D_6, D_8. Our realization of D_{2n} chooses $Q = \{0, 1, \dots, n-1\} \times \{0, 1\}$ and $\Sigma = \{(1, 0), (0, 1)\}$. Since these groups are non-abelian, it is already not so straightforward (compared to parity) to see why constant-depth shortcuts should exist.
- Permutation groups A_4, S_4, A_5, S_5. We choose Q to be the set of $n!$ permutations for S_n (*symmetric group*), and Q to be the set of $\frac{n!}{2}$ even permutations for A_n (*alternating group* on n elements). The generator set for S_n consists of the minimal generators, a transposition and an n-cycle, as well as 6 other permutations. [28] For A_n, we choose the 3-cycles of the form $(12i)$ for $i \in \{3, 4, \cdots, n\}$. Note that A_4, S_4 are solvable (leading to constant-depth shortcuts), while A_5, S_5 are not. Also, note that to learn a constant-depth shortcut for A_4, a model needs to discover the wondrous fact that A_4 has a nontrivial normal subgroup, that of its double transpositions.
- The quaternion group Q_8. This is the smallest example of a non-abelian solvable group which is not realizable as a semidirect product of smaller groups, thus requiring the full wreath product construction (Lemma 19) in our theory.
- The Dyck language $\text{Dyck}_{n,k}$ (correctly nested brackets of k types, with depth at most n). We take $n = 4$, $k = 2$ in the experiments. To realize $\text{Dyck}_{n,k}$ as a semiautomaton simulation problem, the state Q is the state of the stack which implements Dyck language recognition (there are thus $\sum_{i=0}^{n} k^i$ distinct states); [29] Σ is the set of $2k$ opening and closing brackets. The distribution in inputs is slightly different, since there is a notion of "illegal" inputs: if the stack is empty, then the set of feasible inputs contain all the opening brackets; if the stack is full (i.e. reaching depth

[27]In the sense that it induces a non-trivial learning problem on this group.If we only pick the generator $\{1\}$, the output sequence is deterministic, and there is no learning problem.

[28]These other permutations are chosen following the ordering given by the sympy.combinatorics package. They are not necessary for covering the state space (since the minimal set of 2 permutations already suffice to cover Q), but can help speed up the mixing of the states.

[29]In the experiments we use $(k+1)^n$ classes (i.e. each of the n positions can take $(k+1)$ possible values), $\sum_{i=0}^{n} k^i$ of which are reachable.

n), then the only feasible input is the closing bracket for the opening bracket at the top of the stack.

- Gridworld semiautomata $\text{Grid}_4, \text{Grid}_9$, where $Q = \{0, 1, \cdots, n-1\}$ (for $n = 4$ or 9) and $\Sigma = \{\pm 1\}$.[30] For this special case, we have a constant-depth solution as stated in Theorem 12.

Training. We focus on the *online learning* setting for all experiments in this paper: at training iteration i, draw a fresh minibatch of samples from $\mathcal{D}_{\mathcal{A}}$, compute the network's loss and gradients on this minibatch, and update the model's weights using a standard first-order optimizer (we use AdamW [Loshchilov and Hutter, 2017a]). This is to mitigate the orthogonal challenge of overfitting; note that the purpose of these experiments is to determine *whether* standard gradient-based training finds shortcut solutions in these combinatorial settings (in a reasonable amount of time), not *how efficiently*. We do not investigate how to improve sample efficiency in this paper. The results in the paper are based on sinusoidal positional encodings [Vaswani et al., 2017] unless otherwise specified.

Sequence length. We report our main results with sequence length $T = 100$, which is large enough to rule out memorization: for this choice of T, the inputs come from a uniform distribution over $|\Sigma|^{100} > 10^{30}$ sequences, rendering it overwhelmingly unlikely for a sample to appear twice between training and evaluation. We observed positive results in most of the settings for larger T, but training became prohibitively unstable and computationally expensive; mitigating this is an interesting direction for future empirically-focused studies.

Depth. We seek to investigate the sufficient depth for learning to simulate each semiautomaton. Thus, for each problem setting, we vary the number of layers L in the Transformer between 1 and 16. Note that we do not attempt in this work to distinguish between depths $O(\log T)$ and $O(1)$, nor do we attempt to tackle the problem of exhaustively enumerating and characterizing the shortcut solutions for any particular semiautomaton.

Results. For each task and number of layers, we report the highest (Figure 3.15) and median (Figure 3.16) accuracies over 20 runs. The accuracy is calculated at token level (i.e. $\frac{1}{T}\sum_{t\in[T]} \mathbb{1}[\hat{q}_t = q_t]$), as opposed to the sequence-level accuracy (i.e. $\mathbb{1}[\hat{q}_{1:T} = q_{1:T}]$) as reported in Bhattamishra et al. [2020a]. We evaluate in-distribution accuracy on independent (unseen) samples of $\mathcal{D}_{\mathcal{A}}$, which contain 2048 sequences of length $T = 100$. [31] As shown in Figure 3.15, Transformers, trained with standard gradient-based methods, are able to find solutions which generalize well (in-distribution) on all of the tasks. Performance tends to improve as the number of layers increases (there is a small amount of non-monotonicity in some settings due to training instability); the sufficient depth to achieve high accuracy varies depending on the problem setting, as discussed below.

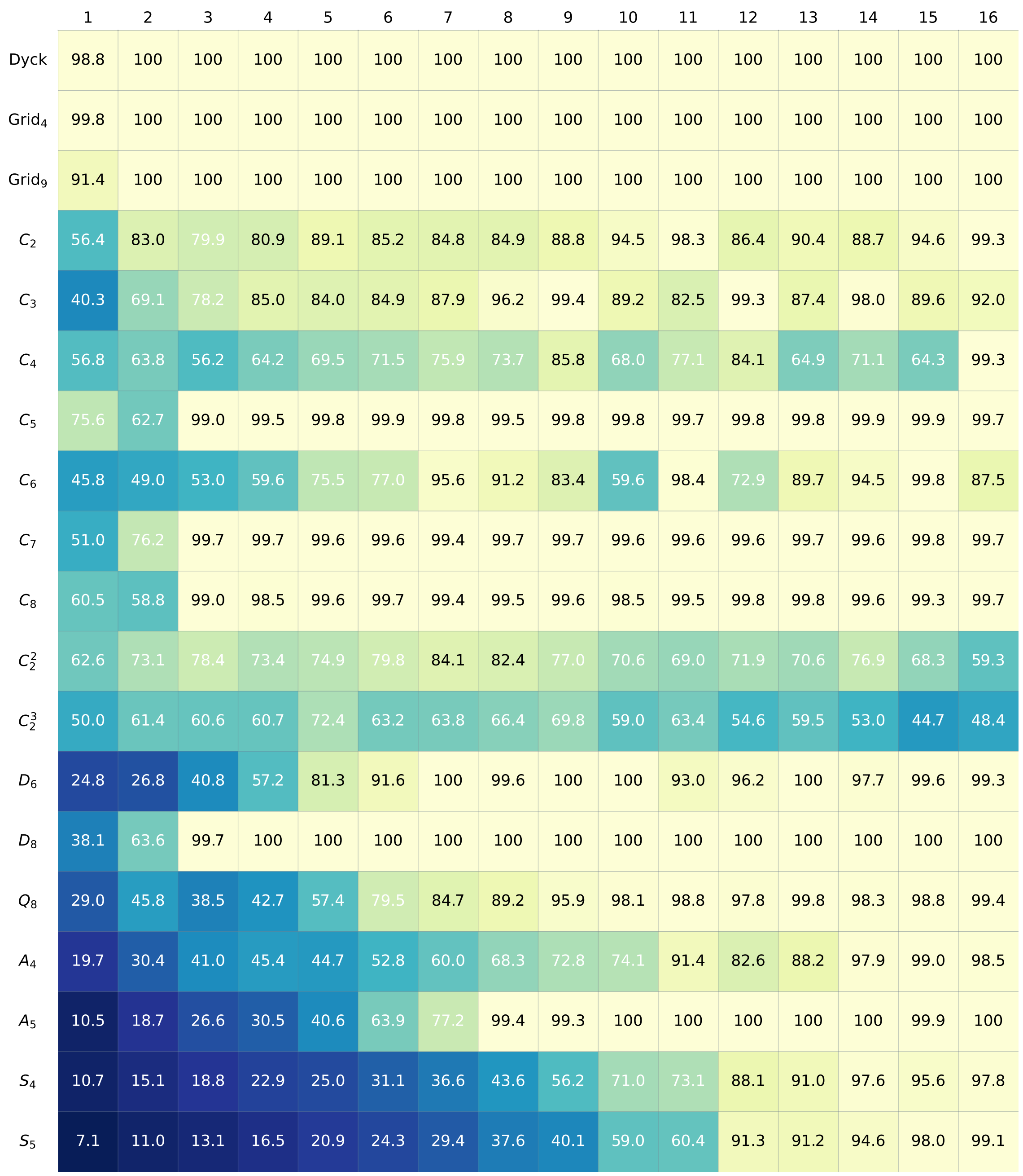
Trends in sufficient depth. The minimum number of layers required to achieve 99%+ performance reflects our beliefs on the difficulty of the task: a high-level trend is that the semigroups which don't

[30] -1 for L, 1 for R. We omit the no-op $\perp$ in the experiment which does not change the difficulty of the task.

[31] This size is sufficient for evaluating the model performance: for example, for C_2 (i.e. parity), evaluating a model on 10 evaluation sets of this size gives a standard deviation of 0.031% in the accuracy.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dyck	99.3	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Grid$_4$	99.9	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Grid$_9$	92.2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100	100	100	100	100	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	99.9	100	100	100	100	100	99.8	100
C_4	95.1	92.3	84.2	99.9	99.7	99.9	100	100	100	100	100	100	100	100	100	100
C_5	89.0	99.1	99.9	100	100	100	100	100	100	100	100	100	100	100	100	100
C_6	59.8	98.7	75.5	99.9	99.8	99.9	99.9	100	100	100	99.8	99.9	100	99.8	99.9	99.9
C_7	90.9	95.0	99.9	99.9	100	99.9	100	100	100	100	100	99.8	100	100	100	100
C_8	79.6	96.2	99.8	99.8	99.9	100	99.9	99.9	100	99.4	99.9	99.9	99.9	100	99.9	99.9
C_2^2	90.5	98.8	99.9	100	100	99.9	100	100	99.9	99.9	100	100	100	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	100	100	91.9	95.9	91.7	90.6	87.5	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100	100	100	100	100	100	100
A_4	25.0	35.4	49.1	59.3	62.6	82.3	90.9	98.0	98.0	99.1	99.8	100	99.7	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100	100	100	100	100	100	100
S_4	11.3	17.6	22.0	27.1	37.7	44.8	50.8	72.5	91.3	97.1	97.9	98.7	99.9	100	99.8	99.9
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	86.3	94.8	90.2	97.2	99.3	99.1	99.9	99.9

Figure 3.15: A complete version of Figure 3.5, for various tasks (rows) and numbers of network layers (columns). Reported performance is the *maximum* test accuracy over 20 runs.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dyck	98.8	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Grid$_4$	99.8	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Grid$_9$	91.4	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
C_2	56.4	83.0	79.9	80.9	89.1	85.2	84.8	84.9	88.8	94.5	98.3	86.4	90.4	88.7	94.6	99.3
C_3	40.3	69.1	78.2	85.0	84.0	84.9	87.9	96.2	99.4	89.2	82.5	99.3	87.4	98.0	89.6	92.0
C_4	56.8	63.8	56.2	64.2	69.5	71.5	75.9	73.7	85.8	68.0	77.1	84.1	64.9	71.1	64.3	99.3
C_5	75.6	62.7	99.0	99.5	99.8	99.9	99.8	99.5	99.8	99.8	99.7	99.8	99.8	99.9	99.9	99.7
C_6	45.8	49.0	53.0	59.6	75.5	77.0	95.6	91.2	83.4	59.6	98.4	72.9	89.7	94.5	99.8	87.5
C_7	51.0	76.2	99.7	99.7	99.6	99.6	99.4	99.7	99.7	99.6	99.6	99.6	99.7	99.6	99.8	99.7
C_8	60.5	58.8	99.0	98.5	99.6	99.7	99.4	99.5	99.6	98.5	99.5	99.8	99.8	99.6	99.3	99.7
C_2^2	62.6	73.1	78.4	73.4	74.9	79.8	84.1	82.4	77.0	70.6	69.0	71.9	70.6	76.9	68.3	59.3
C_2^3	50.0	61.4	60.6	60.7	72.4	63.2	63.8	66.4	69.8	59.0	63.4	54.6	59.5	53.0	44.7	48.4
D_6	24.8	26.8	40.8	57.2	81.3	91.6	100	99.6	100	100	93.0	96.2	100	97.7	99.6	99.3
D_8	38.1	63.6	99.7	100	100	100	100	100	100	100	100	100	100	100	100	100
Q_8	29.0	45.8	38.5	42.7	57.4	79.5	84.7	89.2	95.9	98.1	98.8	97.8	99.8	98.3	98.8	99.4
A_4	19.7	30.4	41.0	45.4	44.7	52.8	60.0	68.3	72.8	74.1	91.4	82.6	88.2	97.9	99.0	98.5
A_5	10.5	18.7	26.6	30.5	40.6	63.9	77.2	99.4	99.3	100	100	100	100	100	99.9	100
S_4	10.7	15.1	18.8	22.9	25.0	31.1	36.6	43.6	56.2	71.0	73.1	88.1	91.0	97.6	95.6	97.8
S_5	7.1	11.0	13.1	16.5	20.9	24.3	29.4	37.6	40.1	59.0	60.4	91.3	91.2	94.6	98.0	99.1

Figure 3.16: The *median* accuracy for various tasks (rows) and numbers of network layers (columns). Reported performance is the *median* test accuracy over 20 runs.

contain groups (which only require memory lookups) are the easiest to learn, and among the groups, the larger non-abelian groups require more layers to learn, with the non-solvable group S_5 requiring the largest depth.[32] Between the non-abelian groups, the difficulty of learning Q_8 compared to D_8 (which has the same cardinality) agrees with our theoretical characterizations of the respective constant-depth shortcuts for these groups: D_8 can be written as a semidirect product of smaller groups, while Q_8 cannot, so our theoretical construction of a constant-depth shortcut must embed Q_8 in a larger structure (i.e. the wreath product).

Improving training stability. Throughout these experiments, we observe the following forms of training instability: high variance in training curves (based on initialization and random seeds for the gradient-based optimization algorithm), and negative progress (i.e. non-monotonic loss curves), even for training runs which eventually converge successfully. This is evident in Figure 3.5(b),(c) and in the significant difference between the maximum accuracies in Figure 3.15 and the median in Figure 3.16.

To stabilize training, we experiment with dropout and exponential moving average (EMA)[33]. The effectiveness of dropout varies across datasets; for example, we find using a dropout of 0.1 (the best among $\{0, 0.1, 0.2, 0.3\}$) to be helpful for Dihedral and Quaternion, while such dropout hurts the training of Dyck and Gridworld. We find EMA to be generally useful, and fix the decay parameter $\gamma = 0.9$ in the experiments since the performance of the EMA model does not seem to be sensitive to the choice of $\gamma \in \{0.85, 0.9, 0.95\}$. Further, increasing the patience of the learning rate scheduler can be helpful.

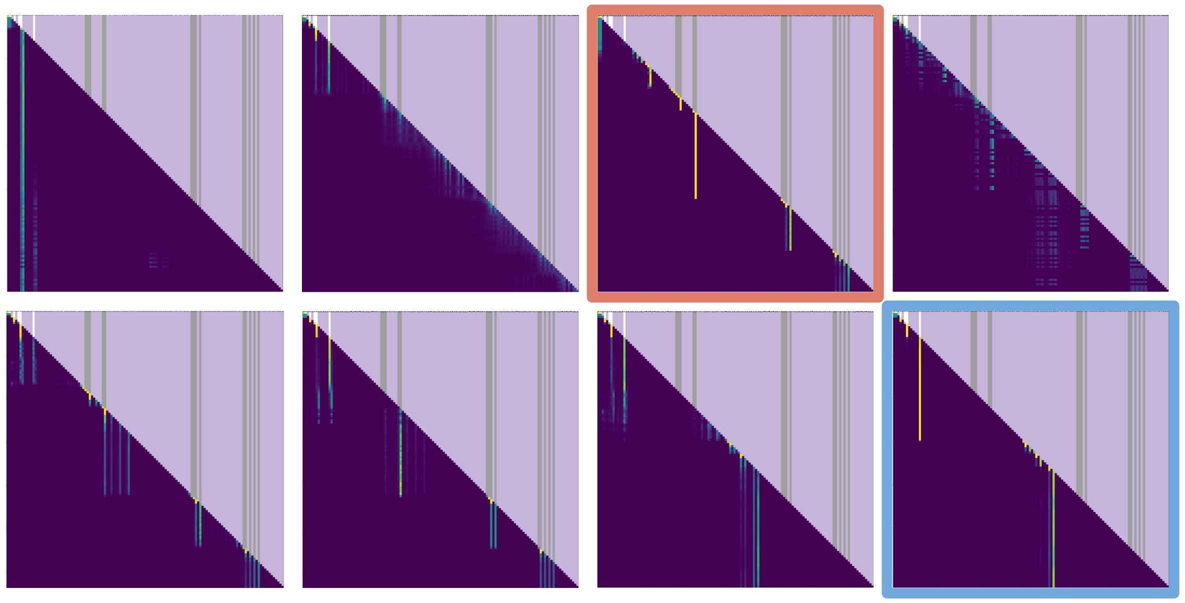
3.4.5.1.2 Visualizing and interpreting attention heads

Although we defer a fine-grained mechanistic interpretability study (*"which group/semigroup factorizations did these shallow Transformers discover, if any?"*) to future work, we provide some preliminary visualizations of attention heatmaps which strongly corroborate their theoretical counterparts. In particular, consider the gridworld setup in Theorem 12. The theoretical construction consists of two steps: the first attention layer calculates the prefix sum of the actions (i.e. the sum of $\{\tilde{\sigma}_i\}_{i\in[T]} \in \{0, \pm 1\}^T$), and the second attention layer identifies the last time the process is at a boundary state (i.e. 0 or S) where the process can be "reset" (i.e. the model can ignore the history before the boundary state and only needs to calculate the sum of subsequent actions).

We have seen in Figure 3.6 that the network indeed learns to 1) compute the prefix sum, as evidenced by the uniform attention in the first layer, and 2) detect boundary states, as highlighted by large attention scores in the last layer. Figure 3.17 provides more examples of attention patterns, which are taken from the last layer of a 4-layer GPT-2 model on two randomly selected Grid_9 sequences. We highlight the locations where the process is at a boundary state (white strips for state 0 or gray strips for state $S = 8$), which align well with the highly activated positions of the attention heads, showing that the model learns to locate the closest boundary states. Moreover, when processing

[32]However, we stress that these experiments do *not* control for the fact that larger groups have richer supervision (for example, A_5 has more informative labels than A_4), possibly accounting for the counterintuitive result that the latter requires more layers, despite being a subgroup of the former.

[33]We use the EMA implementation from https://github.com/fadel/pytorch_ema.


(a)

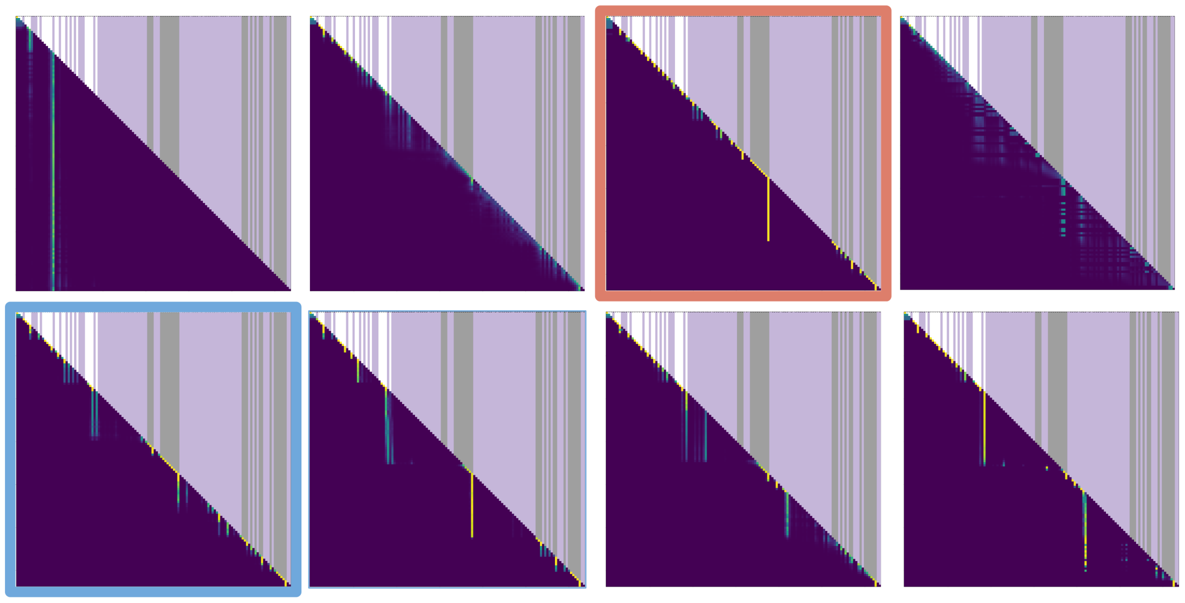

(b)

Figure 3.17: Visualization of the entire set of 8 attention heads on two randomly selected length-128 sequences for models trained on Grid$_9$. The lower triangles visualize the attention patterns, while the upper triangles are ignored by the attention head because of the causal mask. We use the upper triangles to visualize the positions of state 0 and state $S = 8$ in the output: *white strips* mark the position of state 0 and *gray strips* mark state $S = 8$. Example heads that clearly detect state 0 and S are highlighted with blue and red frames, respectively. Note that in many cases, the white/gray strips align with the locations of high attention scores (the bright yellow patterns). This suggests that the model indeed learns to identify the boundary states and that the construction in Theorem 12 agrees with solutions found in practice.

tokens appearing later in the sequence than these highly activated positions, no attention weight is put on tokens *before* these positions. This suggests that these highly activated locations reset the state so that the model does not need to look further back past them.

3.4.5.2 Section 3.4.3: Failures of shortcuts in more challenging settings

Our theoretical and main empirical findings have shown that not only do shallow non-recurrent networks subsume deeper finite-state recurrent models in theory, these shallow solutions can also be found empirically via standard gradient-based training. However, experiments in Section 3.4.2 and Section 3.4.5 are in an idealized setting, with full state supervision during training and in-distribution evaluation at test time. This section studies more challenging settings where these assumptions are relaxed. We consider training under indirect (Section 3.4.5.2.1) or incomplete (Section 3.4.5.2.2) state supervision, and evaluation on sequences that is out-of-distribution (Section 3.4.5.2.3) or of longer lengths (Section 3.4.5.2.4)

3.4.5.2.1 Challenges from indirect supervision

One type of limited supervision is that the observations may not provide full information of the underlying state. To model this, we consider the case where instead of observing the state q directly, we get a function of the state, denoted $\varphi(q)$, where $\varphi : Q \to \tilde{Q}$ is non-injective (i.e. $|\tilde{Q}| < |Q|$). In each of the experiments involving partially-observable semiautomata, we specify the underlying semiautomaton, as well as the observation function φ.

- *Dyck language with stack top observations:* For $\text{Dyck}_{n,k}$, the state Q is the state of the stack which takes $\sum_{i=0}^{n} k^i$ values. We take φ to be the function that takes in a stack and returns the element at the top of the stack, which is either one of the k open brackets if the stack if non-empty, or a special token $\perp$ indicating an empty stack, i.e. $\tilde{Q} := \{1, 2, \cdots, k, \perp\}$. We consider $k = 8$ (as opposed to $k = 2$ in Section 3.4.2) to make the prediction task more challenging.
- *Gridworld with boundary observations:* We consider the case where the underlying semiautomaton is Grid_9 with $Q = \{0, 1, \cdots, 8\}$. The observation function $\varphi : Q \to \{0, 1\}$ outputs whether the current state is of two boundary states, i.e. at state 0 or state $S = 8$.
- *Permutations with single-element observations:* We take the permutation group S_5 with Q is the set of 5! operations. The observation function $\varphi : Q \to \{1, 2, 3, 4, 5\}$ returns the first value of the permutation. For example, $\varphi((2, 1, 4, 3, 5) = 2$. We use a set of 5 generators for the experiments.
- *Cyclic group with "0 mod 4" observations:* We take C_4 as the underlying group with $Q = \{0, 1, 2, 3\}$. The observation function computes whether the current state is state 0, i.e. $\varphi(q) = \mathbb{1}[q = 0]$.
- *Dihedral group with rotation component only:* Recall that $D_{2n} = C_n \rtimes C_2$. We take $n = 4$ with $Q = \{0, 1, 2, 3\} \times \{0, 1\}$, and let the observation function ψ output only the the first component (i.e. $\tilde{Q} = \{0, 1, 2, 3\}$).
- *`(abab)*`:* We consider one semiautomaton which is not featured in Section 3.4.2: the one which recognizes the regular expression `(abab)*`, which is also studied in Bhattamishra et al. [2020a].

Task	Dyck$_{4,8}$	Grid$_9$	S_5	C_4	D_8	$(\texttt{abab})^\star$, (1)	$(\texttt{abab})^\star$, (2)
Observation	stack top	$\mathbb{1}_{\text{boundary}}$	$\pi_{1:t}(1)$	$\mathbb{1}_{0 \bmod 4}$	location	accept	accept
Accuracy	100.0	100.0	99.6	99.9	100.0	100.0	100.0

Figure 3.18: Accuracies with indirect supervision, extending results in Figure 3.7(a). The numbers are the maximum over 25 runs. As a reference, LSTM gets 100% on all tasks.

The underlying semiautomaton has 5 states: 4 states are in a cyclic fashion when seeing repeated patterns of *abab*, and a fifth absorbing "failure" state is entered if any other pattern is seen. For example, the input sequence *abababaaabab* corresponds to states 012301244444, where the 5^{th} "*a*" leads to the absorbing state. The observation function φ computes whether the current state is the "accepting" state (i.e. state 3), with $\tilde{Q} = \{0, 1\}$. For example, the output of φ for the input sequence *ababababa* is 000100010, and the output for the input sequence *ababbabab* is 000111111, i.e. the sequence enters the absorbing state at position 5 and never recovers.

We consider two distributions on the input sequences: (1) the input is always a sequence of the form *abababa* $\cdots$ (i.e. the process is never in the absorbing state), which is the setup in Bhattamishra et al. [2020a]; and (2) the input is of the form *abababa* $\cdots$ with probability 0.5, and is some randomly drawn string of a, b otherwise. Note that case (1) can be solved purely based on the positional encoding, since the label is 1 when the position is a multiple of 4 and 0 otherwise, while case (2) is more difficult since the model needs to take into account the input tokens.

Results. We train GPT-2-like models on sequences of length 40. We use 16 layers for S_5 and 8 layers for other tasks, with embedding dimension $d = 512$ and $H = 8$ attention heads. As shown in Figure 3.18, the model is able to achieve near-perfect in-distribution accuracies for all tasks. An interesting side finding is that the choice of positional encoding turns out to be important for both cases of $(abab)^*$: learning is challenging for linear encoding (i.e. $p_i \propto i$) but is easy when using sinusoidal positional encoding, which is likely because the sinusoidal encoding naturally matches the periodicity in $(abab)^*$. In all other experiments, we use sinusoidal positional encodings unless otherwise noted.

3.4.5.2.2 Challenges from incomplete supervision

Another challenge of limited supervision is that the observation sequence may be incomplete, that is, we may not be able to get supervision on the states at every time step. We consider the task of learning length 100 sequences, where the state at each position is revealed with some probability $p_{\text{reveal}} \in (0, 1]$.

Results. Figure 3.19 shows the accuracy against p_{reveal}, for S_5 and C_2 (i.e. parity). Transformer training pipeline is worse than LSTM at tolerating incomplete supervision: while Transformer is able to maintain the performance across p_{reveal} for C_2, the performance degrades significantly at lower p_{reveal} for S_5. We leave improving the robustness to sparse supervision to future work.

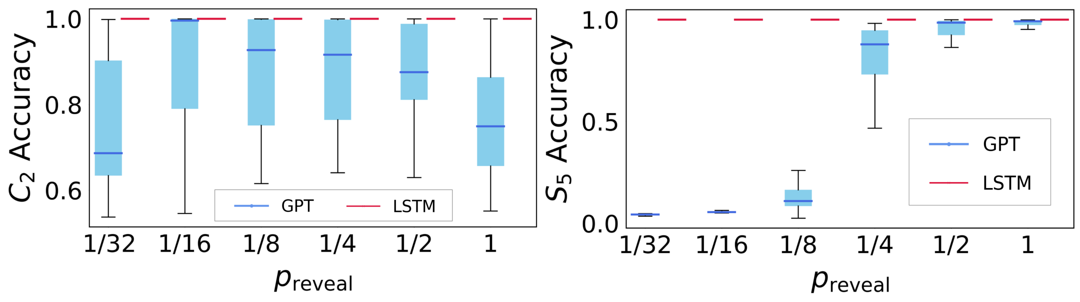


Figure 3.19: Learning from incomplete state sequences, extending results from Figure 3.7(b): accuracy vs. position-wise probability of a hidden token (i.e. p_{reveal}), for GPT and LSTM. While LSTM is able to maintain a perfect accuracy across different values of p_{reveal}, GPT's performance may degrade as labels get sparser. The mean and standard deviation are taken over 25 runs.

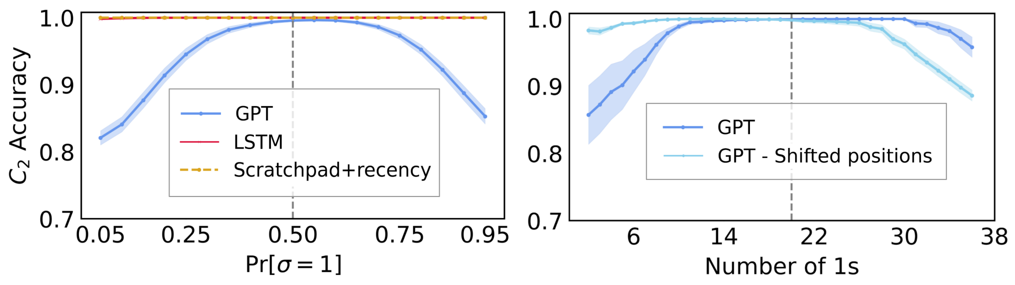


Figure 3.20: OOD generalization performance on C_2: (*Left*) Accuracy on sequences of the same length as training, with varying $Pr[\sigma = 1] = 0.5$. GPT fails at OOD generalization, whereas recurrent solutions implemented by LSTM and Scratchpad with recency bias is robust to different distributions. (*Right*) Accuracy on sequences of the same length as training, with a varying number of 1s in each sequence. GPT has worse performance on counts less frequently seen during training. The lines show the mean accuracy (with shadows showing standard error) over 25 replicates.

3.4.5.2.3 Out-of-distribution generalization

The previous subsections show positive results on learning shallow non-recurrent shortcuts with limited supervision during training, either in the form of indirect observations or incomplete observation sequences. In this section, we study challenges at test time, and evaluate Transformers on their *out-of-distribution* generalization performance. For this and the next subsection, the models are trained in the standard way with full state supervision. The training sequences are of length 40, where each position has an equal probability of being 0 or 1, i.e. $\mathbf{Pr}[\sigma = 1] = 0.5$. At test time, the sequences of the same length as training, but the Bernoulli parameter $\mathbf{Pr}[\sigma = 1]$ varies in the range $\{0.05, 0.1, 0.15, \ldots, 0.9, 0.95\}$.

Negative results for vanilla Transformers. Figure 3.20 (*left*) shows the accuracy as $\mathbf{Pr}[\sigma = 1]$ varies. The performance of the Transformer degrades sharply as the test distribution changes away from training, failing at out-of-distribution generalization. Given the theoretical construction of modular counters (Lemma 15), our hypothesis is that Transformer may be learning a shortcut solution that computes the parity by counting the number of 1s, and that counts less frequently seen during training will cause the model to fail. The experimental results agree with the hypothesis: as $\mathbf{Pr}[\sigma = 1]$ deviates from 0.5, it is less likely for the value of the count (which concentrates around $T \times \mathbf{Pr}[\sigma = 1]$) to be seen during training, hence the performance degrades. In contrast, an LSTM recurrent network maintains perfect accuracy when evaluated on all values of $\mathbf{Pr}[\sigma = 1]$.

We further test this hypothesis by checking how the accuracy changes as we vary the count (i.e. the number of 1s) in the input sequence. As shown in Figure 3.20 *(right)*, Transformer's performance degrades as the count moves away from the expected number during training, agreeing with the hypothesis. It might appear strange that GPT fails at a lower count more than a higher count. However, this may be because the shortcut learns a correlation between the count and the position: during training, a lower count is more likely to appear early in an input sequence, as opposed to the testing scenario where a lower count is equally likely to appear at a later part of an sequence. This is further supported by the observation that training the model with randomly shifted positions significantly improves the performance at lower counts.

Guiding the Transformer to learn the recurrent solution. We investigate one established mitigation for the out-of-distribution brittleness of non-recurrent Transformers: *scratchpad* training and inference. Given a sequence of inputs $(\sigma_1, \ldots, \sigma_T)$ and states $(q_1, \ldots, q_T)$, in the standard (non-recurrent) sequence-to-sequence learning pipeline, the network receives $\sigma_{1:T}$ as input, and outputs the sequence of predictions for q_t. In scratchpad training [Nye et al., 2021a, Wei et al., 2022c], we instead feed the network an interleaved sequence of inputs and states $(\sigma_1, q_1, \sigma_2, q_2, \sigma_3, q_3, \ldots, q_{T-1}, \sigma_T)$ (with an appropriately expanded token vocabulary), and define the network's state predictions to be those at the appropriately aligned positions: $(\hat{q}_1, \perp, \hat{q}_2, \perp, \ldots, \perp, \hat{q}_T)$ (where $\perp$ denotes a position where the prediction is ignored by the loss function). During inference, we iteratively fill in the state predictions. This removes the need for the network to learn long-range dependencies in a single non-recurrent pass, by splitting it into T sequential state prediction problems which can depend on previous predicted state $\hat{q}_{t-1}$; one can think of this as a way to guide a shallow Transformer to learn the recurrent solution (i.e. explicit depth-$\Theta(T)$ iteration of the state transition function), rather than a shortcut.

We note that introducing the scratchpad itself is not sufficient to remove the parallel solution, since the model can simply ignore the scratchpad positions and find the same parallel shortcut as before. The good news is that we can couple scratchpad with an explicit *recency bias* in the attention mechanism [Press et al., 2022] which biases the model towards putting more attention weights on closer input. Intuitively, if the model is only allowed to put attention on the current input token and the current scratchpad (which is simply the current state), then the model is forced to be recurrent; recency bias can be considered as a soft relaxation of the same idea. Combining scratchpad and recency bias, we are able to train a Transformer to learn the recurrent solution, which is resilient to distribution shift; see Figure 3.20 *(left)*. Notice that this mitigation completely foregoes the computational advantage of a shallow shortcut; we leave it to future work to obtain shortcuts which are resilient to distribution shift. Towards this, the constructions used in the proof of Theorem 10 may be helpful. Finally as a side note, even though the state transitions are Markov, the dependency in the input sequence can still be long range, so we do not expect recency bias to help without scratchpad, since in this case the output can depend uniformly on each input positions (e.g. consider parity).

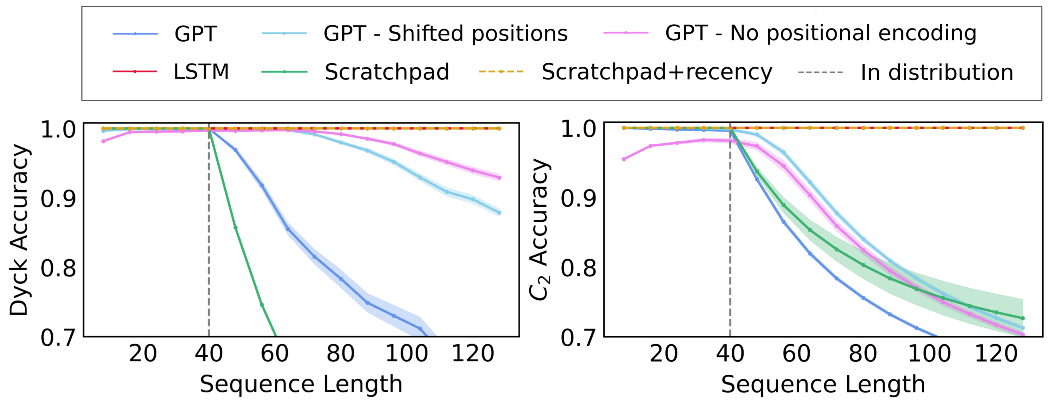


Figure 3.21: Length generalization on Dyck and C_2 (Figure 3.8(b), reproduced here for convenience): Transformer fails at length generalization, but adding Scratchpad [Nye et al., 2021a] and recency bias [Press et al., 2022] serves as a remedy. The lines show the mean accuracy (with shadows showing standard error) over 25 (± 1) replicates.

3.4.5.2.4 Length generalization

Settings for length generalization. Our final setup is length generalization, where the model is evaluated on sequences of lengths unseen during training. Promoting this difficult desideratum of *length generalization* is an intricate problem in its own right; see Yao et al. [2021b], Anil et al. [2022b] for more experiments similar to ours and more discussions on length generalization in 3.7. In the following, we check the length generalization performance on Dyck$_{4,2}$ and C_2 (with $Pr[\sigma = 1] = 0.5$), where the model is trained on sequences of length 40 and tested on sequences of length $\{8, 16, 24, \cdots, 120, 128\}$.

Results. Figure 3.21 shows the performance on sequences of various lengths. In contrast to LSTM's perfect performance on all scenarios, Transformer's accuracy drops sharply as we move to lengths unseen during training. This is not purely due to unseen values of the positional encoding: randomly shifting the positions during training can cover all the positions seen during testing, which helps improve the length generalization performance but cannot make it perfect; we see similar results for removing positional encodings altogether. However, similar to the OOD setup in the previous subsection, we empirically show that the above flaws are circumventable. Using a combination of *scratchpad* (a.k.a. "chain-of-thought") [Nye et al., 2021a, Wei et al., 2022c] and recency bias [Press et al., 2022], we demonstrate that Transformers can be guided towards learning recurrent (depth-T) solutions, which generalize out-of-distribution and to longer sequence lengths (Figure 3.21, yellow curves). The results also confirm that the inclusion of recency bias is necessary: without it, scratchpad training shows no improvement on length generalization.

Impact of positional encoding. Figure 3.21 also shows some interesting findings related to positional encoding, which is believed to be a key component for Transformers and a topic with active research [Ke et al., 2020, Chu et al., 2021]. While this work does not aim to improve positional encoding, some of our results may be of interest for future research.

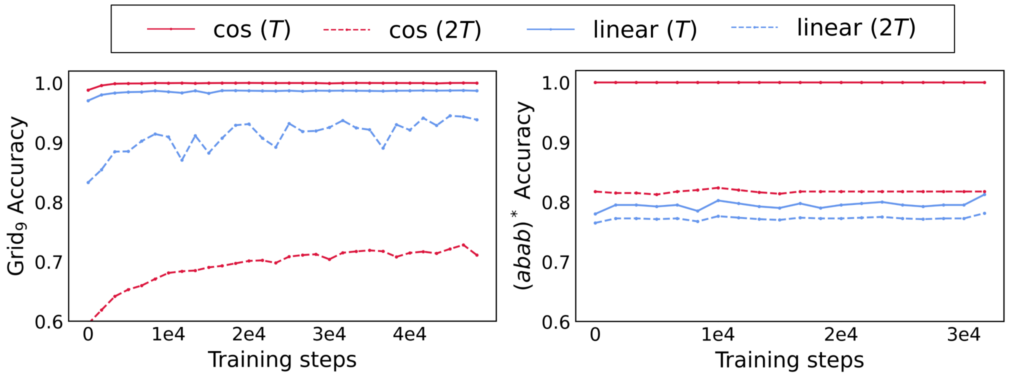


Figure 3.22: Choice of the positional encoding: while having similar or even superior in-distribution performance (on sequences of length $T = 40$), sinusoidal positional encoding may suffer a larger generalization gap than linear positional encoding when testing on length $2T$. The lines show the mean accuracy over 25 replicates.

Sinusoidal vs linear encoding: We find that the conventional sinusoidal encoding (which is the default for results in this paper) seems to generalize worse to unseen length than linear encoding (where $p_i \propto i$), despite having comparable or better in-distribution performance. Figure 3.22 shows examples on Grid$_9$ and partially observed $(abab)^*$, where we compare the accuracy on freshly drawn samples of the same length as the training sequences (i.e. in-distribution), or of twice the training length. For Grid$_9$, both positional encodings achieve comparable accuracy, however the sinusoidal encoding performs significantly worse when tested on sequences of doubled length. For partially observed $(abab)^*$ where the label is whether the current string is a multiple of *abab*, the sinusoidal encoding has a clear advantage over the linear encoding on in-distribution performance. However, when tested on sequences of lengths twice as those during training, the performance gap between the two positional encodings shrinks significantly.

Training with shifted positions: In general, unseen positions appear to be a major contributor to Transformer's failure of length generalization. This is evidenced by the comparison between Transformer trained with absolute positional encoding, and Transformers trained with random shifts added to the positional encoding: for each batch, we sample a random positive integer in [0,400] and add it to the position indices before calculating the positional encoding; this random integer is the same for each batch and varies across batches. Figure 3.21 shows that adding such random shifts gives a significant boost to Transformer's length generalization performance, for both Dyck and C_2. This suggests that a main challenge to length generalization is the distribution shifts due to positions unseen during training, and finding better positional encoding could be a potential remedy for poor length generalization.

As a side note, we also find that *removing positional encoding altogether* helps improve generalization for both parity and Dyck. For the former, removing positional encodings makes sense since parity is a symmetric function where the ordering of the arguments does not matter, [34] though the positive result for Dyck is less clearly understood. Note that removing positional encoding does not mean having

[34]Empirically, we are able to achieve non-trivial accuracy (even when evaluated at the sequence level) without positional encoding, whereas Bhattamishra et al. [2020a] reports 0 accuracy. The discrepancy may be due to different model size: Bhattamishra et al. considers Transformers with up to 4 layers, 4 heads and dimension up to 32, whereas for the parity experiments we consider Transformers with 8 layers, 8 heads, and dimension 512.

no position information, since the use of the causal mask implicitly encodes the position, which is also noted in Bhattamishra et al. [2020a] and concurrent work by Haviv et al. [2022]. Understanding this phenomenon is tangential to the current work and is left to future work.

3.4.5.3 Additional details

Hyperparameters. For GPT-2 models, we fix the embedding dimension and MLP width to 512 and the number of heads to 8 in all experiments in Section 3.4.2, and vary the number of layers from 1 to 16. For LSTM, we fix the embedding dimension to 64, the hidden dimension to 128, and the number of layers to 1. We use the AdamW optimizer [Loshchilov and Hutter, 2017a], with learning rate in {3e-5, 1e-4, 3e-4} for GPT-2 or {1e-3, 3e-3} for LSTM, weight decay 1e-4 for GPT-2 or 1e-9 for LSTM, and batch size 16 for GPT-2 or 64 for LSTM. As detailed in Section 3.4.5.1.1, the models are trained in an online fashion with freshly drawn samples in each batch. The number of freshly drawn samples ranges from 600k to 5000k for different datasets, which is much fewer than the number of possible strings of length 100.

Implementation details. Our experiments are implemented with PyTorch [Paszke et al., 2019]. The Transformers architectures are taken from the HuggingFace Transformers library [Wolf et al., 2019], using the GPT-2 configuration as a base. The LSTM architecture is the default one provided by the PyTorch library.

Computational resources. The experiments were performed on an internal cluster with NVIDIA Tesla P40, P100, V100, and A100 GPUs. For the experiments in Section 3.4.2, each training run took up to 10 hours on a single GPU, for a total of $\approx 10^4$ GPU hours. The remaining experiments in Section 3.4.3 amount to less than 1% of this expenditure.

3.5 Application: exposing *attention glitches* with flipflops

In the previous section, we have shown that while Transformers are representationally capable of implementing shortcuts to simulating automata, these shortcuts are not guaranteed to be found in practice. A natural question is then: what can these theoretical constructions tell us about the practical behavior of Transformers? In this section, we will show that these theoretical understanding provides primitives for probing the fine-grained extrapolative behavior of Transformers. Our investigation is centered at *flip-flop* (aka. one-bit memory unit; recall Example 2), which forms a fundamental building block of algorithmic reasoning according to the celebrated Krohn-Rhodes theorem Krohn and Rhodes [1965]. We will show how a simple task based on flip-flop, which we call *Flip-Flop Language Modeling* (FFLM), can help reveal intrinsic limitations of the attention mechanism.

3.5.1 Setup: Flip-flops and FFLM

3.5.1.1 A reminder on flip-flop

Recall from Example 2 that the *flip-flop automaton* is a two-state machine which remembers a single bit of memory and enables retrieval of this bit, as illustrated in Figure 3.23(a). Here we redefine it for easy reference:

Definition 11 (Flip-flop automaton)**.** *A flip-flop automaton $\mathcal{A} = \{Q, \Sigma, \delta\}$ is defined with state space $Q = \{0, 1\}$, input alphabet $\Sigma = \{\sigma_0, \sigma_1, \bot\}$, and transition function $\delta : Q \times \Sigma \to Q$ where*

$$\begin{cases} \delta(q, \sigma_0) & = 0, \\ \delta(q, \sigma_1) & = 1, \\ \delta(q, \bot) & = q; \end{cases} \qquad \forall q \in \{0, 1\}.$$

The semantics of the input symbols can be intuitively be identified with "write 0", "write 1", and "do nothing". This mathematical object is named after a type of electronic circuit which can store a single bit of state information [Eccles and Jordan, 1918, 1919]; such physical constructions appear ubiquitously in electrical engineering as the building blocks of memory.

Naturally associated with the flip-flop automaton is its *transformation monoid*, the closure[35] of its *state transformations* $\delta(\,\cdot\,, \sigma) : Q \to Q$ under function composition. Identifying each symbol with its state transformation map, we can compute the multiplication table of this monoid ($f \circ g$ for every pair of transformations f, g):

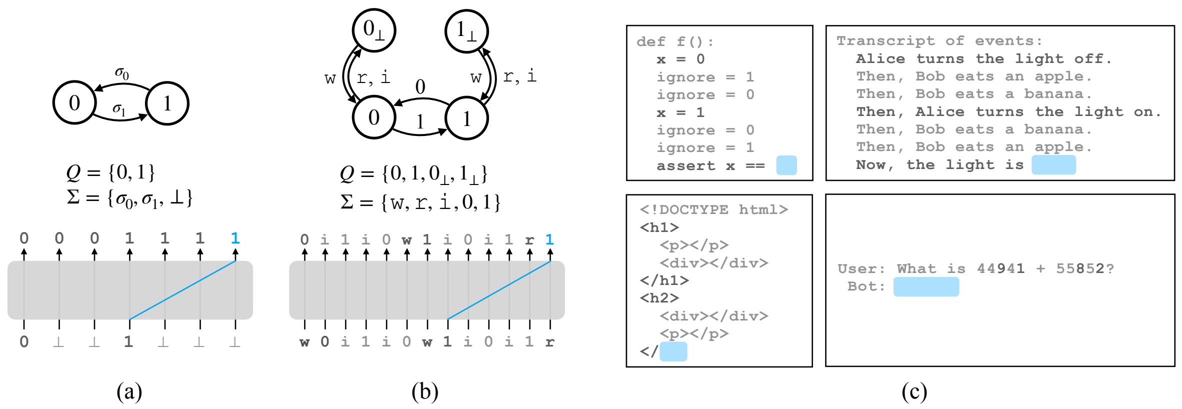
	$g = \sigma_0$	$g = \sigma_1$	$g = \bot$
$f = \sigma_0$	σ_0	σ_0	σ_0
$f = \sigma_1$	σ_1	σ_1	σ_1
$f = \bot$	σ_0	σ_1	$\bot$

This algebraic object is called the *flip-flop monoid* $\mathcal{F}$. Its binary operation $\circ$ is clearly non-invertible (intuitively: the history of the bit cannot be recovered after a "memory write") and non-commutative (the order of "write" operations matters); it also has an identity element $\bot$ (which does nothing to the memory bit). By enumeration of smaller objects, it can be seen that $\mathcal{F}$ is the smallest monoid (in terms of order $|\mathcal{F}|$, or fewest number of automaton states $|Q|$) which has these properties.

3.5.1.2 Flip-Flop Language Modeling (FFLM)

For any even number $T \geq 4$, we define a flip-flop string as a sequence of symbols $\{\texttt{w}, \texttt{r}, \texttt{i}, 0, 1\}^T$, which have the semantics of *instructions* (`write`, `read`, `ignore`) and *data* (one bit). A valid flip-flop string consists of alternating pairs of instructions and data (e.g. "`w 0 i 1 i 0 r 0`"), for which every symbol following a `r` instruction must be equal to the symbol following the most recent `w`; thus, "`w 0 i 1 w 1 r 0`" is not a legal flip-flop string. These sequences can be viewed as correct execution

[35]In this case, the closure is the same as the generator set: no functions distinct from $\sigma_0, \sigma_1, \bot$ can be obtained by composing these three functions. This is not true for a general automaton.



σ_0
σ_1
$Q = \{0, 1\}$
$\Sigma = \{\sigma_0, \sigma_1, \perp\}$
0 0 0 1 1 1 1
0 ⊥ ⊥ 1 ⊥ ⊥ ⊥
(a)
$0_\perp$
$1_\perp$
w
r, i
0
1
$Q = \{0, 1, 0_\perp, 1_\perp\}$
$\Sigma = \{$w, r, i, 0, 1$\}$
0 i 1 i 0 w 1 i 0 i 1 r 1
w 0 i 1 i 0 w 1 i 0 i 1 r
(b)
(c)

```
def f():
  x = 0
  ignore = 1
  ignore = 0
  x = 1
  ignore = 0
  ignore = 1
  assert x ==
```

```
Transcript of events:
  Alice turns the light off.
  Then, Bob eats an apple.
  Then, Bob eats a banana.
  Then, Alice turns the light on.
  Then, Bob eats a banana.
  Then, Bob eats an apple.
  Now, the light is
```

```
<!DOCTYPE html>
<h1>
  <p></p>
  <div></div>
</h1>
<h2>
  <div></div>
  <p></p>
</
```

```
User: What is 44941 + 55852?
 Bot:
```

Figure 3.23: Elementary objects and examples associated with flip-flop languages. (a) the 2-state flip-flop machine (elided transitions are self-loops). (2) A 4-state automaton which processes flip-flop languages (implying the existence of a small RNN). (c) Simple examples of sequential prediction tasks which require processing a flip-flop language.

transcripts of a program which can (perhaps occasionally) `write` to a single bit of memory, and always correctly `read`s its contents. We also require that all sequences begin with `w`.

There are many possible choices of (probabilistic) *flip-flop languages*, which are distributions over valid flip-flop strings. We define a canonical family of them: let $\mathsf{FFL}(T, \mathbf{p})$ be the distribution over length-T flip-flop strings, parameterized by $\mathbf{p} = (p_{\mathtt{w}}, p_{\mathtt{r}}, p_{\mathtt{i}}) \in \Delta(\{\mathtt{w}, \mathtt{r}, \mathtt{i}\})$, such that:

(i) The first instruction x_1 is always `w`, and the last instruction x_{T-1} is always `r`.

(ii) The other instructions are drawn i.i.d. according to $(p_{\mathtt{w}}, p_{\mathtt{r}}, p_{\mathtt{i}})$ with $p_{\mathtt{i}} = 1 - p_{\mathtt{w}} - p_{\mathtt{r}}$.

(iii) The nondeterministic data symbols (paired with `w` or `i`) are drawn i.i.d. and uniformly.

We are interested in whether language models can learn a flip-flop language from samples, which we define as processing the `read` operations *perfectly*. Two variants of the autoregressive language modeling task can be defined on this distribution:

- **Generative ("noisy") mode:** Estimate the conditional next-token distribution $\mathbf{Pr}[x_{t+1}|x_{1:t}]$, for each $t = 1, \ldots, T-1$. In this mode, the sequences can be treated as drop-in replacements for natural text in GPT-style training. Generative FFLMs can be evaluated by checking their completions on prefix "prompts" (e.g. "... `w 0 i 1 i 1 r [?]`").

- **Deterministic ("clean") mode:** Predict only the continuations which are deterministic: correctly output x_{t+1} only for the prefixes $x_{1:t}$ such that $x_t = \mathtt{r}$. At the cost of a slight departure from vanilla language modeling, this setting isolates the long-range memory task. It is similar to the non-autoregressive flip-flop monoid simulation problem discussed in Section 3.5 (Liu et al. [2023a]) with limited supervision. [36]

[36]We observe similar behaviors across these two settings (see Appendix 3.5.6.2), but we report results on the "clean" setting in this paper. Predicting the non-deterministic tokens is irrelevant to the memory task at hand.

These tasks naturally embed the capability of simulating the *flip-flop*, a machine which memorizes a single bit (see Figure 3.23a,b for closely related variants).[37] It is easy to see that recurrent networks and 2-layer Transformers (see Proposition 3) *can* both represent FFLM parsers. The question of whether they *do*, especially from less-than-ideal data, turns out to be extremely subtle, and is the subject of the remainder of this paper.

3.5.1.2.1 Why focus on the flip-flop?

The most immediate rationale for this synthetic benchmark is that flip-flop simulation (maintaining memory in a sequence) is a direct necessity in many reasoning settings (see Figure 3.23c). It is a special (depth-1) case of Dyck language processing [Chomsky and Schützenberger, 1959, Yao et al., 2021a, Zhao et al., 2023], which is necessary for parsing recursive grammars. It also captures certain structures in code or language tasks, such as ignoring irrelevant contexts [Shi et al., 2023] or tracking semantics changes [Zhang et al., 2023]. Thus, more than a toy model, flip-flop languages are embedded verbatim within many sequence processing tasks. We offer some additional perspectives below.

Algebraic properties and expressive power. Flip-flops are the computational building blocks of memory. The *flip-flop monoid* $\mathcal{F}$ (Example 2), an algebraic encoding of a flip-flop's dynamics, is the smallest monoid whose operation is both *non-commutative* and *non-invertible*. $\mathcal{F}$ plays an essential role in the Krohn-Rhodes theory of automata and semigroups [Rhodes et al., 2010], whose central structure theorem [Krohn and Rhodes, 1965, Zeiger, 1967, Eilenberg, 1974] implies that a constant-depth cascade of parallel flip-flops simulates *all* group-free finite-state automata. Thus, in a rigorous sense, the robust learning of flip-flops is not only a *necessary* condition for reasoning, but a *sufficient* condition for a wide class of algorithmic capabilities.

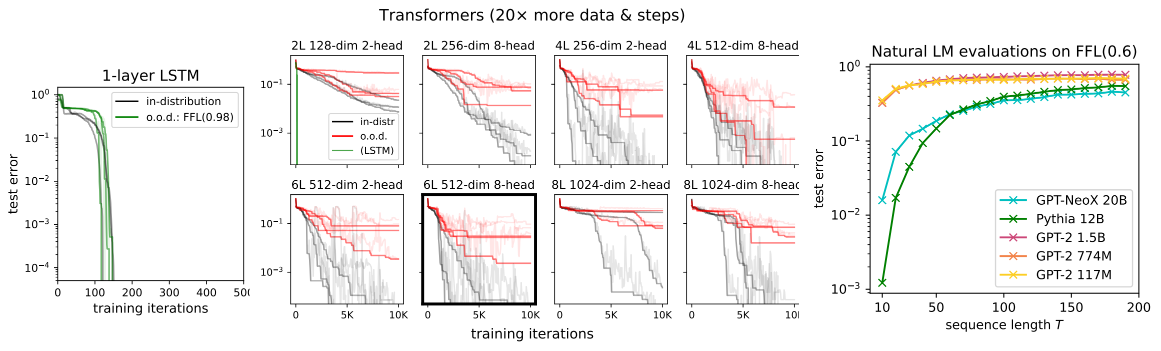
Intended functionality of attention. One can also appeal to the origin of attention mechanisms [Bahdanau et al., 2014, Luong et al., 2015, Vaswani et al., 2017]: attention was specifically designed to *attend* to[38] (i.e. selectively retrieve and copy) data over long-range dependencies. Indeed, it is easy to verify that a single attention head can perform the required lookup (see Proposition 3). It is thus logical to ask how well a purely attention-based architecture performs this elementary operation.

3.5.2 Attention glitches: a long tail of errors for Transformer FFLMs

In our main battery of synthetic experiments, we train neural language models to generate strings from the flip-flop language $\mathsf{FFL}(T = 512, \mathbf{p} = (0.1, 0.1, 0.8))$ (for short, $\mathsf{FFL}(p_{\mathtt{i}} = 0.8)$), and probe whether the networks robustly learn the language. Although every valid flip-flop string is supported in this distribution, some sequences are far rarer than others; we measure tail behavior via probes of extrapolation, defined here as out-of-distribution evaluations which amplify the probabilities of the rare sequences. To create these "challenging" sequences, we sample $> 3 \times 10^5$ sequences from $\mathsf{FFL}(0.98)$ (containing unusually many "sparse" sequences with mostly `ignore` instructions), as well

[37] A further discussion of the rationale for this specific manifestation of flip-flop sequence processing is deferred to Appendix 3.5.5.1.

[38] What this formally entails for representation and generalization is a topic of recent theoretical inquiry [Edelman et al., 2022, Wei et al., 2022a].



```
FFL(p_i = 0.9) : w 1 i 0 i 0 i 1 i 0 i 1 i 0 i 1 i 1 i 1 i 1 i 0 i 1 i 0 i 0 i 1 r 1 w 0 i 0 i 1 i 0 i 1 i 0 i 0 i 0 i 0 i 1 i 1 r 0 w 0 i 1 i 0 i 1 i 0 r 0
   FFL(0.98) : w 0 i 1 i 0 i 1 i 1 i 0 i 0 i 0 i 1 i 1 i 1 i 0 i 0 i 1 i 0 i 0 i 1 i 0 i 1 i 0 i 1 i 1 i 0 i 1 i 0 i 0 ... i 0 i 1 i 0 i 0 i 0 i 1 r 1
    FFL(0.1) : ... r 1 w 0 i 0 w 0 w 0 r 0 w 1 w 1 w 0 w 0 r 0 w 0 r 0 w 1 r 1 r 1 w 0 w 1 w 1 r 1 w 0 w 0 r 0 i 0 r 0 w 1 w 1 w 0 r 0 r 0 r 0 r 0 w 1 r 0
```

Figure 3.24: *Top:* Training curves of recurrent (left) vs. Transformer (center) architectures on FFLM, with *best-so-far* evaluation errors highlighted for clarity. **Transformers fail to extrapolate robustly** to the long tail of long-range dependencies, even on this extremely simple task of remembering one bit. The $\boxed{\text{bolded box}}$ contains our chosen 6-layer 19M-parameter canonical baseline model. We find that the ability to complete flip-flop language prompts emerges in natural language models, but is not robust (right). *Bottom:* examples from the sparser FFL(0.98) and denser FFL(0.1) distributions, causing distinct (*long-range* and *short-range*) failure modes for the baseline Transformer model.

as FFL(0.1) (many "dense" sequences). Training and evaluating the `read` accuracies of Transformer models of various sizes, as well as a recurrent LSTM model, we find the following (see Figure 3.24):

(R1) **Transformers exhibit a long, irregular tail of errors.** Such errors occur on both sparse and dense sequences. Further, a model's out-of-distribution test error varies significantly between random seeds, and even between iterates within the same training run.

(R2) **1-layer LSTM extrapolates perfectly.** In stark contrast, with 20 times fewer training samples and iterations, a small recurrent model achieves 100% accuracy, on 100 out of 100 runs.

As a counterpart to these findings, we observe similar anomalies in real LLMs, when prompted to complete natural textual embeddings (Figure 3.23, top right) of flip-flop tasks:

(R3) **10B-scale natural LMs can correctly process flip-flop languages, but not robustly.** Beyond a certain scale, natural language models can learn to process (natural embeddings of) flip-flop languages from in-context demonstrations. However, this emergent capability is not robust: there exist rare `read` errors, whose probabilities amplify as the sequence length T grows. We provide details for the few-shot evaluation protocol in Section 3.5.6.2.1.

3.5.3 Multiplicity of mechanisms for attention glitches

What failure mechanisms account for these reasoning errors, which occur for both short- and long-range dependencies? In this section, we discuss how Transformer self-attention modules, when tasked with representing flip-flops, can exhibit multiple (perhaps mutually entangled) failure mechanisms. The accompanying propositions are proven in Section 3.5.7.2 and Section 3.5.7.3.

An insufficient explanation: n-gram models. As a warmup, consider a language model $\widehat{\mathbf{Pr}}[x_{t+1}|x_{\leq t}]$ which only depends on the n most recent tokens in the context. Then, if $n \ll \frac{1}{1-p}$, the bulk of $\widehat{\mathbf{Pr}}$'s predictions on $\mathsf{FFL}(p_{\mathtt{i}} = p)$ can be no more accurate than random guessing. This recovers one qualitative trend (degradation of accuracy with dependency length) observed in the experiments. However, this cannot fully explain our findings: it fails to account for the incorrect predictions on dense sequences. Furthermore, the Transformers' outputs on $\mathsf{FFL}(0.98)$ are *mostly* correct; their accuracies on very long-range dependencies are nontrivial, despite not being perfect. There must therefore be subtler explanations for these errors.

Lipschitz limitations of soft attention. Moving to finer-grained failure mechanisms, a known [Hahn, 2020, Chiang and Cholak, 2022] drawback of soft attention is that its softmax operation is "too soft"– for any weight matrices with fixed norms, the attention gets "diluted" across positions as the sequence length T increases, and can fail to perform an intended "selection" operation. We provide a formal statement and proof (Proposition 4) in Section 3.5.7.2.

Difficulty of non-commutative tiebreaking. Can we simply robustify soft attention by replacing it with hard attention? We present a brief analysis which suggests that even hard attention can be brittle. In a stylized setting (one-layer models with linear position encodings), we show that self-attention can *confidently attend to the wrong index*, unless the weight matrices precisely satisfy an orthogonality condition (Proposition 5). This suggests the existence of *spurious local optima*, which we do not attempt to prove end-to-end; however, we provide supporting empirical evidence in the experiments in Section 3.5.7.3.

3.5.4 (Imperfect) mitigations for attention glitches

In this section, we investigate various approaches towards eliminating the long tail of reasoning errors exhibited by Transformer FFLMs. We select the 19M-parameter model (which has $L = 6$ layers, $d = 512$ embedding dimensions, and $H = 8$ heads) from Section 3.5.2 as a canonical baseline, and conduct precise evaluations of various direct and indirect interventions.

3.5.4.1 Direct solutions

Ideal solution: improving data coverage. Prior work has made clear that data significantly impacts the performance [Schuhmann et al., 2022, Eldan and Li, 2023]. Hence, we begin by examining what is perhaps the most obvious solution: removing the need for out-of-distribution extrapolation, by training directly on more diverse examples. Indeed, we verify that this works near-perfectly:

(R4) **Training on rare sequences works best, by a wide margin.** By training on a uniform mixture of FFL distributions with $p_{\mathtt{i}} = \{0.9, 0.98, 0.1\}$, the baseline architecture reliably converges to solutions with significantly fewer errors on each of these 3 distributions (teal violins in Figure 3.25). In 6 out of 25 runs, we did not detect a single error.

This should not be surprising, in light of the realizability of flip-flops by self-attention (and, more generally, the existence of shortcuts functionally identical to RNNs [Liu et al., 2023a]), and corroborates similar conclusions from [Zhang et al., 2021a]. We also find that weaker improvements emerge by straightforwardly increasing scale parameters in the model and training pipelines:

(R5) **Resource scaling (in-distribution data, training steps, network size) helps.** However, the improvements are orders of magnitude smaller than those in (R4), and we observe tradeoffs between sparse- and dense-sequence extrapolation; see the blue violins in Figure 3.25.

Another class of direct solutions is to *externalize the chain of thought* (CoT): train (or finetune, or prompt) the model to explicitly output the intermediate reasoning steps [Nye et al., 2021b, Wei et al., 2022b]. We do not investigate this strategy in this paper, and note that prior work has provided sufficient evidence to affirm its success in inducing the robust learning of recurrences on long synthetic sequences [Anil et al., 2022b, Zhou et al., 2022, Liu et al., 2023a]. Moreover, it cannot be guaranteed that a single indivisible reasoning step in a CoT is free of attention glitches; the focus of this work is to isolate and mitigate this intrinsic architectural issue.

3.5.4.2 Indirect algorithmic controls: a bag of regularization tricks

The interventions listed in Section 3.5.4.1 are all potentially practical, and may shed light on how closed-domain LLM hallucinations will diminish with data quality, scale, and improved inference strategies. However, it is not always *feasible* to implement these fixes under resource constraints (especially data). We next investigate an orthogonal design space, of how to robustify the *internal* memory mechanisms of neural sequence models. Note that the exceptionally strong extrapolative performance of the LSTM provides a "skyline", showing the possibility of far more robust architectures than the Transformer (in the flip-flop setting, with this restricted set of considerations).

Standard regularization heuristics. There is a large array of not-fully-understood algorithmic tricks for "smoothing" the behavior of LLMs. We test the extrapolative behavior of models trained with weight decay and dropout (at the attention, feedforward, and embedding layers), as well as a host of algorithmic choices known to modulate generalization (batch sizes, learning rates, optimizer hyperparameters, position embeddings, activation functions). Due to the extreme variability noted in (R1), we quantify effects on extrapolation by training and evaluating at least 25 replicates for each choice under consideration.

Attention sharpening: a non-standard regularization technique. Inspired by the "diluted hard attention" calculation in Section 3.5.3, and the fact that the attention heads of trained models do not attend sharply (see Figure 3.26), we train Transformer models with *attention-sharpening regularizers*:[39]

[39]While less popular, such losses have been used to sparsify dependencies in similar contexts [Zhang et al., 2018, Sukhbaatar et al., 2021].

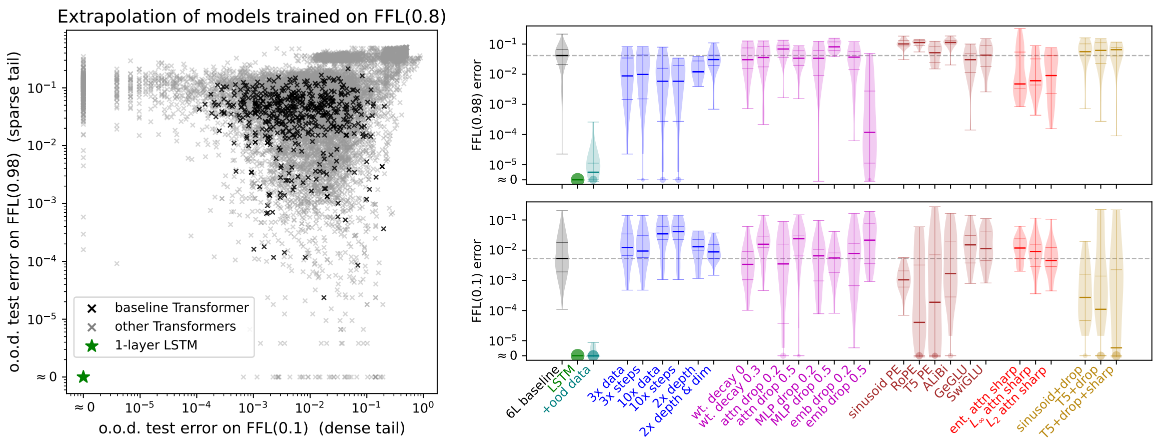


Figure 3.25: A long tail of flip-flop errors for 10,625 Transformer models. *Left:* Out-of-distribution evaluations for all models; some algorithmic choices help substantially (note the logarithmic axes), but **nothing we tried, aside from training on o.o.d. data, could fully eliminate attention glitches**. *Right:* Effects of individual architectural and algorithmic choices on both types of extrapolation (sparse and dense sequences). Some configurations reduce attention glitch rates by orders of magnitude. Horizontal marks denote {min, 25%, median, 75%, max} test errors on $> 3 \times 10^5$ predictions, over 25 replicates (500 for the baseline model). Dots at the bottom indicate runs with 0 error.

during training, for attention weights $\alpha \in \Delta([T])$, adding differentiable loss terms which encourage sparsity (e.g. the mixture's entropy $H(\alpha)$, or negative p-norms $-\|\alpha\|_2$, $-\|\alpha\|_\infty$).

(R6) **Many algorithmic choices influence extrapolative behaviors.** We find that some architectural variants and regularization tricks have orders-of-magnitude effects on the out-of-distribution performance of Transformers; see the purple, brown, red, and gold violins in Figure 3.25 (right). Our strongest improvements on sparse sequences are obtained by large (0.5) embedding dropout and attention-sharpening losses; on dense sequences, non-trainable position embeddings are the most helpful.

(R7) **Despite many partial mitigations, nothing eliminates attention glitches entirely.** The scatter plot in Figure 3.25 (left) gives an overview of our entire search over architectures and hyperparameters, showing (dense-sequence error, sparse-sequence error) pairs for *every* model we trained. We found it extremely difficult to find a setting that reliably produces Transformer models with simultaneous improvements over the baseline on sparse and dense sequences. Recall that it is trivial to do so with an LSTM model.

3.5.4.3 Preliminary mechanistic study

In this section, we move to a simpler setting to gain finer-grained understanding of how sparsity regularization affects the learned solutions. Specifically, we look at the task of *simulating the flip-flip automaton* (Example 2), whose inputs consist of $\{\sigma_0, \sigma_1, \perp\}$ as two types of `write` and 1 no-op. This task can be solved by a 1-layer Transformer with a single attention head which attends sparsely on the most recent `write` position. It also serves as a building block for more complex tasks [Liu et al.,

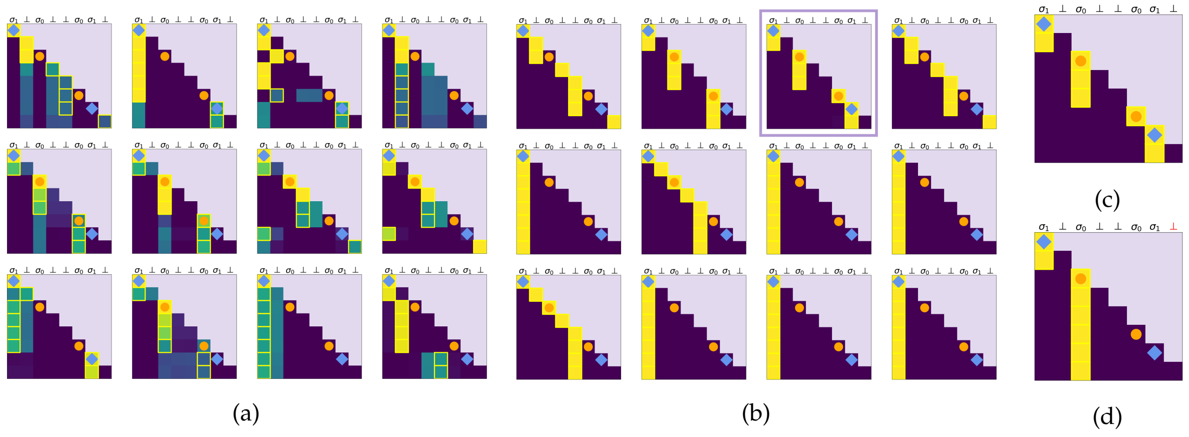


Figure 3.26: Causal attention patterns for flip-flop simulation (Definition 11); orange dots / blue diamonds mark the positions of write tokens σ_0 / σ_1. (a),(b) are subselected respectively from a regular (non-sparse) and a sparse multi-layer model (details in Appendix 3.5.6.5). (c), (d) are from two 1-layer 1-head models. The attention pattern highlighted by the purple box in (b) coincides with the "ideal" attention pattern in (c). However, sparse models can be wrong, as shown in (d) (error marked in red).

2023a], hence observations from this simple setup can potentially be useful in broader contexts.

Figure 3.26 shows examples of attention patterns on the flip-flop simulation task, subselected from 6-layer 8-head models trained with and without attention-sharpening regularization. It is evident that the attention patterns of the sparse model are less complex and easier to interpret compared to those of the un-regularized model. For example, we can identify one head in the sparse model that exactly coincide with the attention pattern [40] that an "ideal" 1-layer 1-head model implements (Figure 3.26c).

(R8) **Attention-sharpening regularizers successfully promote hard attention, but errors persist.** As mentioned in (R7), attention-sharpening regularization cannot fully eliminate the sporadic errors, which are partially induced by the complexity and redundancy of attention patterns. Moreover, sharpened attention can induce additional failure modes, such as confidently attending to incorrect `write` positions. An example is demonstrated in Figure 3.26d, where the attention focuses on an initial `write`, likely caused by the fact that earlier positions are overemphasized due to the use of causal attention masks. Another example occurs in length generalization, where the attention is correct at positions earlier in the sequence, but starts to confidently focus on wrong positions as it moves towards later positions (Proposition 5). Details and more discussions are provided in Section 3.5.6.5.

3.5.5 Discussion

We have introduced *flip-flop language modeling* (FFLM), a synthetic benchmark for probing the fine-grained extrapolative behavior of neural sequence models, based on a one-bit memory operation

[40]While it is well-known that attention patterns can be misleading [Jain and Wallace, 2019, Bolukbasi et al., 2021, Meister et al., 2021] at times, they do provide upper bounds on the magnitude of the dependency among tokens. These upper bounds are particularly useful in the case of (1-)sparse attentions: a (near) zero attention weight signifies the absence of dependency, which greatly reduces the set of possible solutions implemented.

which forms a fundamental building block of algorithmic reasoning. Transformer models, trained on insufficiently diverse flip-flop sequences, make a long tail of sporadic reasoning errors, which we call *attention glitches*. Through extensive controlled experiments, we find that many algorithmic mitigations can reduce the frequency of attention glitches, but none can eliminate them entirely. FFLM provides a minimalistic setting in which Transformers are far inferior to recurrent sequence models, with respect to multiple criteria (efficiency, stability, and extrapolation).

Below, we discuss the broader implications of our findings.

3.5.5.1 Why *this* flip-flop language?

While FFLM is intended to be simple and synthetic, there are alternative forms that can be even simpler and argubly more natural for certain purposes. For instance, a purer instantiation of flip-flop sequence processing is used in both Section 3.4 and the mechanistic interpretability experiments (Section 3.5.4.3), where the sequence-to-sequence network is tasked with *non-autoregressive transduction*: given the sequence of input symbols $\sigma_1, \ldots, \sigma_T$, output the sequence of states $q_1, \ldots, q_T$. This is most natural when studying the Transformer architecture's algebraic representations in their most isolated form.

Our autoregressive sequence modeling setting is a slight departure from this setting; we discuss its properties and rationale below.

- The autoregressive setting "type-checks" exactly with standard state-of-the-art autoregressive (a.k.a. causal, forward, or next-token-prediction) language modeling. This makes it more convenient and intuitive as a plug-and-play benchmark.
- The cost is a layer of indirection: the model needs to associate "instruction" tokens with their adjacent "data" tokens. This is a natural challenge for representation learning, and is certainly a necessary cursor for robust extrapolation on natural sequences that embed similar tasks (like those considered in Figure 3.23c). It is straightforward to remove this challenge: simply tokenize at a coarser granularity (i.e. treat (instruction, data) pair as a distinct vocabulary item).
- The multi-symbol (and variable-length-symbol, etc.) generalizations of the binary flip-flop language are more parsimonious. If there are n instead of 2 tokens, this language can be encoded with $= n + 3$ commands. Without the decoupling of "instruction" tokens from "data", the vocabulary size would scale suboptimally with n.
- The conclusions do not change: in smaller-scale experiments, we observe the same extrapolation failures between the autoregressive and non-autoregressive task formulations.

3.5.5.2 What does this entail about hallucinations in natural LLMs?

The motivating issue for this work is the phenomenon of "closed-domain hallucinations" in non-synthetic LLMs (e.g. the errors demonstrated in Figure 3.2). We hypothesize that attention glitches occur in the internal algorithmic representations of Transformer models of natural language, and that they account for (a non-negligible portion of) the reasoning errors encountered in practice. To our knowledge, this is the first attempt to attribute model hallucinations to a systematic architectural flaw in the Transformer. However, confirming or refuting this hypothesis is far outside the scope of this paper; the opaque indirections and lack of adequate controls on the training data present significant methodological challenges.

Even precisely articulating this hypothesis leaves degrees of freedom which are difficult to resolve. At a high level, *we hypothesize that attention glitches cause (some) closed-domain hallucinations in Transformer models of more complex languages*. However, due to the fact that neural networks' internal representations evade simplistic mechanistic characterization, it is a significant challenge to formulate a rigorous, testable version of this hypothesis. We discuss the subtleties below.

First, we discuss a more general notion of attention glitches, of which the flip-flop errors considered in this papers are a special case. We define attention glitches as *failures of trained attention-based networks to implement a hard retrieval functionality perfectly*. To formalize this notion, there are several inherent ambiguities– namely, the notions of "hard retrieval" and "perfectly", as well as the granularity of "subnetwork" at which an attention glitch can be defined non-vacuously. The FFLM reasoning errors considered in this work provide a minimal and concrete resolution of these ambiguities. We discuss each of these points below:

- **Hard retrieval:** To succeed at FFLM, a network's internal representations must correctly implement the functionality of retrieving a single bit (from a sequence of bits, encoded unambiguously by the network), selected via the criterion of "most recent `write` position". This can be expanded into a richer functional formulation of hard attention, by generalizing the set of possible *retrieved contents* (a discrete set of larger cardinality, or, even more generally, a continuous set), as well as more complex *selection criteria* (e.g. "least recent position").
- **Ground truth:** Of course, to define "errors" or "hallucinations" in reasoning, there must be a well-defined *ideal* functionality. For FFLM, the notion of "closed-domain" reasoning and hallucinations is evident: the ideal behavior is for a model's outputs to coincide with that of the flip-flop machine on all input sequences. This straightforwardly generalizes to all formal languages, where the model is expected to correctly produce the deterministic outputs of automata which parse these languages. By considering expanded notions of "ground truth", it is possible to capture other notions of model hallucinations (such as incorrectly memorized facts). Our work does not address open-domain hallucinations, which may be unrelated to attention glitches.
- **Submodules:** Towards attributing implementations and errors to localized components of a network, it is impossible to provide a single all-encompassing notion of "localized component". This is a perennial challenge faced in the mechanistic interpretability literature. Our work considers two extremes: the entire network (in the main experiments, where we probe end-to-end behavior), and a single self-attention head (in Sections 3.5.3, 3.5.4.3 and Section 3.5.6.5, in which we probe whether a single attention head can learn multiplication in the flip-flop monoid). Even when considering the same functionality, attention glitches can be considered for different choices of "submodule".[41] Our results reveal a key subtlety: in the presence of overparameterization (more layers and parallel heads than necessary according to the theoretical constructions), Transformers learn to process flip-flop languages via soft attention.

[41]Beyond the two extremes considered in this work, some examples include "a subset of attention heads", "a subset of layers", and "a subspace of the entire network's embedding space".

We expect that to effectively debug the full scope of LLM hallucinations, all of the above choices will need to be revisited, perhaps in tandem.

We hypothesize that the algorithmic reasoning capabilities of real LLMs (i.e. their ability to recognize, parse, and transduce formal symbolic languages) are implemented by *internal* subnetworks whose functionalities can be identified with generalizations of the flip-flop machine. To the extent that such modules exist, attention glitches (the failure of these modules to represent the flip-flop operations perfectly, due to insufficient training data coverage) cause sporadic end-to-end errors ("closed-domain hallucinations"). In this work, we have treated the *external* case (where the task is to learn the flip-flop directly).

3.5.5.3 Paths to hallucination-free Transformers?

Our findings suggest that in the near term, there are many mutually-compatible approaches for reducing the frequency of attention glitches: data (particularly with high diversity), scale, and various forms of regularization. Yet, the strikingly outsized benefit of replacing the Transformer with an LSTM network suggests that *architectural* innovations towards the same ends are well worth examining. Obtaining a practical best-of-both-worlds architecture is a grand open challenge, for which new recurrent designs [Katharopoulos et al., 2020, Peng et al., 2023, Dao et al., 2022] show great promise. Note that we do not make the claim that recurrent architectures are the only ones which can extrapolate robustly.[42]

3.5.6 Experimental details

3.5.6.1 Details for LLM addition prompts (Figure 3.2)

These addition problem queries serve as a quick demonstration of (1) non-trivial algorithmic generalization capabilities of Transformer-based LLMs; (2) the brittleness of such capabilities: we directly address this type of reasoning error in this work. Table 3.2,3.3 show these queries and results in detail.

We emphasize that these examples were selected in an adversarial, ad-hoc manner; we do not attempt to formalize or investigate any claim that the errors made by larger models are at longer sequence lengths. We also cannot rule out the possibility that some choice of prompt elicits robust algorithmic reasoning (e.g. the prompting strategies explored in [Zhou et al., 2022]). The only rigorous conclusion to draw from Figure 3.2 is that of non-robustness: even LLMs exhibiting state-of-the-art reasoning continue to make these elementary errors for some unambiguous queries with deterministic answers. It was last verified on May 8, 2023 that GPT-4 (in its ChatGPT Plus manifestation) demonstrates the claimed failure mode.

3.5.6.2 Extrapolation failures of standard Transformers (Section 3.5.2)

This section provides full details for our empirical findings (R1) through (R3).

[42] In particular, for algorithmic reasoning capabilities corresponding to the recurrent execution of deterministic finite-state machines, the results of [Liu et al., 2023a] imply that the Transformer has a "reparameterized" recurrent inductive bias, which *parallelizes* and *hierarchizes* any looped recurrence.

Input	GPT-3.5	GPT-4	Answer
8493 + 2357	10850 ✓	10850 ✓	10850
84935834 + 23572898	108008732 ✗	108508732 ✓	108508732
9991999919909993 + 6109199190990097	16111199100810090 ✗	16101199100890090 ✗	16101199110900090

Table 3.2: Examples (in Figure 3.2) of GPT variants on addition: While models tend to succeed at additions with a small number of digits, they (nondeterministically) fail at longer additions.

Input	GPT-3.5	GPT-4	Answer
4491 + 8759	13250 ✓	13250 ✓	13250
80087394 + 63457948	143045342 ✗	143545342 ✓	143545342
5101611078665398 + 8969499832688802	1.4071110911354202e+16 ✗	14071110911354196 ✗	14071110911354200

Table 3.3: More examples of GPT variants on addition: While models tend to succeed at additions with a small number of digits, they (nondeterministically) fail at longer additions.

Architecture size sweep. We consider a sweep over Transformer architecture dimensionalities, varying the three main size parameters. We emphasize that these are somewhat larger than "toy" models: the parameters go up to ranges encountered in natural sequence modeling (though, of course, far short of state-of-the-art LLMs).

- The *number of layers* (depth) $L \in \{2, 4, 6, 8\}$.
- The *embedding dimension* $d \in \{128, 256, 512, 1024\}$.
- The *number of parallel attention heads* per layer $H \in \{2, 4, 8, 16\}$. In accordance with standard scaling rules-of-thumb, each head's dimension is selected to be d/H.

Other hyperparameter choices. We use a sequence length of $T = 512$, again to reflect a typical length of dependencies considered by nontrivial Transformer models. We use a canonical set of training hyperparameters for this sweep: the AdamW [Loshchilov and Hutter, 2017b] optimizer, with $(\beta_1, \beta_2) = (0.9, 0.999)$, learning rate 3×10^{-4}, weight decay 0.1, 50 steps of linear learning rate warmup, and linear learning rate decay (setting the would-be 10001th step to 0). We train for 10000 steps on freshly sampled data, and choose a minibatch size of 16; consequently, the models in this setup train on 81,920,000 tokens.

Training and evaluation data. We probe the extrapolative behavior of Transformers on the flip-flop language, training on online samples containing mostly moderate-length dependencies ($p_{\mathtt{i}} = 0.8, p_{\mathtt{w}} = p_{\mathtt{r}} = 0.1$), and evaluating on a distribution containing longer-range dependencies ($p_{\mathtt{i}} =$

0.98, $p_{\mathtt{w}} = p_{\mathtt{r}} = 0.01$). Every 100 training steps, we evaluate out-of-distribution test errors achieved by these models, on an online evaluation set of 10^3 sequences (which is identical between and within runs; training curves show these errors), containing 3567 occurrences of the `r` instruction. For offline evaluation, we expand this test set to 10^5 sequences, containing 353875 `r` commands, to obtain more precise measurements of o.o.d. error. Training curves are shown with the smaller test set; all other results are reported using the larger one.

(R1) **Transformers exhibit a long, irregular tail of errors.** Figure 3.27 shows training curves for 3 replicates (random seeds) in each setting, while the scatter plot in the main paper shows variability of out-of-distribution accuracy across random seeds for the baseline setup. We find that Transformers sometimes succeed at extrapolation, but erratically.

(R2) **1-layer LSTM extrapolates perfectly.** We train a 1-layer LSTM [Hochreiter and Schmidhuber, 1997a] network, with hidden state dimension 128 (for a total of 133K parameters), for 500 steps with the same optimizer hyperparameters as above. The LSTM model achieves exactly 0 final-iterate o.o.d. error, over 100 out of 100 replicates.

Canonical baseline. We select the **6-layer, 512-dimensional, 8-head** architecture (with 19M trainable parameters) as our *canonical baseline* model: it is large in relevant dimensions[43] to real Transformers, while being small enough to allow for thousands of training runs at a reasonable cost. To fully understand the variability of this single architectural and algorithmic setup, we train and evaluate 500 replicates in this setting.

Random data vs. random initialization. Recent synthetic probes on the surprising behavior of deep neural nets on hard synthetic tasks [Barak et al., 2022a, Garg et al., 2022] obtain additional insights by disentangling the effects of *data randomness* (i.e. the precise sequence of minibatches) vs. *model randomness* (e.g. random initialization and dropout). We provide a quick demonstration in Figure 3.28 (left) that *both sources of stochasticity matter*. We do not perform a more detailed investigation of their precise influence and roles.

Fully generative setting: similar negative results. As mentioned in Section 3.5.1.2, to capture a setting closer to standard autoregressive (sometimes called GPT-style) language modeling, we find a similar failure to extrapolate when models are trained to predict all tokens, rather than only the deterministic ones (x_{t+1} such that $x_t = \mathtt{r}$). Figure 3.28 (right) exhibits some training curves for this setting, showing non-extrapolation, variability, and instability. We observe that training (to in-distribution convergence) takes slightly longer in this setting, and usually succeeds with the baseline architecture. We do not perform further controlled experiments in this setting.

[43]Except the vocabulary size. In preliminary experiments, we obtained similar findings in the case of token spaces larger than $\{0, 1\}$.

3.5.6.2.1 Evaluating real LLMs on flip-flops

We provide a quick corroboration that while LLMs in practice can perform in-context reasoning when the sequences are unambiguously isomorphic to a flip-flop language. We use the natural language example from Figure 3.23 (top right), and evaluate the capability of popular pretrained LLMs to correctly remember the state of a light switch. Specifically, `write` instructions in the FFLM task are either "Alice turns the light off" or "Alice turns the light on". The `ignore` instructions are either "Bob eats a banana" or "Bob eats an apple". All models are prompted with a translated, length-16 FFLM task that's been translated to English in this way before evaluation.

We measure this accuracy as a function of the sequence length for several well-known LLMs, including GPT-2, GPT-2-large, GPT-2-xl, Pythia-12C, and GPT-NeoX-20B. Figure 3.24 shows how well these models perform on this task (i.e. the correctness of the model when prompted with "The light is turned ") as the sequence length is varied. Consistent with the findings of this paper, larger models tend to perform best at this task, and the quality of all models deteriorates with increased sequence length. Each point on the plot considers 500 sequences of the indicated length. All models were prompted with a randomly generated, length 16 flip flop sequence to allow the model to learn the task in context. Accuracy is measured according to the frequency with which the model correctly predicts the current state of the light switch, as described in Section 3.5.6.2.1.

(R3) **10B-scale natural LMs can correctly process flip-flop languages, but not robustly.**

Note that it is impossible to quantify the degree to which these sequences are "in-distribution" (it is unlikely that any sequences of this form occur in the training distributions for these LLMs). Much like linguistic reasoning evaluations in the style of BIG-bench [Srivastava et al., 2022], we rely on the emergent capability of in-context inference [Brown et al., 2020] of the task's syntax and semantics. As discussed in Appendix 3.5.5.2, this layer of indirection, which is impossible to avoid in the finetuning-free regime, can cause additional (and unrelated) failure modes to those studied in our synthetic experiments. Fully reconciling our findings between the synthetic and non-synthetic settings (e.g. by training or finetuning on sequences of this form, or via mechanistic interpretation of non-synthetic language models) is outside the scope of this paper, and yields an interesting direction for future work.

Interacting directly using FFLM sequences? Given the conversational abilities of LLMs, another way to interact with an existing pretrained model is to explain the definition of FFLM in natural language, and ask the model to output the correct state for `r`. We test this using ChatGPT (with GPT-4), as demonstrated in Figure 3.29. ChatGPT seems to understand the rules and can get short sequences correct (up to sequence length 400), but makes errors with unexpected connections on longer sequences.

3.5.6.3 Benefits of scale (Section 3.5.4.1)

In Section 3.5.4.1, we discussed mitigations that directly modify the training distributions and resources:

(R4) **Training on rare sequences works best, by a wide margin.** See the teal violins in Figure 3.30.

(R5) **Resource scaling (in-distribution data, training steps, network size) helps.** Training on more data from the same distribution, as well as for more steps on the same examples, both improve sparse-sequence performance, at the expense of dense-sequence performance (blue violins in Figure 3.30). There is no discernible monotonic relationship between any of the Transformer's standard architecture size parameters (i.e. number of layers, embedding dimension, and number of parallel self-attention heads per layer) and extrapolative performance (navy violins).

We provide more results specifically related to scaling along various axes. As shown in Figure 3.30, scaling helps improve the OOD performance, especially when more OOD data are introduced. However, the benefit is not clear, especially on dense sequences.

3.5.6.4 Indirect algorithmic controls for extrapolation (Section 3.5.4.2)

As shown in Figure 3.25, various architectural, algorithmic and regularization choices can help improve over the baseline Transformer. We recall the main findings:

(R6) **Many algorithmic choices influence extrapolative behavior.** We sweep over various forms of implicit and explicit regularizers; see Figures 3.31 and 3.32. Details are provided below.

(R7) **Despite many partial mitigations, nothing eliminates attention glitches entirely.** Refer to the scatter plot in Figure 3.25 (left) for a visualization of *every* training run.

Details for architecture variants. There is no clear consensus on the advantages and drawbacks of various positional encodings, but it has been known Dai et al. [2019] that the choice of positional symmetry-breaking scheme modulates long-sequence performance on natural tasks. We evaluate various choices which appear in high-profile LLMs: sinusoidal, learned, ALiBi [Press et al., 2021], and RoPE [Su et al., 2021]. We also try the *zero* positional encoding (which breaks symmetry via the causal attention mask; see Haviv et al. [2022]. We find that non-trainable position encodings help on dense sequences ($\mathsf{FFL}(0.1)$), but have no clear benefit on sparse ones ($\mathsf{FFL}(0.98)$) which require more handling of long-term dependency. We also try the gated activation units considered by [Shazeer, 2020].

Details for attention sharpening. There are many possible choices of continuous regularization terms which can promote sparsity in an attention head's weights– we consider entropy, negative L_2 loss, and negative L_∞ loss. These terms are averaged across every attention head in the Transformer, and added as a surrogate objective during training. We perform a large grid sweep over coefficients $\{0.01, 0.03, ..., 0.1, 0.3, 1, 10, 30\}$, annealing schedules (linear and oscillating, starting from 0, 2000, and 5000 steps), and display in Figure 3.32 the 3 choices which appear on the Pareto front.

Optimizer. We also varied the optimizer parameters for AdamW ($\beta_1 \in \{0.85, 0.9, 0.95\}$, $\beta_2 \in \{0.95, 0.99, 0.999\}$, learning rate $\eta \in \{0.0001, 0.0003, 0.001, 0.003\}$) and found no significant improvements to extrapolation performance.

3.5.6.5 Preliminary mechanistic study and challenges (Section 3.5.4.3)

We continue the discussions in Section 3.5.4.3 and provide preliminary mechanistic interpretability results on *simulating the flip-flop automaton* (Definition 11). Recall the main takeaway:

(R8) **Attention-sharpening regularizers successfully promote hard attention, but errors persist.**

Sparsity regularization helps sharpen the attention. Figure 3.34a,3.34b compare the attention patterns of 1-layer 1-head models with or without attention-sharpening regularization. While both types of models give correct results, the attention-sharpened model puts all attention weights to the most recently `write` position, which is the solution given according to the definition of the task, whereas the attention patterns of the non-regularized model (Figure 3.34a) are much less clean.

Are there solutions other than the "ideal" solution? There is a solution naturally associated with the definition of the flip-flop automaton (i.e. the sparse pattern shown in Figure 3.34b), but it is not necessarily the *only* solution. For example, an equally valid (dense) solution is for the model to attend to every write token of the correct type. This is what the non-regularized (dense) models seems to be implementing, as seen in Figure 3.34a, except for the final row where the model puts non-negligible amount of weight on a `write` of a different type.

Are attention patterns reliable for interpretability? Prior work has pointed out the limitations of interpretations based solely on attention patterns [Jain and Wallace, 2019, Bolukbasi et al., 2021]. The intuition is that attention patterns can interact with other components of the network in various ways; for example, $\boldsymbol{W}_V$ can project out certain dimensions even though they may have contributed to a large attention score. Hence, for multi-layer multi-head non-sparse models, the magnitude of attention weights may not have an intuitive interpretation of "importance" [Meister et al., 2021]. For example, Figure 3.35 shows examples where the attention on an incorrect token may be higher than that of the correct token. [44] However, in a 1-layer 1-head model, 1-sparse attention as shown in Figure 3.34b indeed offers interpretability, since if zero attention weight [45] necessarily means the absence of dependency, which greatly reduces the set of possible solutions implemented. As shown in Figure 3.34c, a `write` may not attend to itself due to the presence of residual link, but the attentions for `read` always focus on the closest `write` as intended.

Sporadic errors persist. Section Section 3.5.4.1 (R5) showed that none of the mitigations was successful at making Transformers reach 100% accuracy. One common failure mode is long-range dependency, where the input sequences contain very few `writes`. The failure could be attributed to multiple factors; we will explore one aspect related to attention patterns, demonstrated with a 1-layer 1-head Transformers with linear position encoding, on a length-834 sequence with 2 `writes`. As shown in Figure 3.33, the attentions for positions early in the sequence correctly attend to the most recent `write`. However, attention starts to "drift" as we move to later positions, and the positions at the end of the sequence attend entirely [46] to the recent `read` tokens, which contains no information for solving the

[44] However, if we consider the "importance / influence" as measured by the norms of the attetnion-weighted value vectors, then the max norm still corresponds to the correct token, which helps explain why the final output is correct.

[45] By "zero" we mean an attention score on the magnitude of 1e-8 in the experiments.

[46] The attention weights that are not on the most recent `write` sum up to around 1e-7.

task. This may be because the attention weights are affected too much by the position encodings, as discussed in Proposition 5.

Optimization hurdles. While sparse solutions may preferred for various reasons, sparsity itself is not sufficient to guarantee good performance: As shown in Figure 3.34d, sparsity regularization can lead to bad local minima, where the model tends to (incorrectly) rely on earlier positions. This is observed across different types of sparsity regularization. While we do not yet have a full explanation of the phenomenon, a possible explanation for this bias is that earlier positions show up more often during training, due to the use of the causal attention: a valid flip-flop solution is for the model to attend to every `write` token of the correct type; positions earlier in the sequence get included in more subsequences because of the causal mask, and are hence more likely to be attended to. We also observe that the phenomenon seems to be closely related to the training distribution. For example, the model is much more likely to get stuck at a bad local minima when $p(\perp) = 0.5$ (denser sequences) compared to $p(\perp) = 0.9$ (sparse sequences).

Effect of sparsity regularization on training dynamics An interesting future direction is to understand the learning dynamics of flip-flop tasks with attention-sharpening regularization, as suggested by the (quantitively and qualitatively) different results and optimization challenges. As some initial empirical evidence that the regularization indeed have a large impact on the dynamics, we found that sharpened attention seems to have a regularization effect on the weight norms and lead to different behaviors of the attention heads (Figure 3.36).

More examples of attention patterns Figure 3.37 shows the full set of attention patterns of two 6-layer 8-head models trained with and without attention-sharpening regularization, corresponding to Figure 3.26 (a,b). Attention-sharpening regularization can be applied in different ways; for example, Figure 3.38 shows results of a model for which only the first layer is regularized. The attention patterns of subsequent layers remain sharpen, even though there is no explicit regularization.

3.5.6.6 Software, compute infrastructure, and resource costs

GPU-accelerated training and evaluation pipelines were implemented in PyTorch Paszke et al. [2017]. For the FFLM experiments, we used the `x-transformers`[47] implementations of the Transformer architecture and variants. For the fine-grained mechanistic interpretability experiments on the pure flip-flops, we used the "vanilla, GPT-2"-like Transformer implementation published by HuggingFace [Wolf et al., 2019]. We plan to make our benchmarks and training code publicly available.

Each training run was performed on one GPU in an internal cluster, with NVIDIA P40, P100, V100, and RTX A6000 GPUs, with at least 16GB of VRAM. Each (6-layer, 512-dimensional, 8-head) baseline model took $\sim$10 minutes to train (and evaluate online) for 10^4 steps. A nontrivial fraction of the compute time ($\sim$ 20%) was spent on fine-grained evaluation through the coarse of training. The vast majority of training runs are close to these specifications; consequently, one set of replicates under

[47]`https://github.com/lucidrains/x-transformers`

identical conditions (i.e. each violin plot in each figure) is the product of ∼4 GPU-hours of training time.

We hope that this computational investment will aid in understanding how to build robust Transformer models and training pipelines at much larger scales.

3.5.7 Formal understanding on mechanisms behind attention glitches (Section 3.5.3)

This section formally states and proves the propositions in Section 3.5.3.

As a quick recap, a Transformer [Vaswani et al., 2017] consists of multiple self-attention layers. Given d-dimensional embeddings of a length-T sequence, denoted as $\boldsymbol{X} \in \mathbb{R}^{T\times d}$, a self-attention layer f computes

$$f(\boldsymbol{X}) = \phi(\boldsymbol{W}_V \text{softmax}(\boldsymbol{X}\boldsymbol{W}_Q\boldsymbol{W}_K^\top \boldsymbol{X}^\top)\boldsymbol{X}\boldsymbol{W}_V\boldsymbol{W}_C). \tag{3.6}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d\times k}$ for $k \leq d$ are the query and key matrix; $\boldsymbol{W}_V, \boldsymbol{W}_C^\top \in \mathbb{R}^{d\times k}$ project the representations from and back to $\mathbb{R}^d$. softmax calculates row-wise softmax. $\phi : \mathbb{R}^d \to \mathbb{R}^d$ is a 2-layer fully-connected network. Residual links and layer norm can be optionally included at different places of a self-attention layer.

3.5.7.1 Realizability of FFL by small Transformers

Proposition 3. *A 2-layer 1-head Transformer with residual connections can represent "deterministic" FFL.*

Proof. Let us consider predicting in the deterministic mode (Section 3.5.1.2). Then we need to predict x_{t+1} given $x_{1:t}$ with $x_t = \texttt{r}$. In order to do this, we need to find the largest $\tau < t$ such that $x_\tau = \texttt{w}$ and output $x_{\tau+1}$. There are multiple ways to implement this, we will consider the following: (1) layer 1 converts FFL to the flip-flop automaton (Definition 11), (2) layer 2 implements the flip-flop construction. For layer 2, we can use the construction described in Liu et al. [2023a]. Here we present the full construction for completeness.

We will consider a two-layer Transformer with one head in each layer followed by a 2-layer MLP and a residual connection. In particular, for $x \in \{\texttt{w}, \texttt{r}, \texttt{i}, 0, 1\}^T$:

$$f(x) = \phi_2(\boldsymbol{W}_V^{(2)} \text{softmax}(f_1(x)\boldsymbol{W}_Q^{(2)}\boldsymbol{W}_K^{(2)\top} f_1(x)^\top) f_1(x)\boldsymbol{W}_V^{(2)}\boldsymbol{W}_C^{(2)})$$
$$\text{where } f_1(x) = E(x) + \phi_1(\boldsymbol{W}_V^{(1)} \text{softmax}(E(x)\boldsymbol{W}_Q^{(1)}\boldsymbol{W}_K^{(1)\top} E(x)^\top)E(x)\boldsymbol{W}_V^{(1)}\boldsymbol{W}_C^{(1)})$$

where $E(x) \in \mathbb{R}^{T\times d}$ is the encoding for the input sequence x given some encoding function E.

Our construction is as follows:

- Select $d = 7, k = 2, H = 1$ (recall from Equation 3.6that d, k are the dimensions of $\boldsymbol{W}_Q, \boldsymbol{W}_K$). Among the $d = 7$ embedding dimension, two dimensions are for the operations (`w` versus `r`,`i`), two for the two `write` values, one for the positional embedding, one for padding, and the final dimension is for storing whether the previous position is the most recent `write`, as calculated by the first layer.

- Select input symbol encodings such that for the token at position t, denoted as x_t,

$$E(x_t) := \mathbb{1}[x_t = \mathtt{w}]e_1 + \mathbb{1}[x_t = \mathtt{r} \vee x_t = \mathtt{i}]e_2 + \mathbb{1}[x_t = 0]e_3 + \mathbb{1}[x_t = 1]e_4 + e_5 + P_t \in \mathbb{R}^7,$$

  where P_t is the positional encoding. We use the linear positional encoding $P_t := (t/C) \cdot e_6$, for some (large) constant C. For a fixed sequence length T, we can set $C = T$.

- $\boldsymbol{W}_Q^{(1)} := \begin{bmatrix} e_5 & e_5 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, $\boldsymbol{W}_K^{(1)} := \begin{bmatrix} 3c\frac{e_1}{2T} & ce_6 \end{bmatrix} \in \mathbb{R}^{7\times 2}$ for $c = O(T\log(T))$, $\boldsymbol{W}_V^{(1)} := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, and $\boldsymbol{W}_C^{(1)^\top} := \begin{bmatrix} e_7 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$.

- $\boldsymbol{W}_Q^{(2)} := \begin{bmatrix} e_5 & e_5 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, $\boldsymbol{W}_K^{(2)} := \begin{bmatrix} ce_7 & ce_6 \end{bmatrix} \in \mathbb{R}^{7\times 2}$ for $c = O(T\log(T))$, $\boldsymbol{W}_V^{(2)} := \begin{bmatrix} e_4 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, and $\boldsymbol{W}_C^{(2^\top} := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$.

In layer 1, the unnormalized attention score for query position i to key position j is

$$\left\langle \boldsymbol{W}_Q^{(1)^\top} x_i, \boldsymbol{W}_K^{(1)^\top} x_j \right\rangle = \left\langle \frac{c}{T} \cdot \left[\frac{3}{2} \cdot \mathbb{1}[x_j = \mathtt{w}], j\right], [1, 1] \right\rangle = \frac{c}{T} \cdot \left(\frac{3}{2}\mathbb{1}[x_j = \mathtt{w}] + j\right).$$

Note that the max attention value for position i is achieved at i if $x_{i-1} \neq \mathtt{w}$, else the max is achieved at position $i-1$.

In the setting of hard attention, the output for the i_{th} token after the attention module is $\mathbb{1}[x_{i-1} = \mathtt{w} \vee x_i = \mathtt{w}]e_7$. Now similar to the constructions in Liu et al. [2023a] (Lemma 6), with a appropriate choice of $c = O(T \log T)$, we can approximate hard attention by soft attention, and subsequently use the MLP to round the coordinate corresponding to e_7. The MLP otherwise serves as the identity function. Together with the residual link, the first layer output (i.e. the second layer input) at position i takes the form

$$f_1(x_i) = E(x_i) + \mathbb{1}[x_{i-1} = \mathtt{w} \vee x_i = \mathtt{w}]e_7.$$

In layer 2, the unnormalized attention score computed for position i attending to j is

$$\begin{aligned}\left\langle \boldsymbol{W}_Q^{(2)^\top} f_1(x_i), \boldsymbol{W}_K^{(2)^\top} f_1(x_j) \right\rangle =& \frac{c}{T} \left\langle [1, 1], \left[\mathbb{1}[x_{j-1} = \mathtt{w} \vee x_j = \mathtt{w}], \frac{j}{T}\right] \right\rangle \\ =& c \cdot \left(\mathbb{1}[x_{j-1} = \mathtt{w} \vee x_j = \mathtt{w}] + \frac{j}{T}\right).\end{aligned}$$

Note that the max attention value is achieved at the position right after the closest $\mathtt{w}$ to x_i. Let us denote this position by $\tau \leq i$, then with hard attention, the output at the i_{th} position is $x_\tau e_1$, as desired. Now similar to before, we can approximate this with soft attention and use the MLP to do the appropriate rounding to get our final construction. □

Remark: The construction in Proposition 3 is *a* construction, but it is not the *only* construction. For example, for the second layer implementation for the flip-flop automaton, there could be an equally valid *dense* solution, where the model uniformly attends to all `write` tokens of the correct type.

3.5.7.2 Failure of soft attention: attention dilution with bounded Lipschitzness

Consider any attention layer with weight matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{k\times d}$. If $\|\boldsymbol{W}_K^\top \boldsymbol{W}_Q\|_2$ is bounded, then the attention cannot be sparse as the sequence length increases:

Proposition 4 (Leaky soft attention)**.** *Assume the latent variables have bounded norm, i.e. $\|\boldsymbol{v}\|_2 \leq 1$ for any latent vector $\boldsymbol{v} \in \mathbb{R}^d$, and let $\sigma_{\max}$ denote the max singular value of $\boldsymbol{W}_K^\top \boldsymbol{W}_Q$. Then for $T = \Omega(\exp(2\sigma_{\max}))$, any sequences of latent vectors $\{\boldsymbol{v}_\tau\}_{\tau\in[T]}$, $\|softmax(\{\boldsymbol{v}_\tau\}_{\tau\in[T]})\|_\infty = 1 - \Omega(1)$.*

Proof. The proof follows directly from a simple rewriting.

For any $\boldsymbol{u}, \boldsymbol{v}$ with $\|\boldsymbol{u}\|_2, \|\boldsymbol{v}\|_2 \leq 1$, the pre-softmax attention score is bounded by $\boldsymbol{u}^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v} \in [-\sigma_{\max}, \sigma_{\max}]$.

$$\frac{\exp(\boldsymbol{v}_t^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v}_T)}{\sum_{\tau\in[T]} \exp(\boldsymbol{v}_\tau^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v}_T)} \leq \frac{\exp(\sigma_{\max})}{\exp(\sigma_{\max}) + (T-1)\exp(-\sigma_{\max})} = 1 - \frac{T-1}{T-1+\exp(2\sigma_{\max})},$$

where the last term is $\Omega(1)$ when $T = \Omega(\exp(2\sigma))$. □

Attention dilution and failure on dense sequences Strictly speaking, attention dilution caused by an increased sequence length does not necessarily affect the output of the layer. For example, if `ignore` gets mapped to a subspace orthogonal to that of `write`, then $\boldsymbol{W}_V$ can project out the `ignore` subspace, making the weighted averaged depending only on the number of `writes`. Hence with the presence of layer norm, attention dilution won't be a problem for the final prediction if the number of `write` is upper bounded regardless of the sequence length.

Moreover, for the experiments in Section 3.5.4.1, denser sequences (i.e. larger $p($`write`$)$) does increase the number of `write` compared to the training distribution, hence attention dilution can be a potential cause for the decrease in performance.

3.5.7.3 Failure of hard attention: bad margin for positional embeddings

In this section, we look at a failure mode that a 1-layer 1-head Transformer has on the flip-flop automaton simulation task. Why do we care about this setup? Simulating the automaton is in fact a sub-task of FFLM. For example, the second layer of the construction in Proposition 3 reduces to the simulation task.

Consider a 1-layer 1-head Transformer with parameters $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{k\times d}$. Write the attention query matrix $\boldsymbol{W}_Q$ as $\boldsymbol{W}_Q = [\boldsymbol{W}_{Qe}, \boldsymbol{W}_{Qp}]$, where $\boldsymbol{W}_{Qe} \in \mathbb{R}^{k\times(d-1)}$ corresponds to the embedding dimensions, and $\boldsymbol{W}_{Qp}\mathbb{R}^k$ corresponds to the dimension for the linear positional encoding. Write $\boldsymbol{W}_K = [\boldsymbol{W}_{Ke}, \boldsymbol{W}_{Kp}]$ similarly.

Then, we claim that the following must be true, regardless of the choice of the token embedding:

Proposition 5. *Consider linear positional encoding, i.e. $p_i = i/C$ for some (large) constant C. Then, perfect length generalization to arbitrary length requires $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} = 0$.*

Proof. Let $\boldsymbol{e}^{(i)} \in \mathbb{R}^{d-1}$ denote the embedding vector (without the position encoding) for token $i \in \{0, 1, 2\}$. Let $\boldsymbol{v}_t = [\boldsymbol{e}_t, p_t]^\top \in \mathbb{R}^d$ denote the embedding for the t_{th} token, where $\boldsymbol{e}_t \in \{\boldsymbol{e}^{(0)}, \boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}\}\mathbb{R}^d$ is the embedding of the token itself, and $p_t := i/C$ is the linear positional encoding.

Let $s_{i\to j}$ denote the pre-softmax attention score that the i_{th} token puts on the j_{th} token, which is given by

$$s_{i\to j} = \langle \boldsymbol{W}_Q \boldsymbol{v}_i, \boldsymbol{W}_K \boldsymbol{v}_j \rangle \tag{3.7}$$

$$= \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}_j + \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp} \cdot p_j + (\boldsymbol{e}_j)^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp} \cdot p_i + \boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \cdot p_i p_j \tag{3.8}$$

$$= \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}_j + \frac{\boldsymbol{e}_i^\top \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} \cdot j + \frac{(\boldsymbol{e}_j)^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot i + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot ij. \tag{3.9}$$

We will prove the proposition in two cases, which respectively require $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$ and $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$.

Case 1: $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$ required Consider the case of long-term dependency, where the input sequence consists of an initial write and a series of reads, i.e. $\sigma_1 = 1$ and $\sigma_t = 0$ for $t > 1$. Then for the T_{th} position, the score for the first write token is

$$s_{T\to 1} = \langle \boldsymbol{W}_Q \boldsymbol{v}_T, \boldsymbol{W}_K \boldsymbol{v}_1 \rangle \tag{3.10}$$

$$= \boldsymbol{e}^{(0)^\top} \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}^{(1)} + \frac{\boldsymbol{e}^{(0)^\top} \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} + \frac{(\boldsymbol{e}^{(1)})^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot T + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T \tag{3.11}$$

$$= \left(\frac{(\boldsymbol{e}^{(1)})^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \right) \cdot T + O(1) = O(T), \tag{3.12}$$

and the score for the last write token is

$$s_{T\to T} = \langle \boldsymbol{W}_Q \boldsymbol{v}_T, \boldsymbol{W}_K \boldsymbol{v}_T \rangle \tag{3.13}$$

$$= \boldsymbol{e}^{(0)^\top} \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}^{(0)} + \frac{\boldsymbol{e}^{(0)^\top} \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} T + \frac{\boldsymbol{e}^{(0)^\top} \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot T + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 \tag{3.14}$$

$$= \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 + O(T). \tag{3.15}$$

Think of C as going to infinity. If $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} > 0$, then there exists a sufficiently large T such that $s_{T\to T} > s_{T\to 1}$. Hence we need $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$.

Case 2: $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$ required Consider the input sequence where $\sigma_1 = 1$, $\sigma_{T-1} = 2$, and $\sigma_t = 0$ for $t \in [T] \setminus \{1, T-1\}$. Similar to the above, calculate the pre-softmax attention scores for σ_1, σ_{T-1} as

$$s_{T\to 1} = O(T) \tag{3.16}$$

$$s_{T\to T-1} = \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 + O(T). \tag{3.17}$$

Since we need $s_{T\to T-1} > s_{T\to 1}$, it must be that $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$.

□

3.6 Application: implication on OOD performance and *myopic interpretability methods*

3.6.1 Setup and notation

Dyck languages A Dyck language [Schützenberger, 1963] is generated by a context-free grammar, where the valid strings consist of balanced brackets of different types (for example, "[()]" is valid but "([)]" is not). Dyck_k denote the Dyck language defined on k types of brackets. The alphabet of Dyck_k is denoted as $[2k] \equiv \{1, 2, \cdots, 2k\}$, where for each type $t \in [k]$, tokens $2t-1$ and $2t$ are a pair of corresponding open and closed brackets. Dyck languages can be recognized by a push-down automaton — by pushing open brackets onto a stack and and popping open brackets when it encounters matching closed brackets. For a string w and $i \leq j \in \mathbb{Z}_+$, we use $w_{i:j}$ to denote the substring of w between position i and position j (both ends included). For a valid prefix $w_{1:i}$, the *grammar depth* of $w_{1:i}$ is defined as the depth of the stack after processing $w_{1:i}$:

$$D(w_{1:i}) = \#\text{Open Brackets in } w_{1:i} - \#\text{Closed Brackets in } w_{1:i}.$$

We overload $D(w_{1:i})$ to also denote the grammar depth of the bracket at position i. For example, in each pair of matching brackets, the closing bracket is one depth smaller than the open bracket. We will use $\tau_{i,d}$ to denote a token of type $i \in [2k]$ placed at grammar depth $d \in \mathbb{N}$.

We consider *bounded-depth* Dyck languages following Yao et al. [2021a]. Specifically, $\mathsf{Dyck}_{k,D}$ is a subset of Dyck_k such that the depth of any prefix of a word is bounded by D,

$$\mathsf{Dyck}_{k,D} := \{w_{1:n} \in \mathsf{Dyck}_k \mid \max_{i \in [n]} D(w_{1:i}) \leq D\}. \tag{3.18}$$

While a bounded grammar depth might seem restrictive, it suffices to capture many practical settings. For example, the level of recursion occurring in natural languages is typically bounded by a small constant [Karlsson, 2007, Jin et al., 2018]. We further define the *length-N prefix set* of $\mathsf{Dyck}_{k,D}$ as

$$\mathsf{Dyck}_{k,D,N} = \{w_{1:N} \mid \exists n \geq N, w_{N+1:n} \in [2k]^{n-N}, s.t.\ w_{1:n} \in \mathsf{Dyck}_{k,D}\}. \tag{3.19}$$

Our theoretical setup uses the following data distribution $\mathcal{D}_{q,k,D,N}$:

Definition 12 (Dyck distribution)**.** *The distribution $\mathcal{D}_{q,k,D,N}$, specified by $q \in (0,1)$, is defined over $\mathsf{Dyck}_{k,D,N}$ such that $\forall w_{1:N} \in \mathsf{Dyck}_{k,D,N}$,*

$$\begin{aligned} p(w_{1:N}) \propto & (1/k)^{\#\{i \mid w_i \text{ is open, } D(w_{1:i})=1\}} \cdot (q/k)^{\#\{i \mid w_i \text{ is open, } D(w_{1:i})>1\}} \\ & \cdot (1-q)^{\#\{i \mid w_i \text{ is closed, } D(w_{1:i})<D-1\}}. \end{aligned} \tag{3.20}$$

That is, $q \in (0,1)$ denote the probability of seeing an open bracket at the next position, except for two corner cases: 1) the next bracket has to be open if the current grammar depth is 0 (1 after seeing the open bracket); 2) the next bracket has to be closed if the current grammar depth is D. Note that at any position, there is at most one valid closing bracket.

Training Objectives. Given a model f_θ parameterized by θ, we train with a *next-token prediction* language modeling objective on a given $\mathcal{D}_{q,k,D,N}$. Precisely, given a loss function $l(\cdot,\cdot) \to \mathbb{R}$, f_θ is trained to minimize the loss function $\min_\theta \mathcal{L}(\theta; \mathcal{D}_{q,k,D,N})$ with

$$\mathcal{L}(\theta; \mathcal{D}_{q,k,D,N}) = \mathbb{E}_{w_{1:N} \sim \mathcal{D}_{q,k,D,N}} \left[\frac{1}{N} \sum_{i=1}^{N} l(f_\theta(w_{1:i-1}), z(w_i)) \right], \tag{3.21}$$

in which $z(w_i) \in \{0,1\}^{2k}$ denotes the one-hot embedding of token w_i. We will omit the distribution $\mathcal{D}_{q,k,D,N}$ when it is clear from the context. We will also consider a ℓ_2-regularized version $\mathcal{L}^{\text{reg}}(\theta) = \mathcal{L}(\theta) + \lambda \frac{\|\theta\|_2^2}{2}$ with parameter $\lambda > 0$.

For our theory, we will consider the mean squared error as the loss function: [48]

$$l := l_{sq}(x, z_i) = \|x - z_i\|_2^2. \tag{3.22}$$

In our experiments, we apply the cross entropy loss following common practice.

Transformer Architecture. We consider a general formulation of Transformer in this work: the l-th layer is parameterized by $\theta^{(l)} := \{W_Q^{(l)}, W_K^{(l)}, W_V^{(l)}, \tau(\mathrm{g}^{(l)})\} \in \Theta$, where $W_K^{(l)}, W_Q^{(l)} \in \mathbb{R}^{m_a \times m}$, and $W_V^{(l)} \in \mathbb{R}^{m \times m}$ are the key, query, and value matrices of the attention module; $\tau(\mathrm{g}^{(l)})$ are parameters of a feed-forward network $\mathrm{g}^{(l)}$, consisting of fully connected layers, (optionally) LayerNorms and residual links. Given $X \in \mathbb{R}^{m \times N}$, the matrix of m-dimensional features on a length-N sequence, the l-th layer of a Transformer computes the function [49]

$$f_l(X; \theta^{(l)}) = \mathrm{g}^{(l)} \left(\mathrm{LN} \left(W_V^{(l)} X \underbrace{\sigma\left(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X) \right)}_{\text{attention pattern}} \right) + X \right), \tag{3.23}$$

where σ is the column-wise softmax operation defined as $\sigma(A)_{i,j} = \frac{\exp(A_{i,j})}{\sum_{k=1}^{N} \exp(A_{k,j})}$, $\mathcal{C}$ is the causal mask matrix defined as $\mathcal{C}_{i,j} = -\inf \cdot \mathbb{1}[i > j]$ where inf denotes infinity. We call $\sigma\left(\mathcal{C} + (W_K^{(l)} X)^\top (W_Q^{(l)} X)\right)$ the *Attention Pattern* of the Transformer layer l. LN represents column-wise LayerNorm operation, whose j_{th} output column is defined as:

$$\mathrm{LN}_{C_{LN}}(A)_{:,j} = \frac{\mathcal{P}_\perp A_{:,j}}{\max\{\|\mathcal{P}_\perp A_{:,j}\|_2, C_{LN}\}}, \mathcal{P}_\perp = \mathcal{I}_m - \frac{1}{m} \mathbf{1}\mathbf{1}^\top. \tag{3.24}$$

Here $\mathcal{P}_\perp$ denotes the projection orthogonal to the $\mathbf{1}\mathbf{1}^\top$ subspace [50] and C_{LN} is called the normalizing constant for LayerNorm.

[48]The challenge of applying our theory to cross-entropy loss is that for some prefixes, their grammatical immediate continuations strictly exclude certain tokens in the vocabulary (e.g. "]" cannot immediately follow "{"), so the optimal cross-entropy loss can only be attained if some parameters are set to infinity. However, when label smoothing is added, the optima is finite again, and analysis similar to ours could plausibly apply.

[49]Equation (3.23) is slightly different from the standard GPT architecture which places X within the layernorm. The two definitions perform similarly empirically, but Equation (3.23) is mathematically cleaner, mainly for the sufficiency part of Theorem 1: when the balance condition holds, the last column of the term $W_V^{(2)} X \sigma\left(\mathcal{C} + (W_K^{(2)} X)^\top (W_Q^{(2)} X)\right)$ will converge to zero when input length converges to infinity. Hence, if not all $\boldsymbol{e}(\tau_{t,d})$ where $\tau_{t,d}$ is a closed bracket aligns with 1^m, then it is impossible for the model to perfectly generate Dyck for arbitrary length.

[50]this is just a compact way to write the standard mean subtraction operation

We will further define the *attention output* at the l-th layer as

$$a_l(X;\theta^{(l)}) = W_V^{(l)} X \sigma\Big(\mathcal{C} + (W_K^{(l)}X)^\top (W_Q^{(l)}X)\Big). \tag{3.25}$$

When $C_{LN} = 0$, we will also consider the *unnormalized attention output* as

$$\tilde{a}_l(X;\theta^{(l)}) = W_V^{(l)} X \tilde{\sigma}\Big(\mathcal{C} + (W_K^{(l)}X)^\top (W_Q^{(l)}X)\Big). \tag{3.26}$$

where $\tilde{\sigma}(A)_{i,j} = \exp(A_{i,j})$ and it holds by definition that $\mathrm{LN}_0(\tilde{a}_l(X;\theta^{(l)})) = \mathrm{LN}_0(a_l(X;\theta^{(l)}))$.

An L-layer Transformer $\mathcal{T}_L$ consists of a composition of L of the above layers, along with a word embedding matrix $W_E \in \mathbb{R}^{m\times 2k}$ and a linear decoding head with weight $W_{\text{Head}} \in \mathbb{R}^{2k\times w}$. When inputting a sequence of tokens into Transformer, we will append a *starting token* $t_{\mathcal{S}}$ that is distinct from any token in the language at the beginning of the sequence. Let $\mathcal{Z} \in \mathbb{R}^{2k\times(N+1)}$ denote the one-hot embedding of a length-N sequence, then $\mathcal{T}_L$ computes for $\mathcal{Z}$ as

$$\mathcal{T}(\mathcal{Z}) = W_{\text{Head}}\Big[f_L(\cdots(f_1(W_E\mathcal{Z})))\Big]_{1:2k,(N+1)}. \tag{3.27}$$

3.6.2 Theoretical Analyses

Many prior works have looked for intuitive interpretations of Transformer solutions by studying the attention patterns of particular heads or some individual components of a Transformer [Clark et al., 2019b, Vig and Belinkov, 2019, Dar et al., 2022]. However, we show in this section why this methodology can be insufficient even for the simple setting of Dyck. Namely, for Transformers that generalize well on Dyck (both in-distribution and out-of-distribution), neither attention patterns nor individual local components are guaranteed to encode structures specific for parsing Dyck. We further argue that the converse is also insufficient: when a Transformer does produce interpretable attention patterns (suitably formalized), there could be limitations of such interpretation as well, as discussed in Section 3.6.5. Together, our results provide theoretical evidence that careful analyses (beyond heuristics) are required when interpreting the components of a learned Transformer.

3.6.2.1 Interpretability Requires Inspecting More Than Attention Patterns

This section focuses on Transformers with 2 layers, which are representationally sufficient for processing Dyck [Yao et al., 2021a]. We will show that even under this simplified setting, attention patterns alone are not sufficient for interpretation. In fact, we will further restrict the set of 2-layer Transformers by requiring the first-layer outputs to only depend on information necessary for processing Dyck:

Assumption 10 (Minimal First Layer)**.** *We consider 2-layer Transformers with a* minimal first layer f_1. *That is, if $\boldsymbol{Z} \in \mathbb{R}^{2k\times(N+1)}$ denotes the one-hot embeddings of an input sequence $t_{\mathcal{S}}, t_1, \dots, t_N \in [2k]$, then we assume the $(j+1)_{th}$ column of the output $f_1(W^E\boldsymbol{Z})$ only depends on the type and depth of t_j, $\forall j \in [N]$.*

Assumption 10 requires the first layer output to depend only on the bracket type and depth, disregarding any other information such as positions; an example of such a layer is given by Yao et al. [2021a]. The construction of a minimal first layer can vary, hence we *directly parameterize its output* instead:

Definition 13 (Minimal first layer embeddings)**.** *Given a minimal first layer, $\boldsymbol{e}(\tau_{t,d}) \in \mathbb{R}^m$ denotes its output embedding of $\tau_{t,d}$ for $t \in [2k]$, $d \in [D]$. $\boldsymbol{e}(t_{\mathcal{S}}) \in \mathbb{R}^m$ is the embedding of the starting token.*

Remark 3 (Simplicity as a feature (not a bug))**.** *It is important to note that while the minimal first layer is a strong condition, it does not weaken our results: We will show that the function class of 2-layer Transformers already contains a rich set of solutions, none of which are necessarily interpretable. Note that the simplicity of this architecture choice is intentional, since our theory on 2-layer Transformers directly implies similar conclusions for larger models, Intuitively, when moving to more complex architectures, the set of solutions can only grow and complicate interpretability further, hence our main conclusion still stands. For example, even though Theorem 1 and Theorem 2 are stated for 2-layer Transformers only, the constructed solutions can be trivially extended to multiple layers by e.g. letting the higher layers perform the identity function, or removing Assumption 10 and allowing the model to flexibly use or ignore positional information. More precisely:*

- *For Transformers with greater width, our Theorem 1 applies directly, since the construction does not depend on the width.*
- *For Transformers with greater depth, it suffices to show that additional layers can perform the identity function. To this end, one can utilize the residue link in the Transformer layer and choose the value matrix to be zero and the FFN (with or without residue connection) to be identity. This construction is implicitly assuming LayerNorm will map zero vector to zero vector, which is true for the common PyTorch implementation and for our paper. Also, it is worth noting that this holds for both the architecture we considered in the paper and the standard GPT-2 architecture.*

3.6.2.1.1 Perfect Balance Condition: Ideal Generalization of Unbounded Length

Some prior works have tried to understand the model by inspecting the attention patterns [Ebrahimi et al., 2020, Clark et al., 2019b, Vig and Belinkov, 2019]. However, we will show that the attention patterns alone are too flexible to be helpful, even for the restricted class of a 2-layer Transformer with a minimal first layer (Assumption 10) and even on a language as simple as Dyck. In particular, the Transformer only needs to satisfy what we call the *balanced condition*:

Definition 14 (Balance condition)**.** *A 2-layer Transformer (Equation (3.27)) with a minimal first layer (Assumption 10 and Definition 13) is said to satisfy the* balance condition*, if for any $i, j_1, j_2 \in [k]$ and $d', d_1, d_2 \in [D]$,*

$$\left(\boldsymbol{e}(\tau_{2i-1,d'}) - \boldsymbol{e}(\tau_{2i,d'-1})\right)^\top (W_K^{(2)})^\top W_Q^{(2)} \left(\boldsymbol{e}(\tau_{2j_1,d_1}) - \boldsymbol{e}(\tau_{2j_2,d_2})\right) = 0. \tag{3.28}$$

The following result shows that under minor conditions the balance condition is both necessary and sufficient:

Theorem 1 (Perfect Balance)**.** *Consider a two-layer Transformer $\mathcal{T}$ (Equation (3.27)) with a minimal first layer (Assumption 10) and $C_{LN} = 0$ (Equation (3.24)). Let $\mathcal{O}$ denote the* optimal prediction *scenario, that is, when the first layer embeddings $\{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}$ (Definition 13) and second layer parameters $\theta^{(2)}$ satisfy*

$$\theta := \{\boldsymbol{e}(\tau_{i,d})\}_{d\in[D],i\in[2k]}, \theta^{(2)}\} = \arg\min_{\tilde{\theta}} \mathcal{L}(\tilde{\theta}; \mathcal{D}_{q,k,D,N}), \forall N,$$

where the objective $\mathcal{L}$ is defined in Equation (3.21). Then,

- *Equation (3.28) is a necessary condition of $\mathcal{O}$, if $W_V^{(2)}$ satisfies $\mathcal{P}_\perp W_V^{(2)} \boldsymbol{e}(\tau_{t,d}) \neq 0, \forall t \in [k], d \in [D]$.*
- *Equation (3.28) is a sufficient condition of $\mathcal{O}$, for a construction in which the set of $2k+1$ encodings $\{\boldsymbol{e}(\tau_{2i-1,d}), \boldsymbol{e}(\tau_{2i,d})\}_{i\in[k]} \cup \{\boldsymbol{e}(t_{\mathcal{S}})\}$ are linearly independent for any $d \in [D]$ and the projection function $\mathrm{g}^{(2)}$ is a 6-layer MLP [51] with $O(k^2D^2)$ width.*

Remark: Recall from Equation (3.24) that $\mathcal{P}_\perp$ projects to the subspace orthogonal to $\mathbf{1}\mathbf{1}^\top$. The assumption in the "necessary condition" part of the theorem can be intuitively understood as requiring all tokens to have nonzero contributions to the prediction after the LayerNorm.

Recall that $\boldsymbol{e}(\tau_{2i-1,d'}), \boldsymbol{e}(\tau_{2i,d'-1})$ denote the first-layer outputs for a matching pair of brackets. Intuitively, Equation (3.28) says that since matching brackets should not affect future predictions, their embeddings should balance out each other. The balance condition Equation (3.28) is "perfect" in the sense that for the theorem, the model is required to minimize the loss for any length N; we will see an approximate version which relaxes this in Theorem 2.

Proof of the necessity of the balance condition. The key idea is reminiscent of the pumping lemma for regular languages. For any prefix p ending with a closed bracket $\tau_{2j,d}$ for $d \geq 1$ and containing brackets of all depths in $[D]$, let p_β be the prefix obtained by inserting β pairs of $\{\tau_{2i-1,d'}, \tau_{2i,d'-1}\}$ for arbitrary $i \in [k]$ and $d' \in [D]$. Denote the *projection of the unnormalized attention output* by

$$u(\tau_{t_1,d_1}, \tau_{t_2,d_2}) := \mathcal{P}_\perp \exp\left(\boldsymbol{e}(\tau_{t_1,d_1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{t_2,d_2})\right) W_V^{(2)} \boldsymbol{e}(\tau_{t_1,d_1}). \tag{3.29}$$

We ignored the normalization in softmax above, since the attention output will be normalized directly by LayerNorm according to Equation (3.23).

By Equation (3.23), for any $X \in \mathbb{R}^{m\times(N+1)}$ we have that

$$\tilde{a}_2(X;\theta^{(2)}) = \sum_{i=1}^{N+1} \mathcal{P}_\perp \exp\left(X_{1:m,i}^\top (W_K^{(2)})^\top W_Q^{(2)} X_{1:m,(N+1)}\right) W_V^{(2)} X_{1:m,(N+1)}.$$

Choosing X as the output of the first layer when the input is p_β, it holds that there exists a vector $v \in \mathbb{R}^m$ such that for any $\beta \in \mathbb{N}$, the next-token logits given by Transformer $\mathcal{T}$ are

$$\mathcal{T}(p_\beta) = W_{\text{Head}} g^{(2)} \left(\frac{v + \beta\left(u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'})\right)}{\left\| v + \beta\left(u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'})\right)\right\|_2} + \boldsymbol{e}(\tau_{2j,d}) \right). \tag{3.30}$$

The proof proceeds by showing a contradiction. Suppose $u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}) \neq 0$. Based on the continuity of the projection function and the LayerNorm Layer, we can show that $\lim_{\beta\to\infty} \mathcal{T}(p_\beta)$ depend only on the depths d, d' and types $2j, 2i-1, 2i$. However, these are not sufficient to determine the next-token probability from p_β, since the latter depends on the type of the last unmatched open bracket in p. This contradicts the assumption that the model can minimize the loss for any length N. Hence we must have

$$u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}) = 0. \tag{3.31}$$

[51] In the construction, we first use 4 layers to convert the input of the projection function to a triplet indicating the type and depth of the last token and the type of the last unmatched bracket when the last token is a closed bracket. We then use another 2 layers to predict the next token probability based on the triplet. This construction is likely improvable.

Finally, as we assumed that $\mathcal{P}_\perp W_V^{(2)} e\left(\tau_{t,d}\right) \neq 0$, we conclude that

$$\left(e\left(\tau_{2i-1,d'}\right) - e\left(\tau_{2i,d'-1}\right)\right)^\top (W_K^{(2)})^\top W_Q^{(2)} e\left(\tau_{2j+1,d}\right) = \ln\left(\frac{\|\mathcal{P}_\perp W_V e\left(\tau_{2i,d'-1}\right)\|_2}{\|\mathcal{P}_\perp W_V e\left(\tau_{2i-1,d'}\right)\|_2}\right),$$

where the right hand side is independent of j, d, concluding the proof for necessity. The proof of sufficiency are given in Appendix 3.6.4.1. □

Note that the perfect balance condition is an orthogonal consideration to interpretability. For example, even the uniform attention satisfies the condition and can solve Dyck: [52]

Corollary 1. *There exists a 2-layer Transformer with uniform attention and no positional embedding (but with causal mask and a starting token* [53] *) that generates the Dyck language of arbitrary length.*

Proof. We will first construct a uniform attention first layer that can generate the embedding in Equation (3.37). Suppose Z is the one-hot embeddings of a prefix p of length n, where each token of type t for $t \in [2k]$ is encoded as $\boldsymbol{o}_t$ and the starting token is encoded as $\boldsymbol{o}_{2k+1}$. Then it holds that

$$\left[Z\sigma\Big(\mathcal{C}\cdot (W_K^{(1)}Z)^\top (W_Q^{(1)}Z)\Big)\right]_{:,n+1} = \sum_{i=1}^{2k} \#\{\text{token of type } t \text{ in } p\}\boldsymbol{o}_t + \boldsymbol{o}_{2k+1}. \tag{3.32}$$

Then we can choose $W_V^{(1)}$ such that for $x \in \mathbb{R}^{2k+1}$,

$$\begin{aligned}(W_V^{(1)}x)_1 =& \sum_{i=1}^{k} x_{2i-1} - x_{2i},\\ (W_V^{(1)}x)_2 =& x_{2k+1},\\ (W_V^{(1)}x)_i =& 0, \forall i \geq 3.\end{aligned}$$

Hence it holds That

$$\left[W_V^{(1)}Z\sigma\Big(\mathcal{C}\cdot (W_K^{(1)}Z)^\top (W_Q^{(1)}Z)\Big)\right]_{:,n+1} = \#\{\text{depth of } p_n\}\boldsymbol{o}_1 + \boldsymbol{o}_2.$$

It is then easy to check $\mathrm{LN}\left(\left[W_V^{(1)}Z\sigma\Big(\mathcal{C}\cdot (W_K^{(1)}Z)^\top (W_Q^{(1)}Z)\Big)\right]_{:,n+1}\right) + Z_{:,n+1}$ is uniquely determined by the type and depth of p_n without repetition. Then by Lemma 38, there exists a 2-layer ReLU MLP with width $O(k^2D^2)$ that can map $\mathrm{LN}\left(\left[W_V^{(1)}Z\sigma\Big(\mathcal{C}\cdot (W_K^{(1)}Z)^\top (W_Q^{(1)}Z)\Big)\right]_{:,n+1}\right) + Z_{:,n+1}$ to the embedding in Equation (3.37). It is then easy to see that the condition in Theorem 1 is satisfied as $W_K^{(2)} = W_Q^{(2)} = 0$. Hence the second layer can be constructed to let the Transformer to output the correct next token probability. □

Since uniform attention patterns are hardly reflective of any structure of Dyck, Corollary 1 proves that attention patterns can be oblivious about the underlying task, violating the "faithfulness" criteria for an interpretation [Jain and Wallace, 2019]. We will further show in Section 3.6.5 that empirically, seemingly structured attention patterns may not accurately represent the natural structure of the task.

[52] This is verified empirically: the uniform-attention models have attention weights fix to 0 and are to fit the distribution almost perfectly ($>$ 99% accuracy).

[53] Here the starting token is necessary because otherwise, the Transformer with uniform attention will have the same outputs for prefix p and prefix $p \oplus p$, in which $\oplus$ denotes concatenation, i.e. $p \oplus p$ means the same string p repeated twice.

3.6.2.1.2 Approximate Balance Condition For Finite Length Training Data

Theorem 1 assumes the model reaches the optimal loss for Dyck prefixes of any length. However, in practice, due to finite samples and various sources of randomness, training often does not end exactly at a population optima. In this case, the condition in Theorem 1 is not precisely met. However, even for models that *approximately* meet those conditions, we will prove that when the second-layer projection function $g^{(2)}$ is Lipschitz, a similar condition as in Equation (3.31) is still necessary.

We will show this by bounding the amount of deviations from the perfect balance. The idea is that for two long prefixes that differ in only the last open bracket, correct next token prediction requires the Transformer outputs on these prefixes to be sufficiently different, hence the part irrelevant to the prediction (i.e. matched brackets) should not have a large contribution.

To formalize this intuition, we define two quantities: 1) $S_{d,d',i,j}$ which measures the effect from one matching pair, and 2) $P_{d,j}$ which measures the effect on the last position from all tokens in a prefix.

Let u be defined as in Equation (3.29). $S_{d,d',i,j}$ is defined as

$$S_{d,d',i,j}[\theta^{(2)}] = u(\tau_{2j,d}, \tau_{2i,d'-1}) + u(\tau_{2j,d}, \tau_{2i-1,d'}), \tag{3.33}$$

which measures how much a matching pair of brackets $(\tau_{2i,d'-1}, \tau_{2i-1,d'})$ changes the input to the LayerNorm upon seeing the last token $\tau_{2j,d}$. Note that under the perfect balance condition, $S_{d,d',i,j}[\theta^{(2)}] = 0$ by Equation (3.31).

The second quantity $P_{d,j}[\theta^{(2)}]$ is defined via an intermediate quantity $Q(2j, d, \tilde{\boldsymbol{t}})$: for any $i \in [k], d \in [D]$ and a length-$(d-1)$ prefix $\tilde{\boldsymbol{t}} \in [2k]^{d-1}$, $Q(i, d, \tilde{\boldsymbol{t}})$ is defined as

$$\begin{aligned} Q(i, d, \tilde{\boldsymbol{t}}) := u(\tau_{2i,d-1}, t_{\mathcal{S}}) + \sum_{1 \le d' < d} u(\tau_{2i,d-1}, \tau_{\tilde{\boldsymbol{t}}_{d'},d'}) \\ + u(\tau_{2i,d-1}, \tau_{2i-1,d}) + u(\tau_{2i,d-1}, \tau_{2i,d-1}), \end{aligned} \tag{3.34}$$

where $\tilde{\boldsymbol{t}}_{d'}$ denotes the d'_{th} entry of $\tilde{\boldsymbol{t}}$. Intuitively, $Q(i, d, \tilde{\boldsymbol{t}})$ denotes the unnormalized second-layer attention output at the last position, given the input sequence $\tilde{\boldsymbol{t}} \oplus \tau_{2i-1,d}\tau_{2i,d-1}$, [54]

For results in this subsection, it suffices to consider prefixes consisting only of open brackets. Let $\boldsymbol{t} := \arg\min_{\tilde{\boldsymbol{t}} \in \{2i-1\}_{i \in [k]}^{d-1}} \|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$, and let $\boldsymbol{t}'$ denote the prefix that minimizes $\|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$ subject to the constraint that $\boldsymbol{t}'$ differs from $\boldsymbol{t}$ at the last (i.e. $(d-1)_{th}$) position, i.e.

$$\boldsymbol{t}' = \arg \min_{\tilde{\boldsymbol{t}}' \in \{2i-1\}_{i \in [k]}^{d-1}, \boldsymbol{t}'_{d-1} \neq \boldsymbol{t}_{d-1}} Q(2j, d, \tilde{\boldsymbol{t}}').$$

Such choices of $\boldsymbol{t}, \boldsymbol{t}'$ guarantees that the two prefixes differ at the last open bracket and hence must have different next-word distributions. Finally, define

$$P_{d,j}[\bar{\theta}^{(2)}] = \|Q(2j, d, \boldsymbol{t}')\|_2. \tag{3.35}$$

In the following theorem, $P_{d,j}$ will be used as a quantity that will denote an upper bound on $S_{d,d',i,j}[\theta^{(2)}]$, meaning that the model should not be sensitive to the insertion of a matching pair of brackets.

[54] We use $s \oplus t$ to denote the concatenation of two strings s, t, same as in Equation (3.37)-(3.38), and use $\tau_i\tau_j$ to denote the concatenation of two tokens τ_i, τ_j.

Theorem 2 (Approximate Balance). *Consider a 2-layer Transformer $\mathcal{T}$ (Equation (3.27)) with a minimal first layer (Assumption 10) and a γ-Lipschitz $\mathrm{g}^{(2)}$ for $\gamma > 0$, trained on sequences of length N with the mean squared loss (Equation (3.22)).*

Suppose the loss is approximately optimal, precisely, the set of second-layer weights $\bar{\theta}_N^{(2)}$ satisfies $\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{q,k,D,N}) \leq (\frac{q(1-q)}{k^2})^N \epsilon$, for every positive integer $N > 8D$ and sufficiently small $\epsilon > 0$. Then, there exists a constant $C_{\gamma,\epsilon,D}$, such that $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$, it holds that

$$\|S_{d,d',i,j}[\bar{\theta}_N^{(2)}]\| \leq \frac{C_{\gamma,\epsilon,D}}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{3.36}$$

Intuitively, Theorem 2 states that when the loss $\mathcal{L}(\theta)$ is sufficiently small, $S_{d,d',i,j}[\theta^{(2)}]$ must be small relative to $P_{d,j}[\bar{\theta}_N^{(2)}]$. Inequality 3.36 can be interpreted as a relaxation of Equation (3.31), which is equivalent to $S_{d,d',i,j}[\theta^{(2)}] = 0$. The proof of Theorem 2 shares a similar intuition as Theorem 1 and is given in Section 3.6.4.2.

A direct corollary of Theorem 2 additionally considers weight decay as well, in which case approximate balance condition still holds, as the regularization strength goes to 0:

Corollary 2. *Consider the setting where a Transformer with a fixed minimal first layer is trained to minimize $\mathcal{L}_\lambda^{reg} = \mathcal{L}_\theta(x) + \lambda \frac{\|\theta\|_2^2}{2}$, which is the squared loss with λ weight decay. Suppose $\mathrm{g}^{(1)}$ of the Transformer is a 2-layer fully connected network and $\mathrm{g}^{(2)}$ of the Transformer is a 6-layer fully connected network. Then, there exists constant $C > 0$, such that if a set of parameters $\theta_{\lambda,N}$ minimizes $\mathcal{L}_\lambda^{reg}$, then it holds $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$ that,*

$$\forall N, \exists \lambda_N, \text{ such that } \forall \lambda \in [0, \lambda_N], S_{d,d',i,j}[\theta_{\lambda,N}^{(2)}] \leq \frac{C}{N} P_{d,i}[\theta_{\lambda,N}^{(2)}].$$

Proof. This proof is in fact a direct combination of Theorems 1 and 2. By Theorem 1 we know there exists a weight $\theta^{(2)*}$ that can reach zero loss for arbitrarily length N. Then it holds that $\|\theta_{\lambda,N}\|_2 \leq \|\theta^{(2)*}\|$ as $\theta_{\lambda,N}$ minimizes the regularized loss. Noticing that bounded weight implies bounded Lipschitzness of $\mathrm{g}^{(2)}$, the rest follows as Theorem 2. □

Remark 4 (Extension to approximate balance condition). *Theorem 1 assumes the model reaches the optimal loss for Dyck prefixes of any length. However, in practice, due to finite samples and various sources of randomness, training often does not end exactly at a population optima. In this case, the condition in Theorem 1 is not precisely met. However, even for models that* approximately *meet those conditions, we will prove that when the second-layer projection function $\mathrm{g}^{(2)}$ is Lipschitz, a similar condition as in Equation (3.31) is still necessary. Details are deferred to Section 3.6.2.1.2.*

3.6.2.2 Interpretability Requires Inspecting More Than Any Single Weight Matrix

Another line of interpretability works involves inspecting the weight matrices of the model [Li et al., 2016, Dar et al., 2022, Eldan and Li, 2023]. Some of the investigations are done locally, neglecting the interplay between different parts of the model. Our result in this section shows that from a representational perspective, isolating single weights can also be misleading for interpretability. For this section only, we will assume the linear head W_{Head} is identity for simplicity. To consider the effect of pruning, we will also extend the parameterization of LayerNorm module (Equation (3.24)) as

$$\mathrm{LN}_{C_{LN}}[b](A)_{:,j} = b\frac{\mathcal{P}_\perp A_{:,j}}{\max\{\|\mathcal{P}_\perp A_{:,j}\|_2, \epsilon\}} + (1-b)A_{:,j},$$

which corresponds to a weighted residual branch; note that the original LayerNorm corresponds to $\mathrm{LN}_C[1]$. [55] Let $\hat{\theta}$ denote the set of parameters of this extended parameterization.

We define the *nonstructural pruning* [56] as:

Definition 15 (Nonstructural pruning)**.** *Under the extended parameterization, a* nonstructural pruning *of a Transformer with parameters $\hat{\theta}$ is a Transformer with the same architecture and parameters $\hat{\theta}'$, so that for any weight matrix W in $\hat{\theta}$, the corresponding matrix W' in $\hat{\theta}'$ satisfies $W'_{i,j} \in \{W_{i,j}, 0\}, \forall i, j$.*

To measure the quality of the pruning, define the ϵ-approximation:

Definition 16 (ϵ-approximation)**.** *Given two metric spaces A, B with the same metric $\|\cdot\|$, a function $f : A \to B$ is an ϵ-approximation of function g with respect to that metric, if and only if,*

$$\forall x \in A, \|f(x) - g(x)\| \le \epsilon\|x\|.$$

The metric, unless otherwise specified, will be the 2-norm for vectors and the 1, 2-norm for matrices:

Definition 17. *The* 1, 2*-norm of a matrix A is the max row norm, i.e. $\|A\|_{1,2} = \max_{i\in[d']} \|A_{:,i}\|_2$.*

With these definitions, we are ready to state the main result of this section:

Theorem 3 (Indistinguishability From a Single Component)**.** *Consider any L-layer Transformer $\mathcal{T}$ (Equation (3.27)) with embedding dimension m, attention dimension m_a, and projection function $\mathrm{g}^{(l)}$ as 2-layer ReLU MLP with width w, for $l \in [L]$.* [57] *For any $\delta \in (0, 1)$ and $N \in \mathbb{N}^+$, consider a $4L$-layer* random *Transformer $\mathcal{T}_{large}$ with embedding dimension $m_{\text{large}} = O(m \log(Lm/\delta))$, attention dimension $m_{\text{large},a} = O(m_a L \log \frac{m_a m L N}{\epsilon\delta})$, and projection function $\mathrm{g}_{\text{large}}$ as 4-layer ReLU MLP with width $w_{\text{large}} = O(\max\{m, w\} L \log \frac{wmLN}{\epsilon\delta})$.*

Assume that $\|W\|_2 \le 1$ for every weight matrix W in $\mathcal{T}$, and suppose the weights are randomly sampled as $W_{i,j} \sim U(-1, 1)$ for every $W \in \mathcal{T}_{large}$. Then, with probability $1-\delta$ over the randomness of $\mathcal{T}_{large}$, there exists a nonstructural pruning (Definition 15) of $\mathcal{T}_{large}$, denoted as $\tilde{\mathcal{T}}_{large}$, which ϵ-approximates $\mathcal{T}$ with respect to $\|\cdot\|_{1,2}$ for any input $X \in \mathbb{R}^{m\times N}$ satisfying $\|\boldsymbol{X}\|_{1,2} \le 1$. [58]

Proof sketch: connection to Lottery Tickets. Theorem 3 can be interpreted as a lottery ticket hypothesis [Frankle and Carbin, 2018, Malach et al., 2020] for randomly initialized Transformers, which can be of independent interest. The proof repeatedly uses an extension of Theorem 1 of Pensia et al. [2020], where it 1) first prunes the $(2l-1)$-th and $2l$-th layers of $\mathcal{T}_{\text{large}}$ to approximate $\mathcal{T}^{(l)}$ for each $l \in [L]$ (Lemma 29), and 2) then prunes the remaining $2L+1$ to $4L$-th layers of $\mathcal{T}_{\text{large}}$ to approximate the identity function. The full proof is deferred to Section 3.6.4.3.

[55] This residue link is added for the ease of proof because it is hard to "undo" a LayerNorm. We also note that in standard architecture like GPT-2, there is typically a residual link after LayerNorm similar to here.

[56] This is as opposed to *structural pruning*, which prunes entire rows/columns of weight matrices.

[57] For notational convenience, we assume all layers share the same dimensions and projection functions. The proof can be trivially extended to cases where the dimensions and projection functions are different.

[58] Here the input and output dimension of $\tilde{\mathcal{T}}_{\text{large}}$ is actually m_{large} which is larger than m; additional dimensions are padded with zeroes. The norm constraint can be easily extended to an arbitrary constant.

Noting that the layers used to approximate the identity can appear at arbitrary depth in $\mathcal{T}_{\text{large}}$, a direct corollary of Theorem 3 is that one cannot distinguish between two functionally different Transformers by inspecting any single weight matrix only:

Corollary 3. *Let $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_{large}$ follow the same definition and assumptions as $\mathcal{T}$ and $\mathcal{T}_{large}$ in Theorem 3. Pick any weight matrix W in $\mathcal{T}_{large}$, then with probability $1-\delta$ over the randomness of $\mathcal{T}_{large}$, there exist two Transformers $\mathcal{T}_{Large,1}, \mathcal{T}_{Large,2}$ pruned from $\mathcal{T}_{large}$, such that $\mathcal{T}_{Large,i}$ ϵ-approximate $\mathcal{T}_i$, $\forall i \in \{1,2\}$, and $\mathcal{T}_{Large,1}$, $\mathcal{T}_{Large,2}$ coincide on the pruned versions of W.*

Hence, one should be cautious when using methods based solely on individual components to interpret the overall function of a Transformer.

3.6.3 Experiments

Our theory in Section 3.6.2 proves the existence of abundant *non-stack-like* attention patterns, all of which suffice for (near-)optimal generalization on Dyck. However, could it be that stack-like solutions are more frequently discovered empirically, due to potential *implicit biases* in the architecture and the training procedure? In this section, we show there is no evidence for such implicit bias in standard training (Section 3.6.3.1). Additionally, we propose a regularization term based on the balance condition (Theorem 1), which leads to better length generalization (Section 3.6.3.2).

Training Details For Figure 3.3, we train 2-layer standard GPT on $\text{Dyck}_{2,4}$ with sequence length no longer than 28. For (a), we train with hidden dimension and network width 200 and learning rate 3e-4. For $(b), (c), (d)$, we train with hidden dimension and FFN width 50 and learning rate 3e-3.

For Figure 3.39, for (a), we train 1-layer transformer without residual link, FFN and the final LayerNorm before the linear head. The hidden dimensions and FFN widths are fixed as 500. For (a), we train the network with learning rate 1e-2 and for $(b), (c), (d)$ we train the network with learning rate 3e-3.

3.6.3.1 Different Attention Patterns Can Be Learned To Generate Dyck

We empirically verify our theoretical findings that Dyck solutions can give rise to a variety of attention patterns, by evaluating the accuracy of predicting the last bracket of a prefix (Equation 3.19) given the rest of the prefix. We only consider prefixes ending with a closing bracket, so that there exists a unique correct closing bracket which a correct parser should be able to determine. The experiments in this section are based on Transformers with 2 layers and 1 head, hidden dimension 50 and embedding dimension 50, trained using Adam. The training data consists of valid $\text{Dyck}_{2,4}$ sequences of length less than 28 generated with $q = 0.5$. When tested in-distribution, all models are able to achieve $\geq 97\%$ accuracy.

Variation in attention patterns First, as a response to (Q1), we observe that attention patterns of Transformers trained on Dyck are not always stack-like (Figure 3.3). In fact, the attention patterns differ even across different random initialization. Moreover, while Theorem 1 implies that position encoding is not necessary for a Transformer to generate Dyck, [59] adding the position encoding [60] does affect the attention patterns (Figures 3.3c and 3.3d).

Specifically, for 2-layer Transformers with a minimal first layer, we experiment with three different types of embeddings $\boldsymbol{e}$: let $\boldsymbol{o}_t$ denote the one-hot embedding where $\boldsymbol{o}_t[t] = 1$,

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_{(t-1)D+d} \in \mathbb{R}^{2kD}, \tag{3.37}$$

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_t \oplus \boldsymbol{o}_d \in \mathbb{R}^{2k+D}, \tag{3.38}$$

$$\boldsymbol{e}(\tau_{t,d}) = \boldsymbol{o}_t \oplus [\cos(\theta_d), \sin(\theta_d)] \in \mathbb{R}^{2k+2}, \theta_d = \arctan(d/(D+2-d)), \tag{3.39}$$

where $\oplus$ denotes vector concatenation. Equation (3.37) is the standard one-hot embedding for $\tau_{t,d}$; Equation (3.38) is the concatenation of one-hot embedding of types and depths. Finally, Equation (3.39) is the embedding constructed in Yao et al. [2021a]. As shown in Figure 3.39, the attention patterns learned by Transformers exhibit large variance between different choices of architectures and learning rates, and most learned attention patterns are not stack-like.

Quantifying the variation We now quantify the variation in attention by comparing across multiple random initializations. We define the *attention variation* between two attention patterns A_1, A_2 as $\text{Variation}(A_1, A_2) = \|A_1 - A_2\|_F^2$, for $A_1, A_2 \in \mathbb{R}^{N\times N}$ over an length-N input sequence. We report the *average attention variation* of each architecture based on 40 random initializations.

Let's consider a special prefix of $[[[[]]]](((())))$, which contains brackets of all types and depths. We observe that for standard two layer training, the average attention variation is 2.20 with linear position embedding, and is 2.27 without position embedding. Both numbers are close to the random baseline value of 2.85 [61], showing that the attention head learned by different initializations indeed tend to be very different. We also experiment with Transformer with a minimal first layer and the embedding in Equation (3.37), where the average variation is reduced to 0.24. We hypothesize that the structural constraints in this setting provide sufficiently strong inductive bias that limit the variation.

3.6.3.2 Guiding The Transformer To Learn Balanced Attention

In our experiments, we observe that although models learned via standard training that can generalize well in distribution, the *length generalization* performance is far from optimal. This implies that the models do not correctly identify the parsing algorithm for Dyck when learning from finite samples. A natural question is: can we guide Transformers towards correct algorithms, as evidenced by improved generalization performance on longer Dyck sequences?

In the following, we measure length generalization performance by the model accuracy on valid Dyck prefixes with length randomly sampled from 400 to 500, which corresponds to around 16 times the length of the training sequences. Inspired by results in Section 3.6.2, we propose a regularization term to encourage more balanced attentions, which leads to better length generalization.

[59] This is verified empirically, as Transformers with no positional encoding achieve $\geq 97\%$ accuracy.

[60] We use the linear positional encoding following Yao et al. [2021a]: for the i_{th} position, the encoding is defined to be $\boldsymbol{e}_p(i) := i/T_{\max}$ for some $T_{\max}$.

[61] The random baseline is calculated by generating purely random attention patterns (from the simplex, i.e. random square matrices s.t. each row sums up to 1) and calculate the average attention variation between them.

Regularizing for balance violation improves length generalization accuracy We denote the *balance violation* of a Transformer as $\beta := \mathbb{E}_{d,d',i,j}\left[S_{d,d',i,j}/P_{d,j}\right]$ for S, P defined in Equations (3.33) and (3.35). Theorem 1 predicts that for models with a minimal first layer, perfect length generalization requires β to be zero. Inspired by this observation, we design a contrastive training objective to reduce the balance violation, which ideally would lead to improved length generalization. Specifically, let p_r denote a prefix of r nested pairs of brackets of for $r \sim U([D])$, and let $\mathcal{T}(s \mid p_r \oplus s)$ denote the logits for s when $\mathcal{T}$ takes as input the concatenation of p_r and s. We define the *contrastive regularization term* $R_{\text{contrastive}}(s)$ as the mean squared error between the logits of $\mathcal{T}(s)$ and $\mathcal{T}(s \mid p_r \oplus s)$, taking expectation over r and p_r:

$$\mathbb{E}_{r\sim U([D]),p_r}\left[\|\mathcal{T}(s \mid p_r \oplus s) - \mathcal{T}(s)\|_F^2\right]. \tag{3.40}$$

Following the same intuition as in the proof of Theorem 1, if the model can perfectly length-generalize, then the contrastive loss will be zero. Models trained with contrastive loss show reduced balance violation as well as improved length generalization performance, as shown in Figure 3.40.

3.6.4 Proofs

3.6.4.1 Proof of Theorem 1: perfect balance

The key step was outlined in Section 3.6.2. We will restate the proof rigorously here.

Theorem 1 (Perfect Balance)**.** *Consider a two-layer Transformer $\mathcal{T}$ (Equation (3.27)) with a minimal first layer (Assumption 10) and $C_{LN} = 0$ (Equation (3.24)). Let $\mathcal{O}$ denote the* optimal prediction *scenario, that is, when the first layer embeddings $\{e(\tau_{i,d})\}_{d\in[D],i\in[2k]}$ (Definition 13) and second layer parameters $\theta^{(2)}$ satisfy*

$$\theta := \{\{e(\tau_{i,d})\}_{d\in[D],i\in[2k]}, \theta^{(2)}\} = \arg\min_{\tilde{\theta}} \mathcal{L}(\tilde{\theta}; \mathcal{D}_{q,k,D,N}), \forall N,$$

where the objective $\mathcal{L}$ is defined in Equation (3.21). Then,

- *Equation (3.28) is a necessary condition of $\mathcal{O}$, if $W_V^{(2)}$ satisfies $\mathcal{P}_\perp W_V^{(2)} e(\tau_{t,d}) \neq 0, \forall t \in [k], d \in [D]$.*
- *Equation (3.28) is a sufficient condition of $\mathcal{O}$, for a construction in which the set of $2k+1$ encodings $\{e(\tau_{2i-1,d}), e(\tau_{2i,d})\}_{i\in[k]} \cup \{e(t_{\mathcal{S}})\}$ are linearly independent for any $d \in [D]$ and the projection function $g^{(2)}$ is a 6-layer MLP [62] with $O(k^2D^2)$ width.*

Proof. We prove the **sufficiency of the balanced condition** below; the proof for the *necessity* has been given in Section 3.6.2.1.

We will denote the dimension of $e(\tau_{t,d})$ as m.

For any $i \in [k], d' \in [D]$, by Equation (3.28), we can assume that there exists $a_{i,d'} \in \mathbb{R}$ such that for all $j \in [k], d \in [D]$, it holds that,

$$a_{i,d'} \triangleq \left(e\left(\tau_{2i-1,d'}\right) - e\left(\tau_{2i,d'-1}\right)\right)^\top (W_K^{(2)})^\top W_Q^{(2)} e\left(\tau_{2j,d}\right). \tag{3.41}$$

[62] In the construction, we first use 4 layers to convert the input of the projection function to a triplet indicating the type and depth of the last token and the type of the last unmatched bracket when the last token is a closed bracket. We then use another 2 layers to predict the next token probability based on the triplet. This construction is likely improvable.

We will first define the possible index sets of $\tau_{t,d}$ as $\mathcal{I} = \{(2t, d) \mid t \in [k], 0 \ \leq d \leq D - 1\} \cup \{(2t - 1, d) \mid t \in [k], 1 \leq d \leq D\}$, and we will define the rank of (t, d) as

$$r(t, d) \triangleq \#\{(t_1, d_1) \mid t_1 < t \text{ or } t_1 = t, d_1 \leq d, (t_1, d_1) \in \mathcal{I}\} \tag{3.42}$$

Then it is clear that $r(t, d)$ is a one-to-one mapping from $\mathcal{I}$ to $[2kD]$. We will then define the collection of all $\boldsymbol{e}(\tau_{t,d})$ as $\boldsymbol{E}$, satisfying that $\boldsymbol{E}_{:,r(t,d)} = \boldsymbol{e}(\tau_{t,d}), \boldsymbol{E}_{:,2kD+1} = \boldsymbol{e}(t_{\mathcal{S}})$.

Because $\boldsymbol{e}(\tau_{t,d})$ are linearly independent, for any $(i, d) \neq (j, d') \in \mathcal{I}$, it holds that $\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'}) \neq 0$. Then based on Lemma 39, there exists a set of orthonormal vectors $\{\mathrm{b}_i\}_{i \in [m-2]}$, such that for any $(i, d), (j, d') \in \mathcal{I}$, it holds that

$$\sum_{i=1}^{m-2} \mathrm{b}_i \mathrm{b}_i^\top \left(\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'})\right) \neq (\boldsymbol{e}(\tau_{i,d}) - \boldsymbol{e}(\tau_{j,d'}) \tag{3.43}$$

$$\mathrm{b}_i^\top 1^m = 0 \tag{3.44}$$

We will further construct the matrix $\boldsymbol{O}$ as [63]

$$\begin{aligned} \boldsymbol{O}_{:,r(2t,d-1)} &= -\exp(a_{t,d}) \mathrm{b}_{tD+d}, \\ \boldsymbol{O}_{:,r(2t-1,d)} &= \mathrm{b}_{tD+d}. \\ \boldsymbol{O}_{:,2kD+1} &= 0. \end{aligned} \tag{3.45}$$

for $t \in [k], d \in [D]$.

We can then choose $W_V^{(2)} \in \mathbb{R}^{m \times m}$ such that

$$W_V^{(2)} \boldsymbol{E} = \boldsymbol{O} \tag{3.46}$$

Such $W_V^{(2)}$ is guaranteed to exist, because $\boldsymbol{E}$ is of full column rank by the linear independence assumption.

Now based on this construction, we will show that the last column of unnormalized attention output (Equation (3.26)) depends only on the sequence of unmatched brackets when the last token is a closed bracket with depth d greater than or equal to 1. [64]

For any valid Dyck prefix p of length n ending with a closed bracket $\tau_{2j,d}$ satisfying $d \geq 1$, suppose the list of unmatched open brackets in p is $[\tau_{2j_1-1,1}, \tau_{2j_2-1,2}, \ldots, \tau_{2j_d-1,d}]$. Then, the remaining tokens in p are pairs of matching brackets. Denote them by $\tau_{2t_k-1,d_k}, \tau_{2t_k,d_k-1}$ for $k \in [K]$. Then the input of the second layer of Transformer X, up to a permutation is

$$XP = [\boldsymbol{e}(\tau_{2t_1-1,d_1}), \boldsymbol{e}(\tau_{2t_1,d_1-1}), \ldots, \boldsymbol{e}(\tau_{2t_K-1,d_K}), \boldsymbol{e}(\tau_{2t_K,d_K-1}), \boldsymbol{e}(\tau_{2j_1-1,1}), \ldots \boldsymbol{e}(\tau_{2j_d-1,d}), \boldsymbol{e}(t_{\mathcal{S}})].$$

[63] Recall the definition of r in Equation (3.42). Comparing $\boldsymbol{O}_{:,r(2t,d-1)}$ and $\boldsymbol{O}_{:,r(2t-1,d)}$: the idea is that a pair of matched brackets are represented by the same direction (i.e. the direction along b_{tD+d}), just with different norms.

[64] When depth $d = 0$, all brackets are matched, the groundtruth next-token distribution is the prior distribution over the open brackets. Because in Equation (3.28) $d_1, d_2 \geq 1$, we handle the depth $d = 0$ case separately in Case 2 "t is even, $d = 0$" towards the end of this proof. In the following, we focus on cases with depth $d \geq 1$.

We will focus on the last column of the unnormalized attention output

$$\begin{aligned}
\tilde{a}_2(X;\theta^{(2)})_{:,n+1} &= \mathcal{P}_\perp \left[W_V^{(2)} X \cdot \tilde{\sigma}\Big(\mathcal{C} \cdot (W_K^{(2)} X)^\top (W_Q^{(2)} X)\Big)\right]_{:,n+1} \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)} X)_{:,s} \left[\tilde{\sigma}\Big(\mathcal{C} \cdot (W_K^{(2)} X)^\top (W_Q^{(2)} X)\Big)\right]_{s,n+1} \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)} X)_{:,s} \exp\left(\Big((W_K^{(2)} X)^\top (W_Q^{(2)} X)\Big)_{s,n+1} \right) \\
&= \sum_{s=1}^{n+1} \mathcal{P}_\perp (W_V^{(2)} X)_{:,s} \exp\Big((W_K^{(2)} X)_{:,s}^\top (W_Q^{(2)} X)_{:,n+1} \Big) \\
&= \sum_{k=1}^{K} \Big(u(\tau_{2t_k,d_k-1}, \tau_{2j,d}) + u(\tau_{2t_k-1,d_k}, \tau_{2j,d}) \Big) + \sum_{s=1}^{d} u(\tau_{2j_s-1,s}, \tau_{2j,d})
\end{aligned} \tag{3.47}$$

in which the last line is by definition of $u(\cdot,\cdot)$ in Equation (3.29).

For any indices s, j_s, j, d, we can simplify the expression for $u(\tau_{2j_s-1,s}, \tau_{2j,d})$ by observing that

$$\begin{aligned}
u(\tau_{2j_s-1,s}, \tau_{2j,d}) &= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, W_V^{(2)} \boldsymbol{e}(\tau_{2j_s-1,s}) \quad \text{by Eq 3.29} \\
&= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, \boldsymbol{O}_{:,r(2j_s-1,s)} \quad \text{by Eq 3.46} \\
&= \mathcal{P}_\perp \exp\Big(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, \mathrm{b}_{j_s D+s} \quad \text{by Equation (3.45)} \\
&= \exp\Big(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, \mathrm{b}_{j_s D+s} \quad \text{by Equation (3.44).}
\end{aligned} \tag{3.48}$$

Likewise by Equation (3.29), Equation (3.46), Equation (3.45), Equation (3.44)

$$u(\tau_{2j_s,s-1}, \tau_{2j,d}) = -\exp\Big(\boldsymbol{e}(\tau_{2j_s,s-1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big) \exp(a_{j_s,s}) \mathrm{b}_{j_s D+s} \tag{3.49}$$

By Equation (3.48) and Equation (3.49),

$$\begin{aligned}
&u(\tau_{2t_k,d_k-1}, \tau_{2j,d}) + u(\tau_{2t_k-1,d_k}, \tau_{2j,d}) \\
&= \exp\Big(\boldsymbol{e}(\tau_{2t_k-1,d_k})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, \mathrm{b}_{t_k D+d_k} \\
&\quad - \exp\Big(\boldsymbol{e}(\tau_{2t_k,d_k-1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big) \exp(a_{t_k,d_k}) \mathrm{b}_{t_k D+d_k} \\
&= \Big[\exp\Big(\boldsymbol{e}(\tau_{2t_k-1,d_k})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big) \\
&\quad - \exp\Big(\boldsymbol{e}(\tau_{2t_k,d_k-1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) + a_{t_k,d_k} \Big) \Big] \mathrm{b}_{t_k D+d_k} \\
&= 0
\end{aligned} \tag{3.50}$$

in which the last line is because the terms inside $\big[\cdots\big]$ cancel each other, because by Equation (3.41)

$$\boldsymbol{e}(\tau_{2t_k-1,d_k})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) = \boldsymbol{e}(\tau_{2t_k,d_k-1})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) + a_{t_k,d_k}$$

Plugging Equation (3.50) and Equation (3.48) into Equation (3.47),

$$\begin{aligned}
\tilde{a}_2(X;\theta^{(2)})_{:,n+1} &= \sum_{s=1}^{d} u(\tau_{2j_s-1,s}, \tau_{2j,d}) \\
&= \sum_{s=1}^{d} \exp\Big(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d}) \Big)\, \mathrm{b}_{j_s D+s}
\end{aligned} \tag{3.51}$$

Therefore, $\tilde{a}_2(X;\theta^{(2)})_{:,n+1}$ lies in the span of $\{\mathrm{b}_{j_s D+s}\}_{s\in[d]}$. We will from now on assume $\langle \mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n}), \mathrm{b}_{j_s D+s}\rangle > M$ for all possible choices of p ending with a closed bracket with grammar depth at least 1 for some constant $M \in (0,1)$. Here M exists because

$$\langle \mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n}), \mathrm{b}_{j_s D+s}\rangle = \frac{\exp\left(\boldsymbol{e}(\tau_{2j_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d})\right)}{\sqrt{\sum_{s'=1}^{d} \exp\left(2\boldsymbol{e}(\tau_{2j'_s-1,s})^\top (W_K^{(2)})^\top W_Q^{(2)} \boldsymbol{e}(\tau_{2j,d})\right)}} > 0,$$

for all possible combination of $j_k, k \in [d]$ and s, and there are only finite number of such combinations.

Constructing the projection function $\mathrm{g}^{(2)}$ We will finally show there exists a 6-layer MLP $\mathrm{g}^{(2)}$ with width $O(D^2k^2)$, such that for any dyck prefix q with n being the length of q, X being the input of the second layer given q and $p(p)$ being the groundtruth next-token probability vector given q [65], it holds that,

$$\mathrm{g}^{(2)}\left(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1}\right) = p(q).$$

We will assume the last token of q is $\tau_{t,d}$. Suppose that $\mathrm{b}_{m-1}, \mathrm{b}_m$ is an orthonormal basis of the normal space of $\mathrm{span}\{\mathrm{b}_1, .., \mathrm{b}_{m-2}\}$, then we can first observe that for $U = \mathrm{b}_m \mathrm{b}_m^\top + \mathrm{b}_{m-1}\mathrm{b}_{m-1}^\top$, it holds that

$$U(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1}) = U\boldsymbol{e}(\tau_{t,d}).$$

is unique for every t, d. Then based on Lemma 38, there exists a 2-layer MLP with width $4kD$ that maps $U(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1})$ to (t,d). This implies that there exists a 2-layer MLP with width $4kD$ that maps $\mathrm{LN}((\tilde{a}_2(X;\theta^{(2)})_{:,n}) + X_{:,n}$ to (t,d).

Further, let matrix $U' = \sum_{j=1}^{Dk} \boldsymbol{o}_j \mathrm{b}_j^\top$ where $\boldsymbol{o}_j$ is the Dk dimension one-hot vector with the $j-$th entries being 1. Then when t is an even number and $d \geq 1$, based on Equation (3.51) and the definition of M,

$$U'(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1})_{t'D+d'} \begin{cases} = 0, & \tau_{2t'-1,d'} \text{ is not an unmatched open brackets in } p. \\ > M, & \tau_{2t'-1,d'} \text{ is an unmatched open brackets in } p. \end{cases}$$

Then based on Lemma 41, there exists 2-layer MLP with width kD that operates on

$$\left(U'(\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1})_{t'D+d'}\right)_{t'\in[k]}$$

for a fixed d' and outputs the nonzero index in it, if such index exists. Hence, we can choose the weight of the first and second layer of $\mathrm{g}^{(2)}$, such that the output of the second layer is $(t,d) \oplus x$, where $2x_{d'} - 1$ is the type of the unmatched open brackets with grammar depth d' if t is an even number, $d \geq d' \geq 1$.

Now based on Lemma 40, we can choose the third and fourth layer of $\mathrm{g}^{(2)}$ to perform indexing and let the output of the fourth layer be (t,d,y), where $y = x_d$ when $d \geq 1$. [66] Notice that this triplet contains all the necessary information to infer $p(q)$ because it uniquely determines the type of last unmatched open bracket,

[65] That is $p(q)_t = p(\text{The next token of } q \text{ has type } t)$

[66] When $d = 0$, y does not matter since there is no unmatched open brackets.

1. If t is odd (i.e. the last bracket is open), and then the type of last unmatched open bracket is t.
2. If t is even and $d = 0$, then all the brackets is matched.
3. If t is even and $d \geq 1$, then the type of last unmatched bracket is y.

One may finally construct a 2-layer MLP f that maps (t, d, y) to the corresponding probability vector. As the input of g has bounded norm,

$$\|\mathrm{LN}(\tilde{a}_2(X;\theta^{(2)})_{:,n+1}) + X_{:,n+1}\|_2 \leq 1 + \max_{t,d} \|\boldsymbol{e}(\tau_{t,d})\|,$$

the output of the constructed 4 layers also has a bounded norm. Hence, we can assume there exists constant $M' > 1$, such that $y \leq M'$. Now we will discuss by the value of t,

1. t is odd, then one can neglect the third dimension and the correct probability is determined by d and can be represented by a width-$2D$ network based on Lemma 38.
2. t is even. When $d = 0$, one can construct a width-1 network mapping any y to the correct probability distribution as it is unique. When $d \geq 1$, one can construct a width-$2K$ network mapping $x_d \in [K]$ to the correct probability distribution based on Lemma 38. Then by Lemma 42, one can construct a width-$4KD$ network that maps (d, y) to the corresponding probability distribution.

Putting together and using Lemma 42 again, one can construct a width-$8K^2D$ network that maps (t, d, y) to the correct next token probability prediction. The proof is then completed.

□

3.6.4.2 Proof of Theorem 2: approximate balance

Theorem 2 (Approximate Balance)**.** *Consider a 2-layer Transformer $\mathcal{T}$ (Equation (3.27)) with a minimal first layer (Assumption 10) and a γ-Lipschitz $\mathrm{g}^{(2)}$ for $\gamma > 0$, trained on sequences of length N with the mean squared loss (Equation (3.22)).*

Suppose the loss is approximately optimal, precisely, the set of second-layer weights $\bar{\theta}_N^{(2)}$ satisfies $\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{q,k,D,N}) \leq (\frac{q(1-q)}{k^2})^N \epsilon$, for every positive integer $N > 8D$ and sufficiently small $\epsilon > 0$. Then, there exists a constant $C_{\gamma,\epsilon,D}$, such that $\forall 0 \leq d' \leq D, 1 \leq d \leq D, i, j \in [k]$, it holds that

$$\|S_{d,d',i,j}[\bar{\theta}_N^{(2)}]\| \leq \frac{C_{\gamma,\epsilon,D}}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{3.36}$$

Proof. The key idea is similar to the proof of necessity in Theorem 1. That is, we will construct two input sequences with different next-word distributions, and show that the approximate balance condition must hold so that inserting (a bounded number of) pairs of matching brackets does not collapse the two predicted distributions given by the Transformer.

Constructing the input sequences. Let $\boldsymbol{t} := \arg\min_{\tilde{\boldsymbol{t}} \in [k]^{d-1}} \|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$, and let $\boldsymbol{t}'$ denote the prefix that minimizes $\|Q(2j, d, \tilde{\boldsymbol{t}})\|_2$ subject to the constraint that $\boldsymbol{t}'$ must differ from $\boldsymbol{t}$ in the last (i.e.

$(d-1)_{th}$) position, i.e.

$$\boldsymbol{t}' = \arg \min_{\tilde{\boldsymbol{t}}' \in [k]^{d-1}, \boldsymbol{t}'_{d-1} \neq \boldsymbol{t}_{d-1}} Q(2j, d, \tilde{\boldsymbol{t}}').$$

The motivation for such choices of $\boldsymbol{t}, \boldsymbol{t}'$ is that since they differ at least by the last position which is an open bracket, they must lead to different next-word distributions. Note also that $P_{d,j}[\bar{\theta}^{(2)}] = \|Q(2j, d, \boldsymbol{t}')\|$.

With the above definition of $\boldsymbol{t}, \boldsymbol{t}'$, consider two valid Dyck prefixes p_1 and p_2 with length no longer than N, defined as follows: for any $d, d' \in [D], i, j \in [k]$, consider a common prefix

$$p = \underbrace{\tau_{2i-1} \ldots \tau_{2i-1}}_{d' \text{ open brackets}} \underbrace{\tau_{2i-1}\tau_{2i} \ldots \tau_{2i-1}\tau_{2i}}_{(\lfloor \frac{N}{2} \rfloor - d' - d - 1) \text{ pairs}} \underbrace{\tau_{2i} \ldots \tau_{2i}}_{d' \text{ closed brackets}},$$

where τ_i denotes a token with type i whose depth is implicit from the context. Set p_1, p_2 as

$$p_1 = p \oplus \boldsymbol{t} \oplus \tau_{2j-1}\tau_{2j},$$
$$p_2 = p \oplus \boldsymbol{t}' \oplus \tau_{2j-1}\tau_{2j}.$$

That is, p_1, p_2 differ in the last unmatched open bracket. In the following, we will show that the approximate balance condition must hold for the predictions on p_1, p_2 to be sufficiently different.

Bounding the difference in Transformer outputs. For a Transformer $\mathcal{T}$ with second layer parameters $\bar{\theta}_N^{(2)}$, with $\mathcal{P}_{\text{next}}(p)$ indicating the next token probability given a prefix p, by triangle inequality, its outputs on p_1, p_2 satisfy

$$\begin{aligned} &\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{T}[\bar{\theta}_N^{(2)}](p_2)\|_2 \\ &\geq \|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_2 - \left(\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{P}_{\text{next}}(p_1)\|_2 + \|\mathcal{T}[\bar{\theta}_N^{(2)}](p_2) - \mathcal{P}_{\text{next}}(p_2)\|_2 \right). \end{aligned} \tag{3.52}$$

Bounding each term separately:

$$\|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_2 \geq \frac{1}{\sqrt{2k}} \|\mathcal{P}_{\text{next}}(p_1) - \mathcal{P}_{\text{next}}(p_2)\|_1 = \frac{1}{\sqrt{2k}} \text{TV}(p_1, p_2),$$

where $\text{TV}(p_1, p_2)$ denotes the TV distance in the next-word distributions from p_1 and p_2, and

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{P}_{\text{next}}(p_1)\|_2 \leq \sqrt{\epsilon},$$

because $\mathcal{L}(\mathcal{T}[\bar{\theta}_N^{(2)}], \mathcal{D}_{q,k,D,N}) \leq \frac{q(1-q)}{k^2})^N \epsilon$ and the probability of sampling any prefix p is greater than $(\frac{q(1-q)}{k^2})^N$, implying that the per sample next-token squared loss on prefix p is no greater than ϵ. Likewise

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_2) - \mathcal{P}_{\text{next}}(p_2)\|_2 \leq \sqrt{\epsilon}.$$

Plugging into Equation (3.52),

$$\|\mathcal{T}[\bar{\theta}_N^{(2)}](p_1) - \mathcal{T}[\bar{\theta}_N^{(2)}](p_2)\|_2 \geq \frac{1}{\sqrt{2k}} \text{TV}(p_1, p_2) - 2\sqrt{\epsilon}. \tag{3.53}$$

Define by A_p the contribution of p to the attention output (before LayerNorm) of the last position of p_1, p_2:

$$A_p = \sum_{1 \le d'' < d'} \Big(u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''}) \Big)$$
$$+ \lfloor \frac{N - 2d' - 2d}{2} \rfloor \Big(u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1}) \Big). \tag{3.54}$$

The attention outputs (before LayerNorm) of p_1, p_2, denoted by $A(p_1)$ and $A(p_2)$, satisfy that

$$\mathcal{P}_\perp A(p_1) = \mathcal{P}_\perp (A_p + Q(2j, d, \boldsymbol{t})),$$
$$\mathcal{P}_\perp A(p_2) = \mathcal{P}_\perp (A_p + Q(2j, d, \boldsymbol{t}')). \tag{3.55}$$

Note that for any prefix p',

$$\mathcal{T}[\bar{\theta}_N^{(2)}](p') = \mathrm{g}^{(2)}(\mathrm{LN}_{C_{LN}}(\mathcal{P}_\perp A(p'))) + \boldsymbol{e}(\tau_{2i,d'}) \tag{3.56}$$
$$= \mathrm{g}^{(2)}\Big(\frac{\mathcal{P}_\perp A(p')}{\|\mathcal{P}_\perp A(p')\|}\Big) + \boldsymbol{e}(\tau_{2i,d'}), \tag{3.57}$$

where $\mathrm{g}^{(2)}$ is γ-Lipschitz. Hence by Equation (3.57) and Equation (3.53), we have

$$\Big\| \frac{\mathcal{P}_\perp A(p_1)}{\|\mathcal{P}_\perp A(p_1)\|_2} - \frac{\mathcal{P}_\perp A(p_2)}{\|\mathcal{P}_\perp A(p_2)\|_2} \Big\|_2 \ge \frac{\mathrm{TV}(p_1, p_2)}{\sqrt{2k}\gamma} - \frac{2\sqrt{\epsilon}}{\gamma} = \Omega_{\frac{1}{\gamma}, \sqrt{\epsilon}}(1). \tag{3.58}$$

Here the TV distance is lower bounded by a constant due to the construction of p_1, p_2, where $\boldsymbol{t}, \boldsymbol{t}'$ differ at the last open bracket.

We will then show that A_p should not be too much larger in norm than $Q(2j, d, \boldsymbol{t})$ or $Q(2j, d, \boldsymbol{t}')$. First, let's state a helper lemma about the contrapositive:

Lemma 24. *For any $\epsilon > 0$, there exists a constant R_ϵ, such that for any $a, b \in \mathbb{R}^d$ and any $r \in \mathbb{R}^d$ such that $\|r\|_2 \ge R_\epsilon \cdot \max\{\|a\|_2, \|b\|_2\}$, it holds that*

$$\Big\| \frac{a + r}{\|a + r\|_2} - \frac{b + r}{\|b + r\|_2} \Big\|_2 \le \epsilon.$$

Proof. Denote $r_0 := \max\{\|a\|_2, \|b\|_2\}$. Then $R_\epsilon := \frac{4r_0}{\epsilon} + 1$ suffices:

$$\Big\| \frac{r + a}{\|r + a\|_2} - \frac{r + b}{\|r + b\|_2} \Big\| \le \|r\| \cdot \Big| \frac{1}{\|r + a\|} - \frac{1}{\|r + b\|} \Big| + \frac{\|a\|}{\|r + a\|} + \frac{\|b\|}{\|r + b\|}$$
$$\le \|r\| \cdot \Big(\frac{1}{\|r\| - r_0} - \frac{1}{\|r\| + r_0} \Big) + \frac{2r_0}{\|r\| - r_0}$$
$$= \frac{2r_0}{\|r\| - r_0} \cdot \Big(\frac{\|r\|}{\|r\| + r_0} + 1 \Big) \le \frac{4r_0}{\|r\| - r_0} \le \frac{4r_0}{R_\epsilon - r_0} \le \epsilon.$$

□

Consider Equation (3.58), Equation (3.55), and Lemma 24 in which

$$a = \mathcal{P}_\perp Q(2j, d, \boldsymbol{t}),$$
$$b = \mathcal{P}_\perp Q(2j, d, \boldsymbol{t}'),$$
$$r = \mathcal{P}_\perp A_p.$$

By Lemma 24, in order for Equation (3.58) to hold, there exists $R_\epsilon \in \mathbb{R}$ such that

$$\|\mathcal{P}_\perp A_p\|_2 \le R_\epsilon \cdot \max\{\|\mathcal{P}_\perp Q(2j,d,\boldsymbol{t})\|_2, \|\mathcal{P}_\perp Q(2j,d,\boldsymbol{t}')\|_2\}.$$

Note that by definition in Equation (3.35), $\|Q(2j,d,\boldsymbol{t})\|_2 \le \|Q(2j,d,\boldsymbol{t}')\|_2 = P_{d,j}[\bar{\theta}_N^{(2)}]$. Hence

$$\|\mathcal{P}_\perp A_p\|_2 \le R_\epsilon \cdot \|\mathcal{P}_\perp Q(2j,d,\boldsymbol{t}')\|_2 \le R_\epsilon \cdot \|\mathcal{P}_\perp\|_2 \cdot \|Q(2j,d,\boldsymbol{t}')\|_2 = R_\epsilon \cdot P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{3.59}$$

As Equation (3.59) holds for p with any d, d', if one choose $d' = 1$, this shows

$$\|u(\tau_{2j,d-1}, \tau_{2i,1}) + u(\tau_{2j,d-1}, \tau_{2i-1,2})\|_2 \le \frac{4R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}]}{N}. \tag{3.60}$$

Further, it holds that for any $1 < d' \le d-1$,

$$\begin{aligned}
&\| \sum_{1\le d''<d'} \Big(u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''})\Big) \\
&+ \lfloor \frac{N-2d'-2d}{2} \rfloor \Big(u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})\Big) \|_2 \\
&\le R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}] \text{ , and} \\
&\| \sum_{1\le d''<d'+1} \Big(u(\tau_{2j,d-1}, \tau_{2i,d''-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d''})\Big) \\
&+ \lfloor \frac{N-2d'-2d-2}{2} \rfloor \Big(u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})\Big) \|_2 \\
&\le R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}].
\end{aligned}$$

Then by triangle inequality,

$$\begin{aligned}
\lfloor \frac{N-2d'-2d-2}{2} \rfloor \| &\Big(u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})\Big) \\
&- \Big(u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})\Big) \|_2 \le 2R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}].
\end{aligned}$$

Because $N \ge 8D$, we have that $\lfloor \frac{N-2d'-2d-2}{2} \rfloor \ge \frac{N}{8}$, hence it holds that

$$\begin{aligned}
\| &\Big(u(\tau_{2j,d-1}, \tau_{2i,d'+1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+2})\Big) \\
&- \Big(u(\tau_{2j,d-1}, \tau_{2i,d'}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'+1})\Big) \|_2 \le \frac{16R_\epsilon P_{d,j}[\bar{\theta}_N^{(2)}]}{N}.
\end{aligned}$$

Combined with Equation (3.60), one can conclude that,

$$S_{d,d',i,j} = \|u(\tau_{2j,d-1}, \tau_{2i,d'-1}) + u(\tau_{2j,d-1}, \tau_{2i-1,d'-1})\| \le \frac{16DR_\epsilon}{N} P_{d,j}[\bar{\theta}_N^{(2)}]. \tag{3.61}$$

This completes the proof. □

3.6.4.3 Proof of Theorem 3: indistinguishability from a single component

We now show the limitation of interpretability from a single component, using a Lottery-Ticket-style argument by pruning from large random Transformers.

Theorem 3 (Indistinguishability From a Single Component). *Consider any L-layer Transformer $\mathcal{T}$ (Equation (3.27)) with embedding dimension m, attention dimension m_a, and projection function $\mathrm{g}^{(l)}$ as 2-layer ReLU MLP with width w, for $l \in [L]$.* [67] *For any $\delta \in (0,1)$ and $N \in \mathbb{N}^+$, consider a $4L$-layer* random *Transformer $\mathcal{T}_{large}$ with embedding dimension $m_{\text{large}} = O(m \log(Lm/\delta))$, attention dimension $m_{\text{large},a} = O(m_a L \log \frac{m_a m L N}{\epsilon\delta})$, and projection function $\mathrm{g}_{\text{large}}$ as 4-layer ReLU MLP with width $w_{\text{large}} = O(\max\{m, w\} L \log \frac{wmLN}{\epsilon\delta})$.*

Assume that $\|W\|_2 \le 1$ for every weight matrix W in $\mathcal{T}$, and suppose the weights are randomly sampled as $W_{i,j} \sim U(-1,1)$ for every $W \in \mathcal{T}_{large}$. Then, with probability $1-\delta$ over the randomness of $\mathcal{T}_{large}$, there exists a nonstructural pruning (Definition 15) of $\mathcal{T}_{large}$, denoted as $\tilde{\mathcal{T}}_{large}$, which ϵ-approximates $\mathcal{T}$ with respect to $\|\cdot\|_{1,2}$ for any input $X \in \mathbb{R}^{m\times N}$ satisfying $\|\boldsymbol{X}\|_{1,2} \le 1$. [68]

Proof. We will first introduce some notation. For vector $x \in \mathbb{R}^a$ and $y \in \mathbb{R}^b$, we will use $x \oplus y$ to denote their concatenation. We will use 0^a to denote the all-zero vector with dimension a. We will also assume without loss of generality that $w \ge 2m$. [69]

We will use $\bar{\boldsymbol{X}}$ to denote $\begin{bmatrix} \boldsymbol{X} \\ 0^{(m_{\text{large}}-m')\times N} \end{bmatrix}$ for $\boldsymbol{X} \in \mathbb{R}^{m'\times N}$ with $m' \le m_{\text{large}}$.

In the following, a *random network* refers to a network whose weights have entries sampled from a uniform distribution, i.e. $W_{i,j} \sim U(-1,1)$ for every weight W in the random network.

We will first recall Lemma 25 from Pensia et al. [2020] which shows that a pruned 2-layer random network can approximate a linear function.

Lemma 25 (Approximating a linear function; Theorem 1 of Pensia et al. [2020] restated). *Let $W \in \mathbb{R}^{m'\times m}$, $\|W\|_2 = O(1)$, then for $\sigma \in \{\text{ReLU}, \mathcal{I}\}$, where $\mathcal{I}$ represents the identity operator, for a random network $g(x) = W_2\sigma(W_1 x)$ with $W_2 \in \mathbb{R}^{m'\times h}, W_1 \in \mathbb{R}^{h\times m}$ for hidden dimension $h = O(m \log(\frac{mm'}{\min\{\epsilon,\delta\}}))$, with probability $1-\delta$, there exists boolean masking matrices M_1, M_2, such that for any $x \in \mathbb{R}^w$,*

$$\|(M_2 \odot W_2)\sigma\big((M_1 \odot W_1)x\big) - Wx\| \le \epsilon\|x\|_2,$$

where $\odot$ denotes the Hadamard product.

We then derive two approximation results Lemmas 26 and 27 based on Lemma 25.

Lemma 26. *Under the setting of Theorem 3, with probability $1 - 2\delta/3$, for any $l \in [L], l' \in [4L-1]$, let $\mathcal{T}^{(l)}$ be the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th and the (l')-th layer $\mathcal{T}_{large}^{(l'-1)}, \mathcal{T}_{large}^{l'}$, named $\tilde{\mathcal{T}}_{large}^{(l'-1)}, \tilde{\mathcal{T}}_{large}^{l'}$ such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \le 2L, \boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

[67]For notational convenience, we assume all layers share the same dimensions and projection functions. The proof can be trivially extended to cases where the dimensions and projection functions are different.

[68]Here the input and output dimension of $\tilde{\mathcal{T}}_{\text{large}}$ is actually m_{large} which is larger than m; additional dimensions are padded with zeroes. The norm constraint can be easily extended to an arbitrary constant.

[69]We can always pad dimensions if w is too small.

1. *$\tilde{\mathcal{T}}_{large}^{(l'-1)}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*

2. *$\tilde{\mathcal{T}}_{large}^{l'} \circ \tilde{\mathcal{T}}_{large}^{(l'-1)}(\bar{\boldsymbol{X}})$ is an $\left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon$-approximation of $\overline{\mathcal{T}^{(l)}(\boldsymbol{X})}$ with respect to $1,2$-norm.*

Lemma 27. *Under the setting of Theorem 3, for any matrix $W \in \mathbb{R}^{4m\times 4m}$, $\|W\|_2 \leq 1$, with probability $1-\delta/4$, for any $l' \in [4L]$, there exists a pruning of the l-th layer $\mathcal{T}_{large}^{(l')}$, named $\tilde{\mathcal{T}}_{large}^{(l')}$, such that when defined on domain $\boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l')}$ is independent of the last $m_{\text{large}} - 4m$ rows of the input.*

2. *$a(x) = \tilde{\mathcal{T}}_{large}^{(l')}(\bar{\boldsymbol{X}})$ is an $\left(\frac{C}{1000L^2}\right)^{4L}\epsilon$-approximation of $\hat{g}(\boldsymbol{X}) = \overline{W\boldsymbol{X}}$ with respect to $1,2$-norm.*

The proof of Lemmas 26 and 27 is deferred to Section 3.6.4.3.1 We can now prove the theorem.

We will first show with induction that if we 1) prune the $(2l-1)$-th and $2l$-th layers of $\mathcal{T}_{\text{large}}$ to approximate $\mathcal{T}^{(l)}$ for each $l \in [L]$, and 2) prune the $2L+1$ to $4L$-th layers of $\mathcal{T}_{\text{large}}$ to approximate identity, then the pruned large transformer will be an ϵ-approximation of $\mathcal{T}$ for any input $\|\boldsymbol{X}\|_{1,2} \leq 1$.

We will perform induction on l: Let $\mathcal{T}^{(1:l)}$ define the composition of layer 1 to l, i.e. $\mathcal{T}^{(1:l)}(\boldsymbol{X}) := \mathcal{T}^{(l)} \circ \mathcal{T}^{(l-1)} \circ \cdots \circ \mathcal{T}^{(1)}(\boldsymbol{X})$, and define $\epsilon_l := \left(\frac{C}{1000L^2}\right)^{4L-3-l}\epsilon$. Suppose that $\mathcal{T}_{\text{large}}^{(1:2l)}$ is an ϵ_l-approximation of $\mathcal{T}^{(1:l)}$. Note that $\|\mathcal{T}^{(1:l)}(\boldsymbol{X})\|_{1,2} \leq (l+1)$, since each attention output has a bounded norm of 1 and every weight matrix in projection function g has spectral norm smaller than 1, hence the norm will at most increment 1 (due to residual connection) after each layer. We have that

$$\left\|\tilde{\mathcal{T}}_{\text{large}}^{(1:2l)}\left(\bar{\boldsymbol{X}}\right)\right\|_{1,2} \leq 4l \leq 4L.$$

Then according to Lemma 36, $\mathcal{T}^{(l+1)}$ is $(1+200L^2/C)$-Lipschitz with respect to the set of intermediate outputs $\{\left(\tilde{\mathcal{T}}_{\text{large}}^{(1:2l)}(\bar{\boldsymbol{X}})\right)_{1:m} \mid \|\boldsymbol{X}\|_{1,2} \leq 1\}$. We also have that $\mathcal{T}^{(1:l)}(\boldsymbol{X})$ is $(1+200L^2/C)^l$-Lipschitz. Now we can apply Lemma 28 to show that $\mathcal{T}_{\text{large}}^{(1:2l+2)}$ can ϵ'-approximate $\mathcal{T}^{(1:l+1)}$ with

$$\begin{aligned}\epsilon' &= \epsilon_l(1+200L^2/C) + \epsilon\left(\frac{C}{1000L^2}\right)^{4L-3}(1+200L^2/C)^l + \epsilon_l\left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon \\ &\leq \left(\frac{C}{1000L^2}\right)^{4L-4-l}\epsilon = \epsilon_{l+1}.\end{aligned}$$

The induction is then completed and we have the composition of $\tilde{\mathcal{T}}_{\text{large}}^{i}$ for $i \in [2L]$ ϵ_L-approximates the composition of $\mathcal{T}$ with $\epsilon_L = \left(\frac{C}{1000L^2}\right)^{3L-3}\epsilon$. We will then perform another induction showing that the composition of $\tilde{\mathcal{T}}_{\text{large}}^{i}$ for $i \in [2L+l]$ ϵ_{l+L}-approximates $\mathcal{T}$ with $\epsilon_{l+L} = \left(\frac{C}{1000L^2}\right)^{3L-3-l}\epsilon$. Suppose the statement holds for $L-1 \geq l \geq 0$.

The induction step is similar, because we have $\mathcal{T}$ is $(1+200L^2/C)^L$ Lipschitz, by Lemma 28, it holds that the composition of $\mathcal{T}_{\text{large}}^{i}$ for $i \in [2L+l+1]$ ϵ'-approximates $\mathcal{T}$ with,

$$\begin{aligned}\epsilon' &= \epsilon_{l+L} + \epsilon\left(\frac{C}{1000L^2}\right)^{4L}(1+200L^2/C)^L + \epsilon_{l+L}\epsilon\left(\frac{C}{1000L^2}\right)^{4L} \\ &\leq \epsilon\left(\frac{C}{1000L^2}\right)^{3L-4-l}\epsilon = \epsilon_{L+l+1}.\end{aligned}$$

This concludes the induction and prove the first claim of the theorem. For the second claim, notice that through similar induction steps, we can prune arbitrary layer of $\mathcal{T}_{\text{large}}$ to approximate identity function and obtain the same approximation rate, this concludes the proof for the second claim. $\square$

3.6.4.3.1 Helper lemmas for Theorem 3

Error Analysis Our first lemma shows that the composition of ϵ-approximation can approximate the composition of the original function.

Lemma 27. *Under the setting of Theorem 3, for any matrix $W \in \mathbb{R}^{4m\times 4m}$, $\|W\|_2 \le 1$, with probability $1-\delta/4$, for any $l' \in [4L]$, there exists a pruning of the l-th layer $\mathcal{T}_{large'}^{(l')}$ named $\tilde{\mathcal{T}}_{large'}^{(l')}$ such that when defined on domain $\boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l')}$ is independent of the last $m_{\text{large}} - 4m$ rows of the input.*
2. *$a(x) = \tilde{\mathcal{T}}_{large}^{(l')}(\bar{\boldsymbol{X}})$ is an $\left(\frac{C}{1000L^2}\right)^{4L}$ ϵ-approximation of $\hat{g}(\boldsymbol{X}) = \overline{W\boldsymbol{X}}$ with respect to $1,2$-norm.*

Proof. One can prune the value matrix on layer l' to zero and the rest is a direct consequence of Lemmas 25 and 43. $\square$

Lemma 28. *Given three metric spaces A, B, C equipped with same metric $\|\cdot\|$. Suppose $f_1 : A \to B, f_2 : B \to C$ are ϵ_1, ϵ_2-approximations of g_1, g_2 with respect to $\|\cdot\|$, where g_1 is a Lipschitz function with constant λ_1 with respect to $\|\cdot\|$ and $\|g_2(x)\| \le \lambda_2 x$, then it holds that, $f_1 \circ f_2$ is an ϵ'-approximation of $g_1 \circ g_2$, with $\epsilon' = (\lambda_2+\epsilon_1)(\lambda_1+\epsilon_2) - \lambda_1\lambda_2$*

Proof. For any $x \in \mathbb{R}^{d_1}$, it holds that,

$$\|f_1(x) - g_1(x)\| \le \epsilon_1\|x\|.$$

This then suggests that,

$$\begin{aligned}
&\|f_2(f_1(x)) - g_2(g_1(x))\| \\
\le&\|f_2(f_1(x)) - g_2(f_1(x))\| + \|g_2(f_1(x)) - g_2(g_1(x))\| \\
\le&\epsilon_2\|f_1(x)\| + \lambda_2\|f_1(x) - g_1(x)\| \\
\le&\epsilon_2\|g_1(x)\| + (\lambda_2+\epsilon_2)\|f_1(x) - g_1(x)\| \\
\le&(\epsilon_2\lambda_1 + \epsilon_1\lambda_2 + \epsilon_1\epsilon_2)\|x\|.
\end{aligned}$$

$\square$

Approximating ReLU MLP We will first show an extension of Lemma 25, illustrating that a pruned wide 4-layer ReLU MLP can approximate any 2-layer ReLU MLP.

Lemma 29. *Consider any 2-layer ReLU MLP $g : \mathbb{R}^{4m} \to \mathbb{R}^{4m}$ parameterized by $W_1 \in \mathbb{R}^{4m\times w}, W_2 \in \mathbb{R}^{w\times 4m}$, $\|W_1\|_2, \|W_2\|_2 \le 2\sqrt{2}$, for any $\delta, \epsilon \in (0,1)$, consider a random 4-layer ReLU MLP f with input and output dimension $4m$ and width $w' = O(w\log(\frac{wm}{\min\{\epsilon,\delta\}}))$ parameterized by $W_{\text{large},i}$, with probability $1-\delta$ over the randomness of weight of f, there exists a nonstructural pruning of f named $\tilde{f}$, such that $\tilde{f}$ is an $\epsilon-$approximation of f with respect to $2-$norm.*

Proof. Choose $\epsilon_0 = \epsilon/8$. We only need to show there exists boolean matrices M_1, M_2, M_3, M_4, such that,

$$\Big\| \Big(M_4 \odot W_{\text{large},4} \text{ReLU}\big((M_3 \odot W_{\text{large},3}) \text{ReLU}\big((M_2 \odot W_{\text{large},2}) \text{ReLU}\big((M_1 \odot W_{\text{large},1}) x\big)\big)\big)\Big) \\ - W_2 \text{ReLU}\left(W_1 \boldsymbol{X}\right) \Big\|_2 \leq \epsilon.$$

By Lemma 25, there exists boolean matrices $M_1 \in \mathbb{R}^{w' \times 4m}$ and $M_2' \in \mathbb{R}^{w \times w'}$, such that for any $x \in \mathbb{R}^{4m}$,

$$\| \left(\begin{bmatrix} M_2' \\ 0^{(w'-w)\times w'} \end{bmatrix} \odot W_{\text{large},2} \right) \text{ReLU}\big((M_1 \odot W_{\text{large},1}) x\big) - \begin{bmatrix} W_1 x \\ 0^{w'-w} \end{bmatrix} \|_2 \leq \epsilon_0 \|x\|_2.$$

Hence we can choose $M_2 = \begin{bmatrix} M_2' \\ 0^{(w'-w)\times w'} \end{bmatrix}$ and have $f_1(x) = \text{ReLU}\big((M_2 \odot W_{\text{large},2}) \text{ReLU}\big((M_1 \odot W_{\text{large},1}) x\big)\big)$ is ϵ_0-approximation of $g_1(x) = \begin{bmatrix} \text{ReLU}(W_1 x) \\ 0^{w'-w} \end{bmatrix}$.

Again by Lemma 25, there exists boolean matrices $M_3' \in \mathbb{R}^{w' \times w}$ and $M_4 \in \mathbb{R}^{4m \times w'}$, such that for any $y \in \mathbb{R}^{w}$,

$$\| \left(M_4 \odot W_{\text{large},4} \right) \text{ReLU} \left(\begin{bmatrix} M_3', 0^{w' \times (w'-w)} \end{bmatrix} \begin{bmatrix} y \\ 0^{w'-w} \end{bmatrix} \right) \leq \epsilon_0 \|y\|_2$$

Hence we can choose $M_3 = \begin{bmatrix} M_3', 0^{w' \times (w'-w)} \end{bmatrix}$, and have $f_2(x) = \text{ReLU}\big((M_4 \odot W_{\text{large},4}) \text{ReLU}\big((M_3 \odot W_{\text{large},3}) x\big)\big)$ is ϵ_0-approximation of $g_2(x) = W_2 x$.

It is also easy to check g_1 and g_2 are both $2\sqrt{2}$-lipschitz and $g_1(0) = 0$. By Lemma 28, we conclude that $\tilde{f} = f_1 \odot f_2$ is ϵ'-approximation of $g = g_1 \odot g_2$, with $\epsilon' = 4\sqrt{2}\epsilon_0 + \epsilon_0^2 \leq \epsilon$. □

This lemma then yields the following corollaries.

Corollary 1. *Under the setting of Theorem 3, with probability $1 - \delta/4$, for any $l \in [L], l' \in [4L]$, there exists a pruning of the projection function $\mathrm{g}_{\text{large}}^{(l')}$, named $\mathrm{g}_{\text{large}}^{(\tilde{l}')}$, such that*

1. *$\mathrm{g}_{\text{large}}^{(\tilde{l}')}$ is independent of the last $m_{\text{large}} - m$ dimension of the input.*

2. $a(x) = \mathrm{g}_{\mathrm{large}}^{(\tilde{l}')}\left(\begin{bmatrix} x \\ 0^{m_{\mathrm{large}}-m} \end{bmatrix}\right)$ *is an* $\left(\frac{C}{1000L^2}\right)^{4L}$ ϵ*-approximation of* $\hat{g}(x) = \begin{bmatrix} \mathrm{g}^{(l)}(x) \\ 0^{m_{\mathrm{large}}-m} \end{bmatrix}$ *with respect to* $2-$*norm.*

Proof. One can construct such pruning by pruning the last $m_{\mathrm{large}} - m$ rows of the weight of the last layer and the last $m_{\mathrm{large}} - m$ columns of the weight of the first layer of $\mathrm{g}_{\mathrm{large}}^{(l')}$ to zero and then apply Lemma 29. □

Approximating Attention Patterns We will now show that the attention pattern can be approximated by pruning random Transformer layers.

Lemma 30. *For any $\delta, \epsilon \in (0,1)$, for any $W \in \mathbb{R}^m$, $\|W\|_2 \le 1$, for two random matrix $W_1, W_2 \in \mathbb{R}^{m' \times m}$ where $m' = O(m \log(\frac{m}{\min\{\epsilon,\delta\}}))$, suppose $\boldsymbol{X} \in \mathbb{R}^{m \times N}$, then there exists nonstructural pruning of W_1, W_2, named $\tilde{W}_1, \tilde{W}_2$, such that*

$$\|\boldsymbol{X}^\top \tilde{W}_1{}^\top \tilde{W}_2 \boldsymbol{X} - \boldsymbol{X}^\top W \boldsymbol{X}\|_\infty \le \epsilon \|\boldsymbol{X}\|_{1,2}^2$$

Here we adopt $\|\|_\infty$ in vector sense, meaning the entry with largest absolute value.

Proof. Suppose without loss of generality, $\|\boldsymbol{X}\|_{:,i} \le 1$. According to Lemma 25, there exists nonstructural pruning of W_1, W_2, named $\tilde{W}_1, \tilde{W}_2$, such that for any $x \in \mathbb{R}^m$, $\|x\|_2 \le 1$,

$$\|\tilde{W}_1{}^\top \tilde{W}_2 x - Wx\|_2 \le \epsilon.$$

This then suggests that,

$$\|y^\top (\tilde{W}_1{}^\top \tilde{W}_2 x - Wx)\|_2 \le \epsilon \|y\|_2 \le \epsilon.$$

This concludes the proof. □

The next lemma shows how error propogates through the softmax operators.

Lemma 31. *For any dimension d, suppose $x, y \in \mathbb{R}^d$ satisfies $\|x - y\|_\infty \le \epsilon$, then it holds that,*

$$\sum_{i=1}^d \left|\frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} - \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)}\right| \le \exp(2\epsilon) - 1.$$

Proof. One can observe that,

$$\exp(-\epsilon)\exp(x_i) \le \exp(y_i) \le \exp(\epsilon)\exp(x_i)$$

This then suggests,

$$\frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} \exp(-2\epsilon) \le \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)} \le \exp(2\epsilon) \frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)}.$$

Hence,

$$\sum_{i=1}^d \left|\frac{\exp(x_i)}{\sum_{i=1}^n \exp(x_i)} - \frac{\exp(y_i)}{\sum_{i=1}^n \exp(y_i)}\right| \le \max\{\exp(2\epsilon) - 1, 1 - \exp(-2\epsilon)\} = \exp(2\epsilon) - 1.$$

This concludes the proof. □

Approximating Attention Module We will need the following lemma showing there exists a pruning of the value matrix in $\mathcal{T}_{\text{large}}$ such that it has eigenvalues with magnitude $\Theta(1)$.

Lemma 32. *For a matrix $W \in \mathbb{R}^{m_{\text{large}} \times m_{\text{large}}}$, with probability at least $1 - \frac{\delta}{10L}$, there exists a pruning of W, named W', such that all the nonzero entries is contained in a $d \times d$ submatrix of W' that satisfies that (1) all its eigenvalues are within $(\frac{1}{2}, 1)$, (2) the index of row specifying the submatrix and the index of column specifying the submatrix are disjoint.*

Proof. As $w_{\text{large}} = \Omega(m \log(\frac{dL}{\delta}))$, hence we can split $W_{1:\lceil m_{\text{large}}/2 \rceil, \lceil m_{\text{large}}/2 \rceil + 1:m_{\text{large}}}$ into $(m \times (m$ blocks, each with width at least $O(\log(\frac{(m}{\delta}))$ [70]. Within each block, with probability $1 - \frac{\delta}{10Lm_{\text{large}}}$, there exists at least one entry that has value at least $\frac{1}{2}$. We can then choose d disjoint entries in W that are all at least $\frac{1}{2}$, indexed with $\{(a_i, b_i)\}_{i \in [d]}$ where $a_i < a_j$ and $b_i < b_j$ for $i < j$. We can then prune all other entries to zero. Consider the submatrix defined by entries (a, b) for $a \in \{a_i\}_{i \in m}$ and $b \in \{b_i\}_{i \in m}$. Then, this submatrix will be diagonal and contains eigenvalues within $(\frac{1}{2}, 1)$. Further $\{a_i\}_{i \in m}$ and $\{b_i\}_{i \in m}$ must be disjoint because $a_i \le \lceil m_{\text{large}}/2 \rceil < b_i$. The proof is then completed. $\square$

We will also prove that LayerNorm with nonzero normalization constant is Lipschitz.

Lemma 33. *For LayerNorm function defined as $\text{LN}(x) = \frac{\mathcal{P}_\perp x}{\max\{\|\mathcal{P}_\perp x\|_2, C\}}, x \in \mathbb{R}^m$, for any $x, y \in \mathbb{R}^m$, it holds that,*

$$\left\|\text{LN}(x) - \text{LN}(y)\right\|_2 \le 2\|x - y\|_2 / C.$$

Proof. We will proceed by a case analysis:

1. If $\|\mathcal{P}_\perp x\|_2, \|\mathcal{P}_\perp y\|_2 \le C$, then $\left\|\text{LN}(x) - \text{LN}(y)\right\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{C} \le \frac{1}{C}\|x - y\|_2$.
2. If $\|\mathcal{P}_\perp x\|_2, \|\mathcal{P}_\perp y\|_2 > C$, then $\left\|\text{LN}(x) - \text{LN}(y)\right\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{\|\mathcal{P}_\perp y\|_2} + |1 - \frac{\|\mathcal{P}_\perp x\|_2}{\|\mathcal{P}_\perp y\|_2}| \le \frac{2}{C}\|x - y\|_2$.
3. If $\|\mathcal{P}_\perp x\|_2 < C$ and $\|\mathcal{P}_\perp y\|_2 > C$, then $\left\|\text{LN}(x) - \text{LN}(y)\right\|_2 = \frac{\|\mathcal{P}_\perp x - \mathcal{P}_\perp y\|_2}{\|\mathcal{P}_\perp y\|_2} + |\frac{\|\mathcal{P}_\perp x\|_2}{C} - \frac{\|\mathcal{P}_\perp x\|_2}{\|\mathcal{P}_\perp y\|_2}| \le \frac{2}{C}\|x - y\|_2$.

The cases exhaust all possibilities, thus the proof is completed. $\square$

Finally, we will need a lemma showing how error accumulates when we consider both attention patterns and the value matrices.

Lemma 34. *For any dimension d and positive number N, for $P, Q \in \mathbb{R}^{d \times d}$ satisfying that $\|P\|_2, \|Q\|_2 \le 1$, for any $x \in \mathbb{R}^{d \times N}$, if matrix $A \in \mathbb{R}^{N \times N}, B \in \mathbb{R}^{d \times N}$ satisfy that,*

$$\|A - \sigma(x^\top Q x)\|_{1,1} \le \epsilon_1.$$
$$\|B - Px\|_{1,2} \le \epsilon_2.$$
$$\forall i, k \in [N] \sum_{j \in [N]} A_{j,i} = 1, A_{k,i} \ge 0.$$

[70] $O(\cdot)$ hides absolute constants arising from the change of basis in the logarithm.

Then it holds that,

$$\|BA - Px\sigma(x^\top Qx)\|_{1,2} \le (\epsilon_1\|PX\|_{1,2} + \epsilon_2).$$
$$\|\mathrm{LN}_C(BA) - \mathrm{LN}_C(Px\sigma(x^\top Qx))\|_{1,2} \le 2(\epsilon_1\|PX\|_{1,2} + \epsilon_2)/C.$$

Proof. For any $i \in N$, we will have

$$\begin{aligned}
&\left\|(BA)_{:,i} - \left(Px\sigma(x^\top Qx)\right)_{:,i}\right\|_2 \\
=&\left\|\sum_{j\in[N]} A_{j,i}B_{:,j} - \left(\sigma(x^\top Qx)\right)_{j,i}(PX)_{:,j}\right\|_2 \\
\le&\|\sum_{j\in[N]} A_{j,i}(PX)_{:,j} - \left(\sigma(x^\top Qx)\right)_{j,i}(PX)_{:,j}\Big\|_2 + \|\sum_{j\in[N]} A_{j,i}\,(PX-B)_{:,j}\|_2 \\
\le&\|PX\|_{1,2}\sum_{j\in[N]} |A_{j,i} - \left(\sigma(x^\top Qx)\right)_{j,i} + \|PX-B\|_{1,2} \\
\le&\|PX\|_{1,2}\|A-\sigma(x^\top Qx)\|_{1,1} + \|PX-B\|_{1,2} \le \epsilon_1\|PX\|_{1,2} + \epsilon_2.
\end{aligned}$$

The rest follows from Lemma 33

□

A LayerNorm of larger dimension can be made to be functionally equivalent to a LayerNorm of a smaller dimension. Precisely:

Lemma 35. *Given any dimension $d < d'$, it holds that for any $x \in \mathbb{R}^d$,*

$$\mathrm{LN}_C(\begin{bmatrix}\mathcal{P}_\perp x \\ 0^{d'-d}\end{bmatrix}) = \begin{bmatrix}\mathrm{LN}_C(x) \\ 0\end{bmatrix}.$$

Proof. The proof follows directly from definition. □

We will now formally define attention module.

Definition 18 (Attention Module)**.** *We will define attention module $a(\boldsymbol{X} \mid W_V, W_K, W_Q)$ as*

$$a(\boldsymbol{X}) = \mathrm{LN}_C\left(W_V\boldsymbol{X}\sigma(\boldsymbol{X}^\top W_K^\top W_Q\boldsymbol{X})\right).$$

Lemma 36. *Attention module is lipschitz with respect to $1,2$-norm for bounded input. Precisely, consider attention module (Definition 18)parameterized by $\|W_V\|_2, \|W_K\|_2, \|W_Q\|_2 \le 1$ with input domain $\|\boldsymbol{X}\|_{1,2} \le 4L$, $a(\boldsymbol{X})$ is $200L^2/C$-lipschitz with respect to $1,2-$norm.*

Proof. We have that

$$a(\boldsymbol{X}) = \mathrm{LN}_C\left(W_V\boldsymbol{X}\sigma(\boldsymbol{X}^\top W_K^\top W_Q\boldsymbol{X})\right).$$

Choose ϵ to be a sufficiently small constant, such that, $\exp(32L\epsilon) - 1 \leq 64L\epsilon$. Consider $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ satisfying that $\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_{1,2} \leq \epsilon$ and $\|\boldsymbol{X}\|_{1,2} \leq 4L$, $\|\tilde{\boldsymbol{X}}\|_{1,2} \leq 4L$, we will have

$$\begin{aligned}
&\left| \left(\boldsymbol{X}^\top W_K^\top W_Q \boldsymbol{X} - (\tilde{\boldsymbol{X}})^\top W_K^\top W_Q (\tilde{\boldsymbol{X}}) \right)_{i,j} \right| \\
=&\left| (\boldsymbol{X}_{:,i} - \tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q \boldsymbol{X}_{:,j} + (\tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q (\boldsymbol{X}_{:,j} - \tilde{\boldsymbol{X}}_{:,j}) + (\boldsymbol{X}_{:,i} - \tilde{\boldsymbol{X}}_{:,i})^\top W_K^\top W_Q (\boldsymbol{X}_{:,j} - \tilde{\boldsymbol{X}}_{:,j}) \right| \\
\leq& 8L\epsilon + \epsilon^2 \leq 16L\epsilon.
\end{aligned}$$

By Lemma 31, this implies,

$$\|\sigma(\boldsymbol{X}^\top W_K^\top W_Q \boldsymbol{X}) - \sigma((\tilde{\boldsymbol{X}})^\top W_K^\top W_Q (\tilde{\boldsymbol{X}}))\|_{1,1} \leq \exp(32L\epsilon) - 1 \leq 64L\epsilon.$$

We also have

$$\begin{aligned}
\|W_V \left(\boldsymbol{X} - \tilde{\boldsymbol{X}} \right) \|_{1,2} &\leq \epsilon. \\
\|W_V \boldsymbol{X}\|_{1,2} &\leq 4L
\end{aligned}$$

Lemma 34 then implies that

$$\|a(\boldsymbol{X}) - a(\tilde{\boldsymbol{X}})\|_{1,2} \leq 200L^2\epsilon / C.$$

This then concludes the proof. □

We can now prove that a large Transformer Layer and an attention module of the larger Transformer can be pruned to approximate the attention module of a smaller Transformer Layer module.

Lemma 37. *Under the setting of Theorem 3, with probability $1 - \delta/2$, for any $l \in [L], l' \in [4L-1]$, let $a^{(l)}$ be the attention module on the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th layer $\mathcal{T}_{large}^{(l'-1)}$, named $\tilde{\mathcal{T}}_{large}^{(l'-1)}$ and the attention module on l'-th layer $a_{\text{large}}^{l'}$ named $a_{\text{large}}^{\tilde{l'}}$, such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \leq 2L$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l'-1)}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*
2. *$\left(a_{\text{large}}^{\tilde{l'}} \circ \tilde{\mathcal{T}}_{large}^{(l'-1)} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}} - m) \times N} \end{bmatrix} \right) \right)_{1:m}$ is an $\left(\frac{C}{1000L^2} \right)^{4L-1} \epsilon$-approximation of $a^{(l)}(x)$ with respect to $1,2$-norm.*
3. *$\left(\tilde{\mathcal{T}}_{large}^{(l'-1)} \left(\begin{bmatrix} x \\ 0^{(m_{\text{large}} - m) \times N} \end{bmatrix} \right) \right)_{1:m}$ is an $\left(\frac{C}{1000L^2} \right)^{4L} \epsilon$-approximation of $\boldsymbol{X}$ with respect to $1,2$-norm.*

Proof. We will use the shorthand $\epsilon_0 = \left(\frac{C}{1000L^2} \right)^{4L} \epsilon$ and prune in the following order. It holds that for $\epsilon \leq 1$, $\exp(8L^2\epsilon_0) - 1 \leq 16L^2\epsilon_0$.

1. We will prune $W_V^{\text{large},(l')}$ according to Lemma 32 and name the pruned matrix $W_V^{\tilde{\text{large}},(l')}$. By Lemma 32, all the nonzero entries is contained in a $d \times d$ submatrix of W' that satisfies that all its eigenvalues are within $(\frac{1}{2}, 1)$. We will assume WLOG the submatrix is the one specified by row $1 \dots d$ and column $d+1 \dots 2d$ and name the submatrix as W.

2. We will then prune $\mathcal{T}_{\text{large}}^{(l'-1)}$ according to Lemma 27 to output ϵ_0-approximation of $\boldsymbol{X} \in \mathbb{R}^{m \times N} \to \begin{bmatrix} \boldsymbol{X} \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-2m)\times N} \end{bmatrix}$. As W is defined as the submatrix pruned by $W_V^{(t+1)}$, it holds that $W_V^{\tilde{\text{large}},(l')} \begin{bmatrix} \boldsymbol{X} \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} = \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix}$.

3. Finally we will prune $W_K^{\text{large},(l')}, W_Q^{\text{large},(l')}$ according to Lemma 30 to approximate $(W_K^{(l)})^\top W_Q^{(l)}$ up to ϵ_0 error.

we can now calculate the approximation error. For any $\boldsymbol{X} \in \mathbb{R}^{m \times N}$, $\|\boldsymbol{X}\|_{1,2} \le 2L$, suppose

$$\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{X} + \delta_1 \\ W^{-1}\mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} + \delta_2 \\ 0^{(m_{\text{large}}-2m)\times N} \end{bmatrix}$$

Then by our constrution, it holds that $\forall i \in \{1, 2\}$, $\|\delta_i\|_{1,2} \le \epsilon_0 \|\boldsymbol{X}\|_{1,2}$.

We would then have

$$W_V^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X}) = \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X} + W_V^{\tilde{\text{large}},(l')}\delta_2 \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix} \tag{3.62}$$

By our construction, it holds that $\|W_V^{\tilde{\text{large}},(l')}\delta_2\|_{1,2} \le 2\|\delta_2\|_{1,2} \le 2\epsilon_0\|\boldsymbol{X}\|_{1,2}$.

Further, by the construction of $W_K^{\tilde{\text{large}},(l')}, W_Q^{\tilde{\text{large}},(l')}$, it holds that,

$$\begin{aligned} &\Big\| \left(W_K^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right) \\ &- (W_K^{(l)}\boldsymbol{X} + W_K^{(l)}\delta_1)^\top (W_Q^{(l)}\boldsymbol{X} + W_Q^{(l)}\delta_1)\Big\|_\infty \le \epsilon_0 \end{aligned} \tag{3.63}$$

As for any $i, j \in [N]$

$$\begin{aligned}
&\left|\left((W_K^{(l)}\boldsymbol{X} + W_K^{(l)}\delta_1)^\top (W_Q^{(l)}\boldsymbol{X} + W_Q^{(l)}\delta_1) - (W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right)_{i,j}\right| \\
\leq &\left|(W_K^{(l)}\boldsymbol{X}_{:,i})^\top (W_Q^{(l)}\delta_1)_{:,j}\right| + \left|(W_K^{(l)}\delta_1)_{:,i}^\top (W_Q^{(l)}\boldsymbol{X})_{:,j}\right| + \left|(W_K^{(l)}\delta_1)_{:,i}^\top (W_Q^{(l)}\delta_1)_{:,j}\right| \\
\leq &\|\boldsymbol{X}\|_{1,2}^2 (2\epsilon_0 + \epsilon^2) \leq 4\|\boldsymbol{X}\|_{1,2}^2 \epsilon_0.
\end{aligned}$$

combined with Equation (3.63),

$$\left\| \left(W_K^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right) - (W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right\|_\infty \leq \epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2). \tag{3.64}$$

By Lemma 31, this implies

$$\begin{aligned}
&\Bigg\|\sigma\left(\left(W_K^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)\right) \\
&- \sigma\left((W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right)\Bigg\|_{1,1} \leq 4\epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2).
\end{aligned} \tag{3.65}$$

By Lemma 34, Equations (3.62) and (3.65) imply,

$$\begin{aligned}
&\Bigg\|W_V^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\sigma\left(\left(W_K^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)^\top \left(W_Q^{\tilde{\text{large}},(l')}\mathcal{T}^{(\tilde{l'}-1)}(\boldsymbol{X})\right)\right) \\
&- \begin{bmatrix} \mathcal{P}_\perp W_V^{(l)}\boldsymbol{X}\sigma\left((W_K^{(l)}\boldsymbol{X})^\top W_Q^{(l)}\boldsymbol{X}\right) \\ 0^{(m_{\text{large}}-m)\times N} \end{bmatrix}\Bigg\|_{1,2} \leq 8\epsilon_0(1 + 4\|\boldsymbol{X}\|_{1,2}^2)\|\boldsymbol{X}\|_{1,2} \leq 80L^2\epsilon_0.
\end{aligned}$$

Now according to Lemmas 33 and 35, it holds that

$$\|a_{\text{large}}^{\tilde{l'}} \circ \tilde{\mathcal{T}}_{\text{large}}^{(l'-1)}\left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N}\end{bmatrix}\right)_{1:m} - a^{(l)}(x)\|_{1,2} \leq 160L^2\epsilon_0/C.$$

This concludes the proof. □

Approximating Transformer Layers We will finally show that two random Transformer layers can be pruned to approximate a given Transformer layer.

Lemma 26. *Under the setting of Theorem 3, with probability $1 - 2\delta/3$, for any $l \in [L], l' \in [4L-1]$, let $\mathcal{T}^{(l)}$ be the l-th layer of $\mathcal{T}$, there exists a pruning of the $(l'-1)$-th and the (l')-th layer $\mathcal{T}_{large}^{(l'-1)}, \mathcal{T}_{large}^{l'}$, named $\tilde{\mathcal{T}}_{large}^{(l'-1)}, \tilde{\mathcal{T}}_{large}^{l'}$ such that when defined on domain $\|\boldsymbol{X}\|_{1,2} \leq 2L, \boldsymbol{X} \in \mathbb{R}^{m\times N}$,*

1. *$\tilde{\mathcal{T}}_{large}^{(l'-1)}$ is independent of the last $m_{\text{large}} - m$ rows of the input.*

2. $\tilde{\mathcal{T}}^{l'}_{large} \circ \tilde{\mathcal{T}}^{(l'-1)}_{large}(\bar{\boldsymbol{X}})$ *is an* $\left(\frac{C}{1000L^2}\right)^{4L-3}$ ϵ*-approximation of* $\overline{\mathcal{T}^{(l)}(\boldsymbol{X})}$ *with respect to* 1, 2*-norm.*

Proof. We will prune the $(l'-1)$-th layer and the attention module of the l'-th layer according to Lemma 37 to approximate $a^{(l)}$ and the projection function of the l'-th layer according to Corollary 1. Notice that $\left\|a^{(l)}(\boldsymbol{X}) + \boldsymbol{X}\right\|_{1,2} \leq (\frac{2}{C}+1)\|\boldsymbol{X}\|_{1,2}$ and $\mathrm{g}^{(l)}$ is 1-Lipschitz. Then, according to Lemma 28,

$$\left(\tilde{\mathcal{T}}^{l'}_{\text{large}} \odot \tilde{\mathcal{T}}^{(l'-1)}_{\text{large}}\left(\begin{bmatrix} x \\ 0^{(m_{\text{large}}-m)\times N}\end{bmatrix}\right)\right)_{1:m}$$

is an ϵ'-approximation of $\mathcal{T}^{(l)}(x)$, with

$$\epsilon' \leq (\frac{2}{C}+1)\left(\frac{C}{1000L^2}\right)^{4L}\epsilon + \left(\frac{C}{1000L^2}\right)^{4L-2}\epsilon + \left(\frac{C}{1000L^2}\right)^{8L-2}\epsilon^2 \leq \left(\frac{C}{1000L^2}\right)^{4L-3}\epsilon.$$

This concludes the proof. □

3.6.4.4 Technical Lemmas

Lemma 38. *Given any dimension d and number of samples n, for any size-n dataset $\{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, there exists a width-$2n$ two-layer MLP $f : \mathbb{R}^d \to \mathbb{R}$ with ReLU activation such that, $f(\boldsymbol{x}_i) = y_i$ for any $i \in [n]$.*

Proof. We will first choose direction $w \in \mathbb{R}^d$, $\|w\|_2 = 1$ and margin $\gamma > 0$ such that for any $i \neq j$ in $[n]$, it holds that,

$$\left|\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle\right| \geq 2\gamma.$$

We will assume WLOG $w^\top \boldsymbol{x}_i$ is increasing in i.

Then we will construct an auxilliary series z_i for $i \in [n]$ such that,

$$z_1 = y_1/\gamma$$
$$z_i = y_i/\gamma - 2\sum_{j=1}^{i-1} z_j, i \in \{2, \dots n\}.$$

Finally consider the following two-layer MLP with ReLU activation,

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} z_i \mathrm{relu}\left(\langle w, \boldsymbol{x} - \boldsymbol{x}_i\rangle + \gamma\right) - z_i \mathrm{relu}\left(\langle w, \boldsymbol{x} - \boldsymbol{x}_i\rangle - \gamma\right),$$

we will show that $f(\boldsymbol{x}_i) = y_i$ for any $i \in [n]$. Notice that

$$z_j \mathrm{relu}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle + \gamma\right) - z_j \mathrm{relu}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle - \gamma\right) = \begin{cases} 0, & j > i, \\ \gamma z_i, & j = i, \\ 2\gamma z_j, & j < i. \end{cases}$$

Thus it holds,

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{n} z_j \mathrm{relu}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle + \gamma\right) - z_j \mathrm{relu}\left(\langle w, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle - \gamma\right)$$
$$= \sum_{j=1}^{i-1} 2\gamma z_j + \gamma z_i = y_i.$$

□

Lemma 39. *Given any sets $\{x_i\}_{i \in m}$ satisfying that $x_i \in \mathbb{R}^n$ and $x_i \neq 0$, there exists a set of orthonormal vectors $\{u_j\}_{j\in[n-2]}$ of $\mathbb{R}^n$ such that (1) $u_j^\top 1^n = 0$ for any $j \in [n-2]$ and (2) $\sum_{j\in[n-2]} u_j^\top x_i u_j \neq x_i$ for any $i \in [m]$.*

Proof. There exists a vector $v \in \mathbb{R}^n$ such that $v^\top x_i \neq 0$ for any $i \in [m]$. We can then construct an orthonormal basis $\{u_j\}_{j\in[n-2]}$ of $\mathbb{R}^n$ as the basis of the normal space of $\mathrm{span}(v, 1^n)$. Then the lemma holds. □

Lemma 40. *Given any dimension n and constant M, there exists a 2-layer width-$2n$ ReLU network $f : \mathbb{R}^{n+1} \to \mathbb{R}$ such that for any $x \in [0, M]^n, y \in [n]$, $f(x \oplus y) = x_y$.*

Proof. The construction is as followed, we will choose f as

$$f(x \oplus y) = \sum_{i=1}^{n} \mathrm{ReLU}(x_i + M(y-i)) - \sum_{i=1}^{n} \mathrm{ReLU}(x_i + M(y-i-1)) - M(y-1).$$

Then as we have

$$\mathrm{ReLU}(x_i + M(y-i)) - \sum_{i=1}^{n} \mathrm{ReLU}(x_i + M(y-i-1)) = \begin{cases} M, & i \leq y-1; \\ x_i, & i = y; \\ 0, & i \geq y+1. \end{cases}$$

The proof is completed. □

Lemma 41. *Given any dimension n and constant $M > 0$, there exists a 2-layer width-$2n$ ReLU network $f : \mathbb{R}^n \to \mathbb{R}$ such that for any $x \in \mathbb{R}^n$ satisfying there exists $i \in [n]$, $x_i > M$ and $\forall j \neq i, x_j = 0$, it holds that $f(x) = i$.*

Proof. The construction is as followed, we will choose f as

$$f(x) = \sum_{i=1}^{m} i \left(\mathrm{ReLU}(x_i) - \mathrm{ReLU}(x_i - M) + M\right)/M.$$

The proof is completed. □

Lemma 42. *Given any dimension n and natural numbers K, m, M, if there exists K different 2-layer width-m ReLU networks $f_k : \mathbb{R}^n \to \mathbb{R}$, then there exists a 2-layer width-$2Km$ ReLU network $f : \mathbb{R}^{n+1} \to \mathbb{R}$, such that $f(\begin{bmatrix} k \\ x \end{bmatrix}) = f_k(x)$ when $x \in [0, M]^n$.*

Proof. Suppose that

$$f_k(x) = \sum_{i=1}^{m} a_{k,i}\mathrm{ReLU}(w_{k,i}^\top x + b_{k,i}) + b_k.$$

Then we can construct

$$f(\begin{bmatrix} y \\ \\ x \end{bmatrix}) = \sum_{k=1}^{K}\sum_{i=1}^{m} a_{k,i}\mathrm{ReLU}(w_{k,i}^\top x + b_{k,i} + M(y-k)) - a_{k,i}\mathrm{ReLU}(w_{k,i}^\top x + b_{k,i} + M(y-k-1))$$
$$+ b_k - c_{k,i}\mathrm{ReLU}(y+1-k),$$

where $c_{k,i}$ satisfies

$$\forall i, k', \sum_{k=1}^{k'} c_{k,i}(k'+1-k) = M\sum_{k=1}^{k'-1} a_{k,i}.$$

The proof is then completed. □

Lemma 43. *Given any dimension n and $W \in \mathbb{R}^{n\times n}$, $\|W\|_2 \le 2$, there exists a 2-layer width-$2n$ ReLU network $f : \mathbb{R}^n \to \mathbb{R}$ such that for any $x \in \mathbb{R}^n$, it holds that $f(x) = Wx$ and both weight matrices parameterizing f has spectral norm less than $2\sqrt{2}$.*

Proof. The construction is straightforward, one can choose

$$f(x) = [I_n, -I_n]^\top \mathrm{ReLU}\left(\begin{bmatrix} Wx \\ \\ -Wx \end{bmatrix}\right).$$

□

3.6.5 Discussions: Are interpretable attention patterns useful?

Our results demonstrate that Transformers are sufficiently expressive that a (near-)optimal loss on Dyck languages can be achieved by a variety of attention patterns, many of which may not be interpretable.

However, multiple prior works have shown that for multi-layer multi-head Transformers trained on natural language datasets, it is often possible to locate attention heads that produce interpretable attention patterns [Vig and Belinkov, 2019, Htut et al., 2019, Sun and Marasović, 2021]. Hence, it is also illustrative to consider the *"converse question"* of (Q1): when some attention heads do learn to produce attention patterns that suggest intuitive interpretations, what benefits can they bring?

We discuss this through two perspectives:

- **Reliability of interpretation** (Section 3.6.5.1): Is the Transformer necessarily implementing a solution consistent with such interpretation based on the attention patterns?

- **Usefulness for task performance** (Section 3.6.5.2): Are those interpretable attention heads more important for the task than other uninterpretable attention heads?

We present preliminary analysis on these questions, and motivate future works on the interpretability of attention patterns using rigorous theoretical analysis and carefully designed experiments.

3.6.5.1 Can interpretable attention patterns be misleading?

We show through a simple argument that interpretations based on attention patterns can sometimes be misleading, as we formalize in the following proposition:

Proposition 6. *Consider an L-layer Transformer $\mathcal{T}$ (Equation equation 3.27). For any $W_K^{(l)}, W_Q^{(l)} \in \mathbb{R}^{m_a \times m}$ ($l \in [L]$), there exist $W_{\text{Head}} \in \mathbb{R}^{2k \times w}$ and $b_{\text{Head}} \in \mathbb{R}^{2k}$ such that $\mathcal{T}(\mathcal{Z}) = 0, \forall \mathcal{Z}$.*

While its proof is trivial (simply setting $W_{\text{Head}} = 0$ and $b_{\text{Head}} = 0$ suffices), Proposition 6 implies that the solution represented by the Transformer could possibly be independent of the attention patterns in all the layers (1 through l). Hence, it could be misleading to interpret Transformer solutions solely based on these attention patterns.

Empirically, Transformers trained on Dyck indeed sometimes produce misleading attention patterns.

We present one representative example in Figure 3.41, and Figure 3.42, in which *all interpretable attention patterns are misleading*. We also present additional results in Figure 3.43, in which *some interpretable attention patterns are misleading, and some are not*.

Similar message has been conveyed in prior works Bolukbasi et al. [2021], and future works may aim to achieve the *faithfulness*, *completeness*, and *minimality* conditions in Wang et al. [2023].

3.6.5.2 Are attention heads with interpretable patterns more important?

Kovaleva et al. [2019] observes that, when the "importance" of an attention head is defined as the performance drop the model suffers when the head is disabled, then for most tasks they test, the most important attention head in each layer *does not* tend to be interpretable.

However, experiments by Voita et al. [2019] led to a seemingly contradictory observation: when attention heads are systematically pruned by finetuning the Transformer with a relaxation of L_0-penalty (i.e. encouraging the number of remaining attention heads to be small), most remaining attention heads that survive the pruning can be associated with certain functionalities such as positional, syntactic, or attending to rare tokens.

These works seem to bring mixed conclusions to our question: are interpretable attention heads more important for a task than uninterpretable ones? We interpret these results by conjecturing that the definition of "importance" (reflected in their experimental design) plays a crucial role:

- When the importance of an attention head is defined *treating all other attention heads as fixed*, motivating experiments that prune/disable certain heads while keeping other heads unchanged [Michel et al., 2019, Kovaleva et al., 2019], the conclusion may be mostly pessimistic: mostly no strong connection between interpretability and importance.

- On the other hand, when the importance of an attention head is defined *allowing all other attention heads to adapt to its change*, motivating experiments that jointly optimize all attention heads while penalizing the number of heads [Voita et al., 2019], the conclusion may be more optimistic: the heads obtained as a result of this optimization tend to be interpretable.

We think the following trade-offs apply:

- On one hand, the latter setting is more practical, since Transformers are typically not trained to explicitly ensure that the model performs well when a single attention head is individually disabled; rather, it would be more intuitive to think of a group of attention heads as jointly representing some transformation, so when one head is disabled, other heads should be fine-tuned to adapt to the change.
- On the other hand, when all other heads change too much during such fine-tuning, the resulting set of attention heads no longer admit an unambiguous one-to-one map with the original set of (unpruned) attention heads. As a result, the interpretability and importance obtained from the set of pruned heads do not necessarily imply those properties of the original heads.

A comprehensive study of this question involves multi-head extensions of our theoretical results (Section 3.6.2), and carefully-designed experiments that take the above-mentioned trade-offs into consideration, which are interesting directions for future work.

3.7 Additional discussion of related work

Relevant applications. We first provide references for the “reasoning-like” applications of neural networks mentioned in this chapter.

- Program synthesis: [Chen et al., 2021b, Schuster et al., 2021, Li et al., 2022b].
- Mathematical reasoning: [Lample and Charton, 2019, Polu and Sutskever, 2020, Drori et al., 2022].
- Neural dynamics models for decision-making: recurrent [Hafner et al., 2019, Ye et al., 2021, Micheli et al., 2022] and non-recurrent [Chen et al., 2021a, Janner et al., 2021].

3.7.1 Formal languages and neural networks.

Dyck languages are particularly interesting for their completeness property: the Chomsky-Schützenberger representation theorem [Chomsky and Schützenberger, 1959] states that all context-free languages can be (homomorphically) represented by the intersection of a Dyck language and a regular language. For more on this topic, see the discussion in Yao et al. [2021b]. In the context of regular languages (which in general induce finite-state automata), our findings imply that $O(\log T)$-depth networks can simulate all context-free languages (Theorem 10), and $O(1)$-depth networks can represent some of them. The obstructing regular languages are the ones whose associated syntactic monoids are non-solvable. We further note that the gridworld semigroups are *aperiodic* and thus simulable by star-free regular expressions [Schützenberger, 1965] and AC^0 circuits [Chandra et al., 1983, Barrington and Thérien, 1988]. We did not see a way for this to generically entail $O(1)$-depth shortcuts with self-attention. For the relation between the Chomsky hierarchy and various neural networks *in practice*, Delétang et al. [2022] provide an extensive empirical study for memory-augmented RNNs and Transformers on tasks spanning all 4 levels of the hierarchy, and conclude the Transformers lack the ability to even recognize regular languages. Their results do not contradict with ours, since they measure performance on "inductive inference", which is similar to our length generalization setup where we also see the failure of Transformer.

3.7.2 Algebraic structures in deep learning.

Another area where tools from abstract algebra are used to reason about neural networks is *geometric deep learning*, a research program which seeks to understand how to specify inductive biases stemming from algebraic invariances. For a recent survey, see Bronstein et al. [2021]. In contrast, this work studies the ability of a fixed architecture to learn a wide variety of algebraic operations, in the absence of special priors (but a large amount of data). There are certainly possible connections (e.g. *"how do you bias an architecture to perform operations in a known group, when there is limited data?"*) to explore in future work.

3.7.3 Different axes of generalization: length, size, and algorithmic.

There has been much recent interest in quantifying out of distribution generalization of trained models under distribution shifts that maintain some notion of "logical" invariance. Wei et al. [2022c], Anil et al. [2022b] empirically investigate the ability of pre-trained Transformers to generalize to longer sequence length for parity-like problems modelled as language tasks. Xu et al. [2020] study size generalization in graph neural networks where they train on small graphs and evaluate on larger sized graphs with similar structural properties. Schwarzschild et al. [2021], Bansal et al. [2022] focus on length and algorithmic generalization for recurrent models where they train on simple/easy instances of the underlying problem and evaluate on harder/complex instances using the power of recurrence to simulate extra computational steps, inspired by the ideas of Neural Turing Machines [Graves et al., 2014] and Adaptive Computation Time [Graves, 2016]. We view our results as complementing those of Yao et al. [2021b], Anil et al. [2022b] for a richer class of problems. Our use of scratchpad is inspired by Nye et al. [2021a], Wei et al. [2022c], Anil et al. [2022b].

3.7.4 Transformers and variants

Recurrent Transformers. Our work is not the first to notice that Transformer architectures make brittle predictions out-of-distribution. Indeed, even the seminal paper introducing the architecture [Vaswani et al., 2017] notes that length generalization is promoted by a subtle hyperparameter choice (namely, the positional encoding scheme). Furthermore, there have been several attempts to reconcile this gap by modifying Transformers to behave more like RNNs; [Dehghani et al., 2019, Nye et al., 2021a, Wei et al., 2022c, Anil et al., 2022b, Hutchins et al., 2022]. Kasai et al. [2021] consider training a non-recurrent Transformer, and finetuning it into an RNN. All of these works have some element of natural language experiments: either the task is end-to-end language modeling, or the synthetic reasoning task is framed as a natural language problem, for a pretrain-finetune pipeline. We view our work as strengthening the foundations of these lines of inquiry. Theoretically, we provide structural guarantees for how shallow non-recurrent models can (perhaps deceptively) fit recurrent dynamics over long sequences. Empirically, we perform a *pure* (no confounds arising from the influence of a natural langauge corpus) analogue of the experiments seeking to help neural networks follow long chains of reasoning.

Recurrent vs. non-recurrent sequence transduction. As mentioned briefly towards the end of Section 3.4.3, the setting of indirectly-supervised semiautomata matches that of autoregressive generative modeling (a.k.a. next-token prediction), if the continuations of the sequence depend on the state of a latent semiautomaton. This is the case in (for example) generating Dyck languages [Yao et al., 2021b], where the possible continuations are $\{$all possible open brackets, if the stack q_t is not full$\} \cup \{$close bracket which pairs with the top of the stack $q_t\}$. We note that when an autoregressive model is used for sequence generation via a token-by-token inference procedure, this amounts to a special case of scratchpad inference (with a naive 1-step training procedure): the constant-depth network is used as a single iteration of a recurrent network, whose state is the completed prefix of the current generated sequence. Non-autoregressive natural language generation and transduction are an exciting area of research [Gu et al., 2017]; for a recent survey, see Xiao et al. [2022]. Our results are relevant to this line of work, suggesting that there may not be an expressivity barrier to expressing deep recurrent linguistic primitives, but there may be issues with out-of-distribution robustness.

Transformers as universal computation machines. Pérez et al. [2021] show that an infinite-precision Transformer achieves Turing completeness, as a single forward pass through a 3-layer decoder can simulate one transition step of a Turing machine. Giannou et al. [2023] exhibit a 13-layer Transformer whose weights are hard-coded to a universal Turing machine, and can be looped to perform any computation. These works show that one pass through a Transformer can implement a single computational step of a Turing machine. In contrast, our results show how a shallow Transformer can sometimes execute a computational loop over the entire context in a single non-recurrent pass. This requires a significantly more refined analysis, which depends on the global algebraic structure induced by the automata in question.

Theoretical role of depth. Our theoretical results can be interpreted as a *depth separation* result: contingent on $\mathsf{TC}^0 \neq \mathsf{NC}^1$, it takes strictly more layers to simulate non-solvable semiautomata, compared to their solvable counterparts. In a similar spirit, there have been several works establishing depth separation for feed-forward neural networks (mostly using ReLU activations) [Telgarsky, 2016, Eldan and Shamir, 2016, Daniely, 2017, Lee et al., 2017, Safran et al., 2019]. These results are usually constructive in nature, that is, they show the existence of functions that can be represented by depth L but would require exponential width for depth $L-1$ (or $\sqrt{L}$, depending on the result).

Universal function approximation with other networks. More elementary neural architectures, such as MLPs, have the ability to represent arbitrary functions, given sufficiently many neurons [Hornik et al., 1989, Cybenko, 1989]. The ACC^0 circuit construction described in [Barrington and

Thérien, 1988] can be implemented by any such architecture, not just the Transformer– there is a naive black-box way to "compile" each gate in the circuit into a network with the same depth as the ACC^0 circuit. A natural question is: *why, then, should we prefer Transformers?* The primary advantage of Transformers comes from the position-wise weight sharing of the attention layers and the casual structure from causal attention maps. Unlike MLPs, the shared parameters in Transformers allow for significant reduction in the total parameter count for representing the two main operations across positions: modular counters, and resets. In particular, these functions can be represented with $O(1)$ trainable parameters in the attention and MLP weight matrices (i.e. independent of T), as opposed to the $\Theta(T)$ parameters in a vanilla MLP, where position-wise parameter sharing is not available. In a sense, the Transformer architecture is naturally suited for implementing this construction.

3.7.5 Transformer optimization

Given multiple global optima, understanding Transformer solutions requires analyzing the training dynamics. Recent works theoretically analyze the learning process of Transformers on simple data distributions, e.g. when the attention weights only depend on the position information [Jelassi et al., 2022], or only depend on the content [Li et al., 2023]. Our work studies a syntax-motivated setting in which both content and position are critical. We also highlight that Transformer solutions are very sensitive to detailed changes, such as positional encoding, layer norm, sharpness regularization [Foret et al., 2020], or pre-training task [Liu et al., 2022b]. On a related topic but towards different goals, a series of prior works aim to improve the training process of Transformers with algorithmic insights [Nguyen and Salazar, 2019, Xiong et al., 2020, Liu et al., 2020, Zhang et al., 2021c, Li and Gong, 2021, *inter alia*]. An end-to-end theoretical characterization of the training dynamics remains an open problem; recent works that propose useful techniques towards this goal include Gao et al., 2023, Deng et al., 2023.

3.7.6 Generalization challenges

Hallucinations and long-range dependencies in NLP. The empirical literature is rife with corroborations that neural language models have trouble with robustly fitting long-range memory and multi-step reasoning [Khandelwal et al., 2018, Sun et al., 2021, Sukhbaatar et al., 2021, Malkin et al., 2022, Saparov and He, 2022, Orvieto et al., 2023, Creswell et al., 2023]. Such failures can result in "hallucinations": incorrect outputs which either directly contradict factual input in the context, or contain information absent in the context [Ji et al., 2023].

Hallucination can be attributed to various factors, such as the noisiness in data sources [Dhingra et al., 2019, Dziri et al., 2022], imperfect encoding/decoding [Parikh et al., 2020, Tian et al., 2019], or the discrepancy in training and evaluation setups [He et al., 2019]. In particular, the most related to our paper are the characteristics inherent to the model itself. For example, prior work has found that Transformers tend to be biased towards information covered during training [Petroni et al., 2019, Longpre et al., 2021], a potential cause to their poor out-of-distribution performance.

In terms of mitigation, various "external" methods (i.e. ones which do not modify the internal representations of the neural network) have been proposed to address some of the above factors, or post-processing model generations [Dziri et al., 2021, Chen et al., 2021c], possibly based on several forward passes [Wang et al., 2022b, Zheng et al., 2023a]. Another line of work that have gained much popularity and success is to incorporate explicit memory mechanisms, which we discuss next.

Explicit memory mechanisms in Transformers. Prior work has shown that augmenting the neural network with memory modules or knowledge base helps improve the performance on long-range texts [Khandelwal et al., 2019, Wu et al., 2022, Bertsch et al., 2023]. An approach particularly effective for large-scale Transformers is to ask the model to output immediate reasoning steps to a "scratchpad" which the model subsequently processes [Nye et al., 2021b, Wei et al., 2022b, Zhou et al., 2022, Anil et al., 2022b, Shao et al., 2023], similar to writing to and reading from a memory tape. A particular way to interact with the scratchpad is to interlace every other token with an annotation of "as a reminder, this is the state" [Liu et al., 2023a, Lanchantin et al., 2023], so that there are no more explicit long-range dependencies. However, this strategy is the same as the recurrent solution implementable by RNNs, and it does not always exist, especially when attention glitches occur in an internal component of the model.

Transformers and algorithmic tasks. Compared to real-world language datasets, synthetic tasks provide a cleaner and more controlled setup for probing the abilities and limitations of Transformers. Specific to algorithmic reasoning, Liu et al. [2023a] puts a unifying perspective on the ability of small Transformers to succeed at tasks corresponding to algorithmic primitives. Specific tasks of interest include hierarchical languages [Yao et al., 2021a, Zhao et al., 2023], modular prefix sums [Anil et al., 2022b], adders [Nogueira et al., 2021, Nanda and Lieberum, 2022], regular languages [Bhattamishra et al., 2020a], and following a chain of entailment Zhang et al. [2022].

3.7.7 Interpretability

Interpreting Transformer solutions Prior empirical works show that Transformers trained on natural language data can produce representations that contain rich syntactic and semantic information, by designing a wide range of "probing" tasks [Raganato and Tiedemann, 2018, Liu et al., 2019, Hewitt and Manning, 2019b, Clark et al., 2019b, Tenney et al., 2019, Hewitt and Liang, 2019, Kovaleva et al., 2019, Lin et al., 2019, Wu et al., 2020, Belinkov, 2022, Liu and Neubig, 2022] (or other approaches using the attention weights or parameters in neurons directly Vig and Belinkov, 2019, Htut et al., 2019, Sun and Marasović, 2021, Eldan and Li, 2023). However, there is no canonical way to probe the model, partially due to the huge design space of probing tasks, and even a slight change in the setup may lead to very different (sometimes even seemingly contradictory) interpretations of the result [Hewitt and Liang, 2019]. In this work, we tackle such ambiguity through a different perspective—by developing formal (theoretical) understanding of solutions learned by Transformers. Our results imply that it may be challenging to try to interpret Transformer solutions based on individual parameters [Li et al., 2016, Dar et al., 2022], or based on constructive proofs (unless the Transformer is specially trained to be aligned with a certain algorithm, as in Weiss et al., 2021).

Interpreting attention patterns Prior works [Jain and Wallace, 2019, Serrano and Smith, 2019, Rogers et al., 2020, Grimsley et al., 2020, Brunner et al., 2020, Prasanna et al., 2020, Meister et al., 2021, Bolukbasi et al., 2021, Haab et al., 2023, *inter alia*] present negative results on deriving explanations from attention weights using approaches by Vig and Belinkov [2019], Kobayashi et al. [2020, *inter alia*]. However, Wiegreffe and Pinter [2019] argues to the contrary by pointing out flaws in the experimental design and arguments of some of the prior works; they also call for theoretical analysis on the issue. Hence, a takeaway from these prior works is that expositions on explainability based on attention requires clearly defining the notion of explainability adopted (often task-specific). In our work, we restrict our main theoretical analysis to the fully defined data distribution of Dyck language (Definition 12), and define "interpretable attention pattern" as the stack-like pattern proposed in prior theoretical [Yao et al., 2021a] and empirical [Ebrahimi et al., 2020] works. These concrete settings and definitions allow us to mathematically state our results and provide theoretical reasons.

3.7.7.0.1 Mechanistic interpretability

It is worth noting that the challenges highlighted in our work do not contradict the line of prior works that aim to improve *mechanistic interpretability* into a trained model or the training process [Cammarata et al., 2020, Elhage et al., 2021, Olsson et al., 2022, Nanda et al., 2023, Chughtai et al., 2023, Li et al., 2023, Wang et al., 2023, Zhong et al., 2023]: although we prove that components (e.g. attention scores) of trained Transformers do not generally admit intuitive interpretations based on the data distribution, it is still possible to develop circuit-level understanding about a particular model, or measures that closely track the training process, following these prior works.

3.7.7.0.2 Interpretable machine learning

In even broader contexts of Interpretable Machine Learning in general, Lipton [2017] outlined common pitfalls of interpretability claims, Chen et al. [2022] recommended reasonable paths forward, and Bilodeau et al. [2022] proved impossibility results on applying some common classes of simple feature attribution methods on rich model classes.

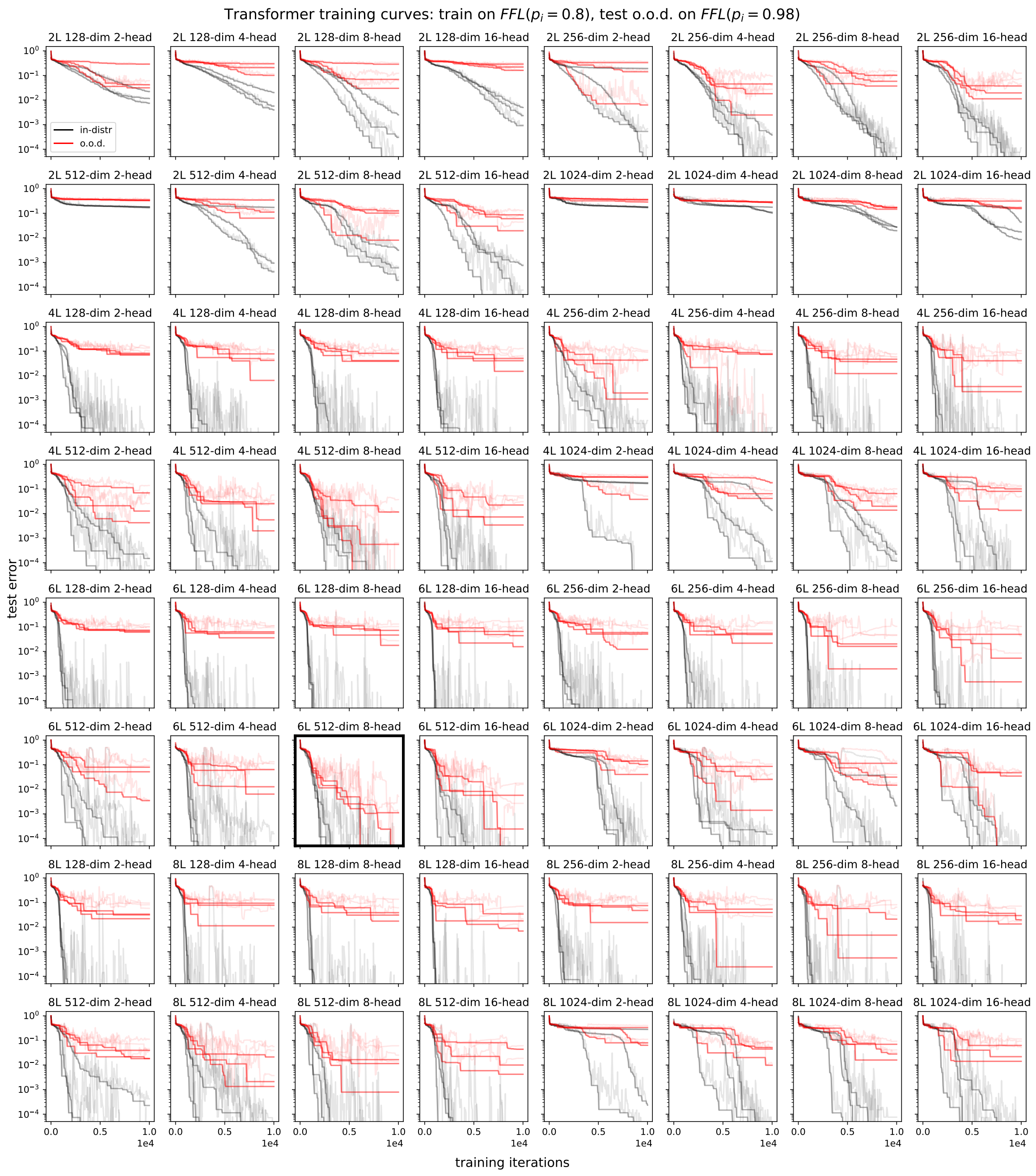


Figure 3.27: Examples of training curves over various Transformer architectures, ranging from 46K to 101M trainable parameters. We exhibit 3 (randomly selected) random seeds for each architecture. Lighter curves show raw error percentages, while solid curves denote the lowest error so far in each run. Notice the following: (1) **non-convergence of shallow models** (despite representability) (2) **failure of most runs to extrapolate** (i.e. reach 0% out-of-distribution error); (3) **high variability** between runs; (4) erratic, **non-monotonic progress** on out-of-distribution data, even when the in-distribution training curves appear flat; (5) **a small LSTM outperforms all of these Transformers** (see Figure 3.24). The $\boxed{\text{bolded box}}$ represents our 19M-parameter baseline model.

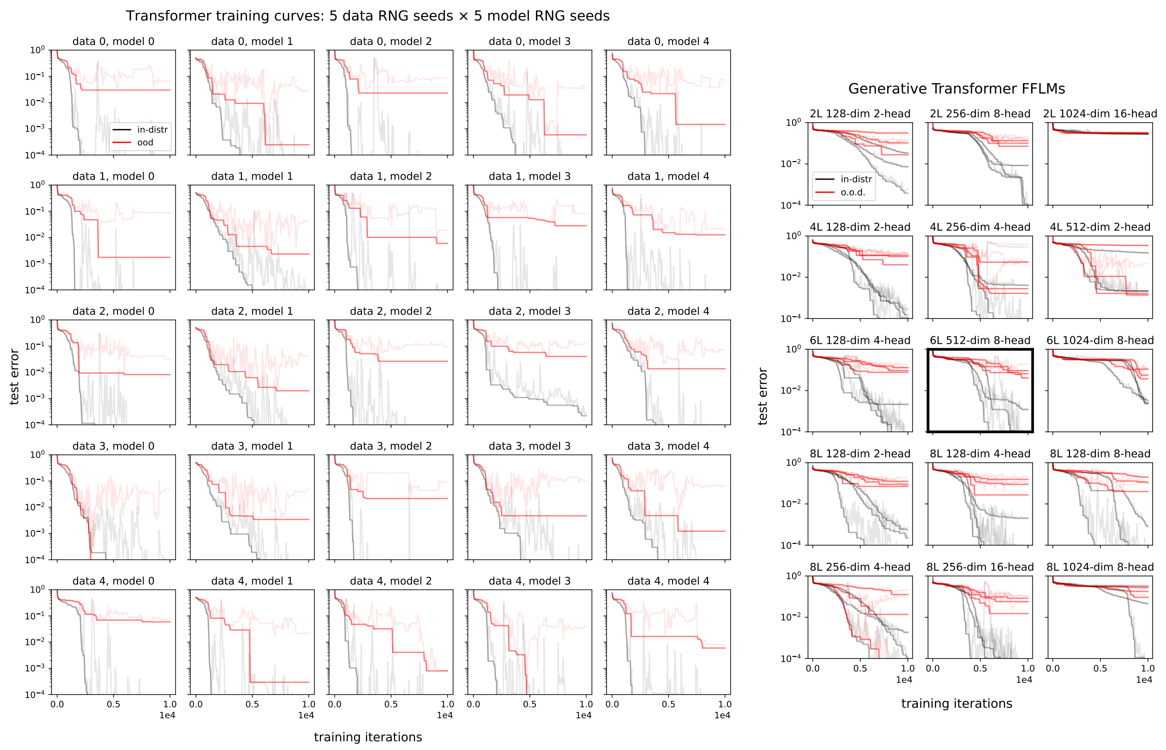
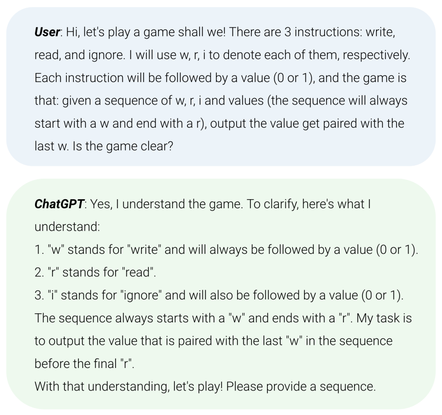
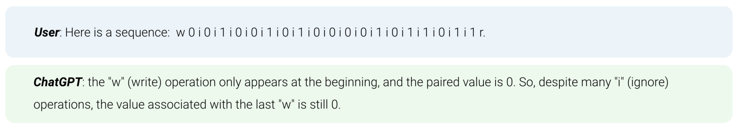


Figure 3.28: Additional training curves. *Left:* Identical baseline architecture, varying the 5 data seeds and 5 model seeds: models in the same row encounter the same sequence of data, while models in the same column start from identical initializations. **Both sources of randomness affect training dynamics and extrapolation**, and it is not clear which is more important. *Right:* Similar findings for models trained in "fully generative" mode (scoring on all tokens); baseline architecture is in the $\boxed{\text{bolded box}}$.


User: Hi, let's play a game shall we! There are 3 instructions: write, read, and ignore. I will use w, r, i to denote each of them, respectively. Each instruction will be followed by a value (0 or 1), and the game is that: given a sequence of w, r, i and values (the sequence will always start with a w and end with a r), output the value get paired with the last w. Is the game clear?

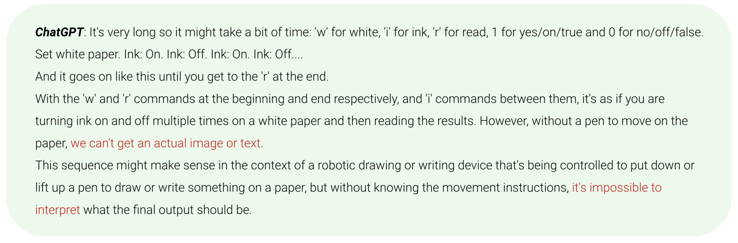
ChatGPT: Yes, I understand the game. To clarify, here's what I understand:
1. "w" stands for "write" and will always be followed by a value (0 or 1).
2. "r" stands for "read".
3. "i" stands for "ignore" and will also be followed by a value (0 or 1).
The sequence always starts with a "w" and ends with a "r". My task is to output the value that is paired with the last "w" in the sequence before the final "r".
With that understanding, let's play! Please provide a sequence.

(a) Explaining FFLM to ChatGPT.


User: Here is a sequence: w 0 i 0 i 1 i 0 i 0 i 1 i 0 i 1 i 0 i 0 i 0 i 1 i 0 i 1 i 1 i 0 i 1 i 1 r.

ChatGPT: the "w" (write) operation only appears at the beginning, and the paired value is 0. So, despite many "i" (ignore) operations, the value associated with the last "w" is still 0.

(b) Correct on short sequences.


ChatGPT: It's very long so it might take a bit of time: 'w' for white, 'i' for ink, 'r' for read, 1 for yes/on/true and 0 for no/off/false.
Set white paper. Ink: On. Ink: Off. Ink: On. Ink: Off....
And it goes on like this until you get to the 'r' at the end.
With the 'w' and 'r' commands at the beginning and end respectively, and 'i' commands between them, it's as if you are turning ink on and off multiple times on a white paper and then reading the results. However, without a pen to move on the paper, we can't get an actual image or text.
This sequence might make sense in the context of a robotic drawing or writing device that's being controlled to put down or lift up a pen to draw or write something on a paper, but without knowing the movement instructions, it's impossible to interpret what the final output should be.

(c) Wrong on long sequences (input length 1000).

Figure 3.29: Examples of interacting with ChatGPT-4 (as of 05/22/2023) by explaining FFLM to it.

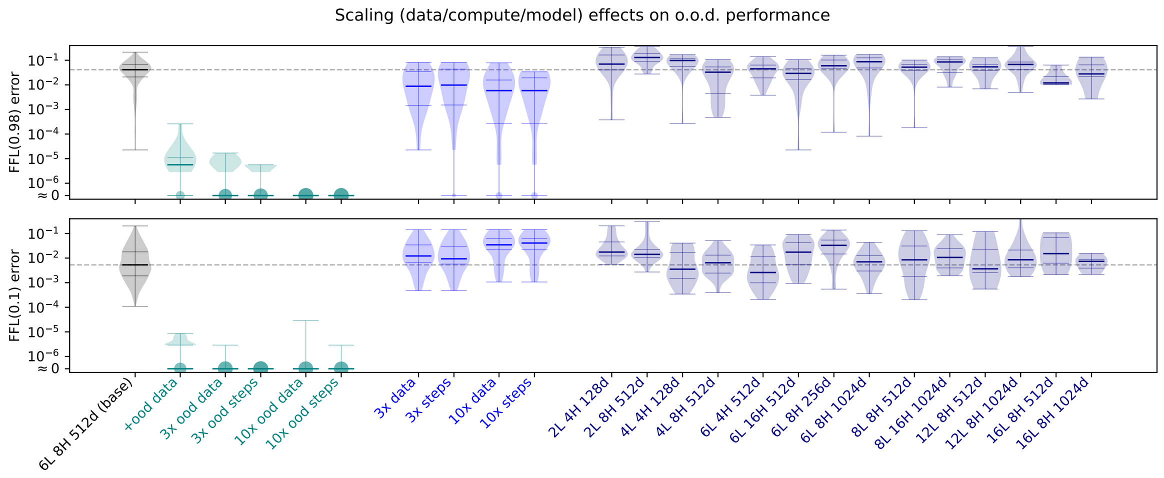


Figure 3.30: Full comparisons of various scaling axes. Increasing training data diversity is by far the most effective way to mitigate attention glitches in FFLM. The other scaling axes (increasing the amount of fresh data, increasing the number of optimization steps on the same dataset, and changing the model size) have mixed effects on rare-sequence performance.

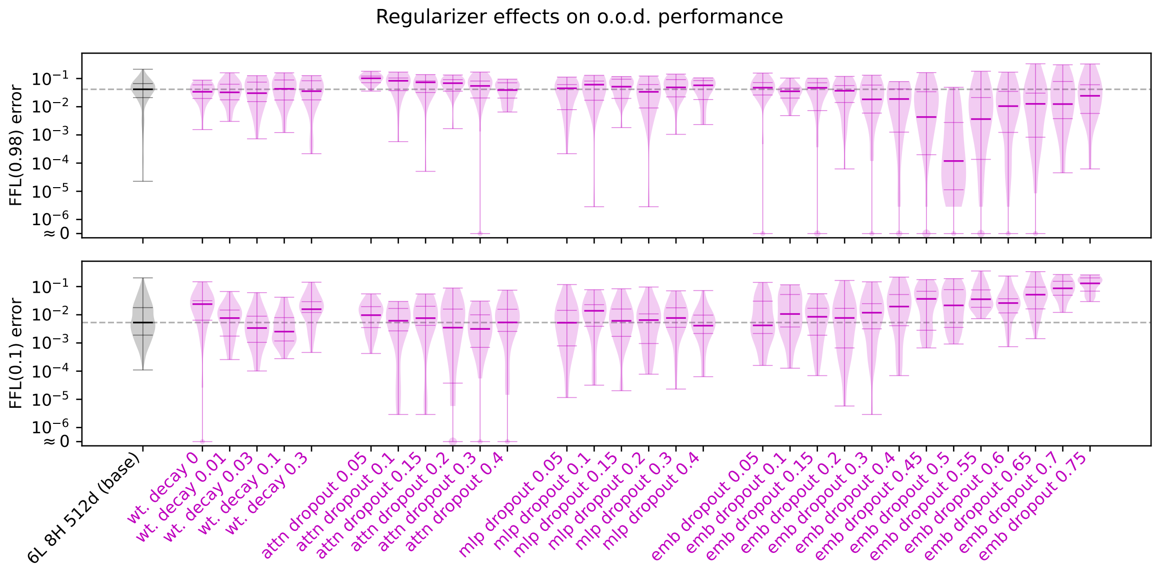


Figure 3.31: Full comparisons of standard regularizers (weight decay, and 3 forms of dropout). While some regularizer choices reduce rare-sequence error rates (in particular, large embedding dropout reduces sparse-sequence errors by 2 orders of magnitude), nothing eliminates them entirely.

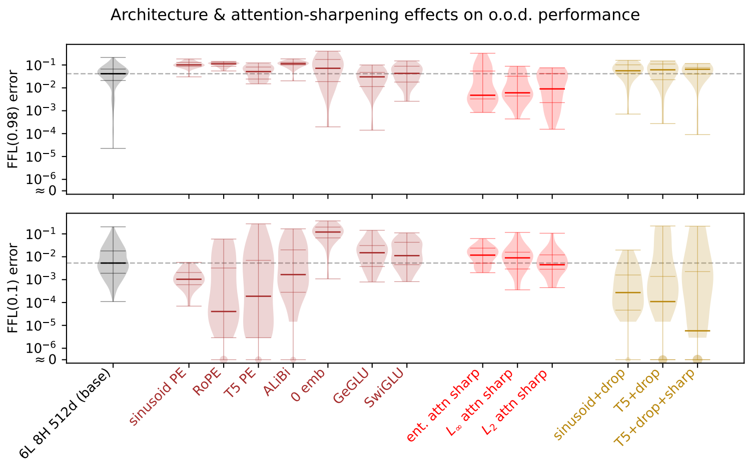


Figure 3.32: Full comparisons of architectural changes, attention-sharpening losses, and combinations of indirect algorithmic controls.

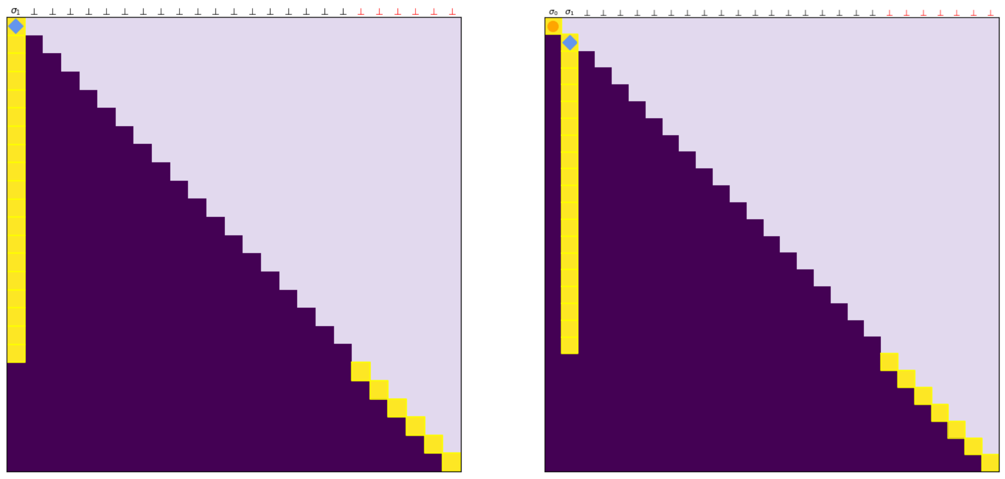
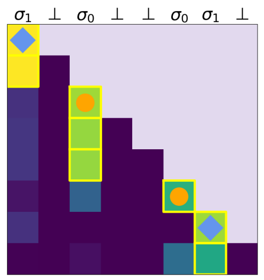
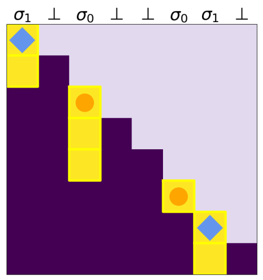


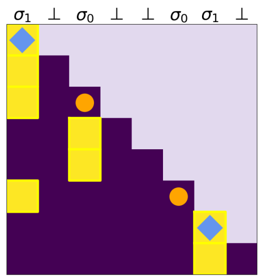
Figure 3.33: Attention drifts as the length increases. The model is trained on length-500 sequences with $p(\sigma \neq \perp) = 0.5$. The testing sequences are (a) $[2, \underbrace{0 \cdots, 0}_{800}]$, and (b) $[1, \underbrace{0 \cdots, 0}_{32}, 2, \underbrace{0 \cdots, 0}_{800}]$. We sample every 32 positions for visualization.



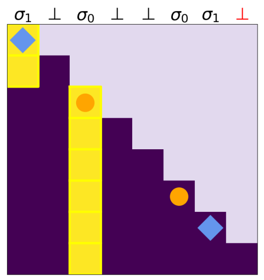
(a)



(b)

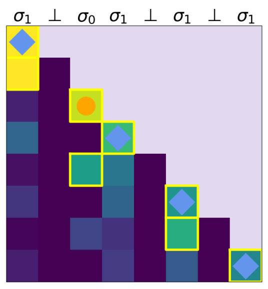


(c)



(d)

Figure 3.34: Attention-sharpening regularization on 1-layer 1-head models. Compared to a non-regularized model (3.34a), the sparsity-regularized model (3.34b) shows clear attention at the last write position. However, sparse attention does not have to align with the "ideal" pattern (3.34c), and can even be wrong (3.34d). Positions with yellow borders are where the max attention in each row occur; errors are marked in red.



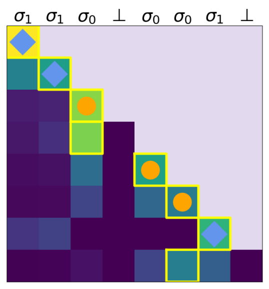
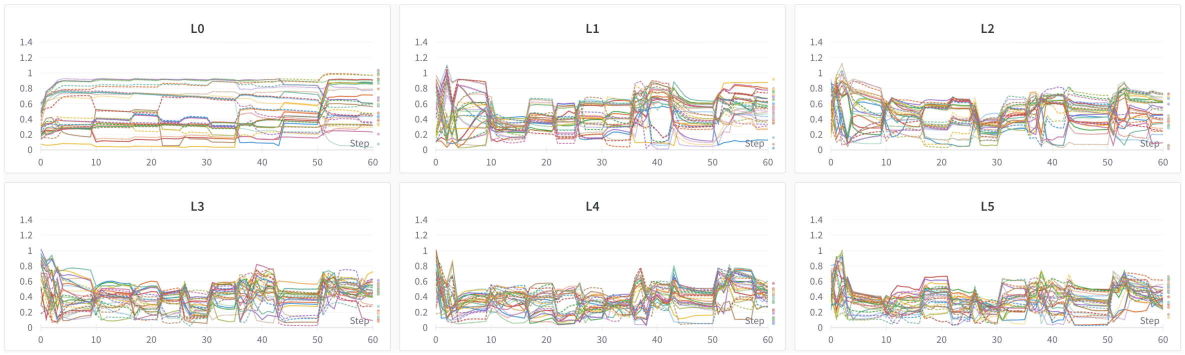
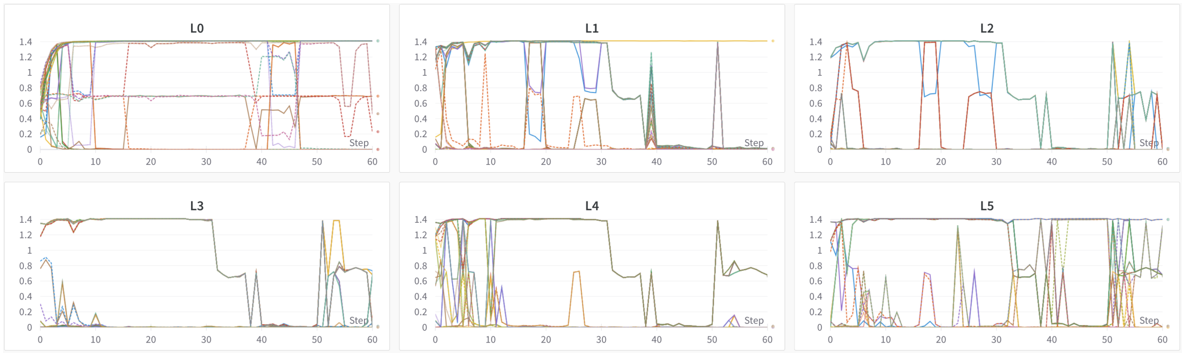


Figure 3.35: Non-sparse attention pattern can be misleading: a non-sparse model may put more attention on an incorrect token (i.e. a token that is not the `write` with the right type), while making the correct predictions. Yellow boxes mark the position of the max attention of each row.

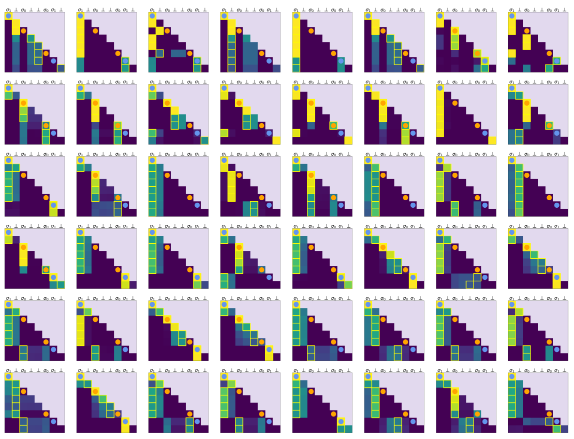


(a) Model trained without sparsity regularization.

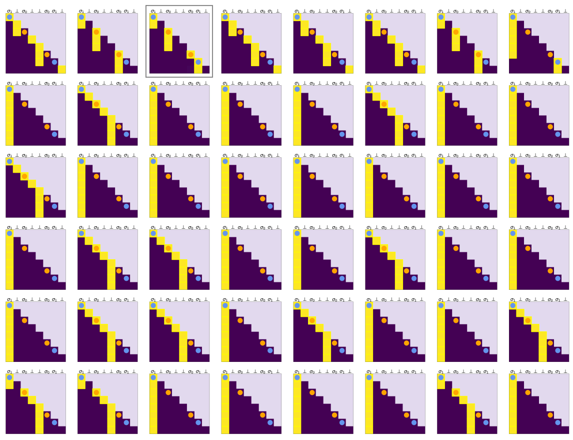


(b) Model trained with entropy sparsity regularization with $\lambda = 0.01$.

Figure 3.36: Examples of the ℓ_2 difference in attention patterns from two 6-layer 8-head 512-dimension models. Differences are calculated between all pairs of heads in the same layer.

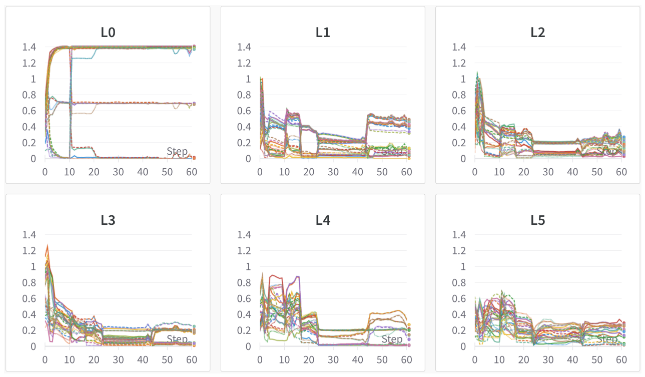


(a) Without regularization.

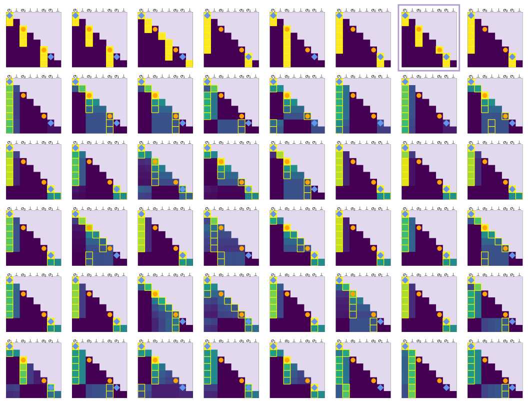


(b) With attention-sharpening regularization.

Figure 3.37: Attention patterns for 6-layer 8-head 512-dimension models on the input sequence $[\sigma_1, \perp, \sigma_0, \perp, \perp, \sigma_0, \sigma_1, \perp]$: attention-sharpening regularization lead to cleaner attention patterns. 1 attention head in the first layer of the regularized model (marked by the purple box) matches the "ideal" attention pattern Figure 3.26c.

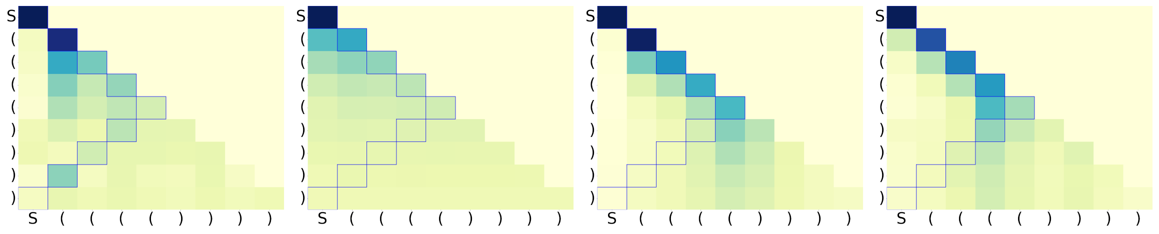


(a) ℓ_2 differences between pairs of attention heads in the same layer, throughout training (x-axis).



(b) Attention patterns on the input sequence $[\sigma_1, \perp, \sigma_0, \perp, \perp, \sigma_0, \sigma_1, \perp]$.

Figure 3.38: Attention heads and attention patterns for a 6-layer 8-head 512-dimension model, trained with attention-sharpening regularization (entropy regularization with strength 0.01) on the first layer only. 1 attention head in the first layer (marked by the purple box) matches the "ideal" attention pattern Figure 3.26c.



(a) Embedding 3.37, run 1 (b) Embedding 3.37, run 2 (c) Embedding 3.39 (d) Embedding 3.38

Figure 3.39: **Second-layer attention patterns of two-layer Transformers with a minimal first layer**: (a), (b) are based on embedding 3.37 with different learning rates, where the attention patterns show much variance as Theorem 1 predicts. (c), (d) are based on embedding 3.39 and 3.38. Different embedding functions lead to diverse attention patterns, most of which are not stack-like.

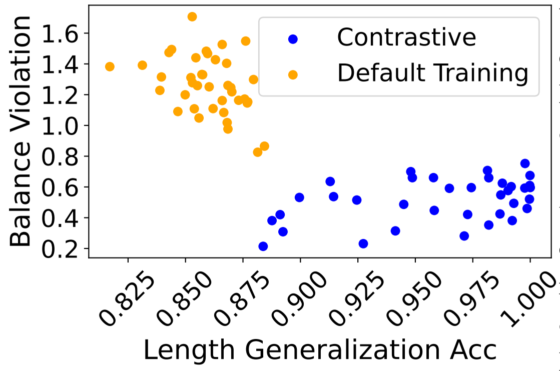


Figure 3.40: **Relationship Between Balance Violation and Length Generalization.** Accuracy from Transformers with minimal first layer with embedding 3.37, using both standard training and contrastive regularization (Equation (3.40)). Standard training leas to high balance violations which negatively correlate with length generalization performance. Contrastive regularization helps reduce the balance violation and improve the length generalization performance.

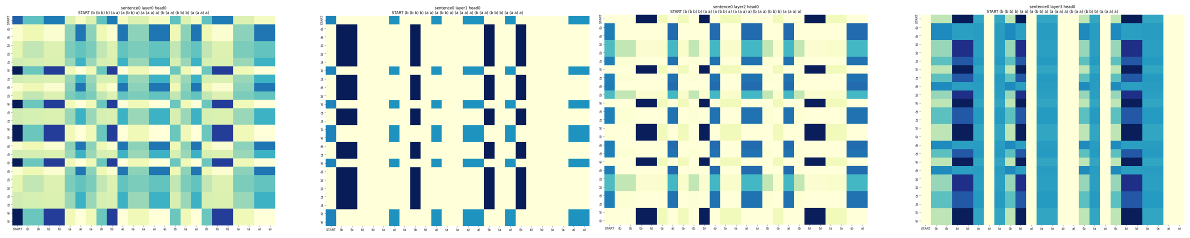


Figure 3.41: **Even interpretable attention patterns can be misleading**: For a 4-layer Transformer trained on Dyck with the *copying* task (with $>$ 96% validation accuracy), i.e. the output should be exactly the same as the input, the attention patterns in some layers seem interpretable: (layer 2) attending to bracket type a) or (b; (layer 3) attending to closing bracketss; (layer 4) neve attending to bracket type a); However, none of them are informative of the copying task. This is possible because Transformers can use the residual connections (or weights MLPs or the value matrices) to solve copying, bypassing the need of using attention.

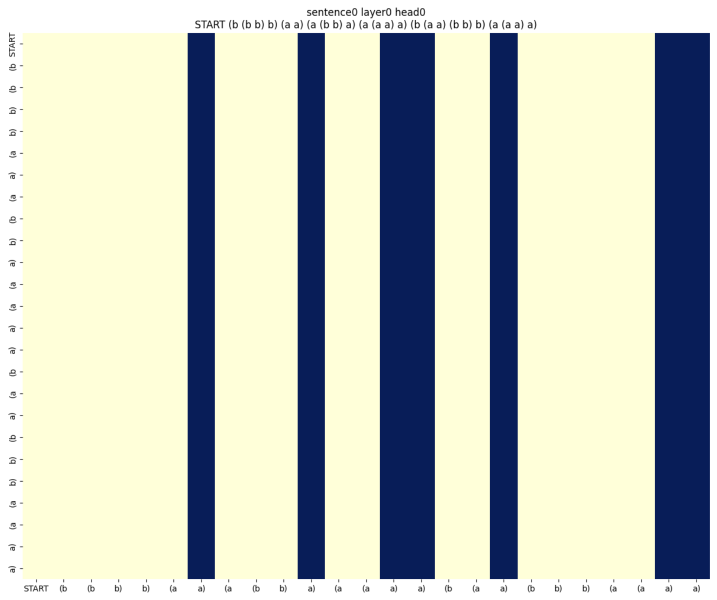
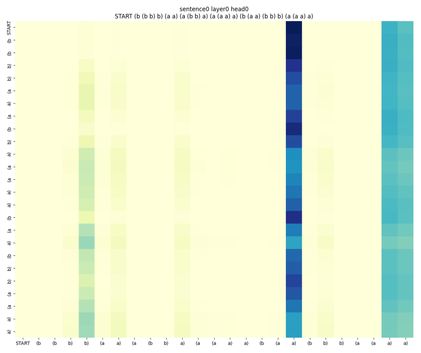
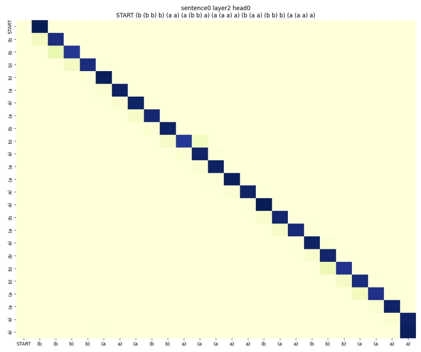


Figure 3.42: **Even interpretable attention patterns can be misleading**: For a 1-layer Transformer trained on Dyck with the *copying* task (with $>$ 90% validation accuracy), i.e. the output should be exactly the same as the input, the attention pattern seems to be attending to closing brackets only, but that is not informative of the copying task.



(a) layer 1 of 4



(b) layer 3 of 4

Figure 3.43: **Even interpretable attention patterns can be misleading**: For a 4-layer Transformer trained on Dyck with the *copying* task (with $>$ 96% validation accuracy), i.e. the output should be exactly the same as the input, both types of attention patterns are common: (a) attending to closing bracketss, which is uninformative of the copying task; (b) attending to the current position, which solves the copying task.

Chapter 4

Understanding and improving the learning process

Chapter 2 and Chapter 3 study properties of the *optimal* solution of a task. In practice though, the solution is found through a gradient-based search using finite samples. Since the learning problem is highly non-convex for most problems of interest, the solution that we end up with can differ drastically depending on various factors of the learning process. This chapter considers three such factors: the loss function, the gradient update step, and the supervision signals used in training. We will show how modifying these factors can accelerate training.

4.1 Improving the optimization landscape of noise-contrastive estimation

Noise contrastive estimation (NCE) is a method for learning parameterized statistical models [Gutmann and Hyvärinen, 2010a, 2012]. To estimate a distribution P_*, NCE trains a discriminant model to distinguish between samples of P_* and a known distribution Q of our choice, often referred to as the "noise" distribution. If the function class for the discriminant model is representationally powerful enough, the optimal model learns the density ratio p_*/q, from which we can extract the density p_* since q is known [Menon and Ong, 2016, Sugiyama et al., 2012]. Compared to the well-studied maximum likelihood estimation (MLE), NCE avoids calculating the (often intractable) partition function, while maintaining the asymptotic consistency of MLE [Gutmann and Hyvärinen, 2012].

It is empirically well-documented that the choice of the noise distribution Q is crucial to both the statistical and algorithmic efficiency of NCE [Gutmann and Hyvärinen, 2010a, 2012, Rhodes et al., 2020, Goodfellow et al., 2014, Gao et al., 2020]. However, it has been observed in practice that even when following the standard guidelines for choosing Q, NCE can still yield parameter estimates far from the ground truth [Rhodes et al., 2020, Goodfellow et al., 2014, Gao et al., 2020]. Most recently, Rhodes et al. [2020] identified a phenomenon they call the "density chasm," observing empirically that NCE performs poorly when the KL divergence between P_* and Q is large. One example is when

P_*, Q are both tightly concentrated unimodal distributions with faraway modes; the region between the two modes will have a small density under both distributions, thus forming a "chasm". While it makes intuitive sense that NCE does not perform well under such settings—since disparate Q and P_* are easy to distinguish and do not require the model to learn much about P_* in order to do well on the classification task—there has not been a theoretical analysis of this phenomenon. In fact, it is not even clear whether the difficulty is statistical or algorithmic in nature.

In this work, we formally study the challenges for NCE with a fixed Q with a focus on distributions in an exponential family. We show that when the noise distribution Q is poorly chosen, the loss landscape can become extremely flat: in particular, even when P^* and Q are two univariate Gaussian with unit variance, the loss gradient and curvature can become exponentially small in the difference in their means. We prove that this poses challenges for standard first order and even second-order optimization methods, forcing them to take an exponential number of steps to converge to a good parameter estimate. Thus, standard approaches to minimizing convex functions such as gradient descent—or even more advanced techniques such as momentum or Newton's method—are not suited to the NCE objective unless Q is close to P_* in KL sense.

To remedy this issue, we study an alternative method for optimizing the NCE objective. We consider instead Normalized Gradient Descent (NGD) whereby the gradient is normalized to have unit norm at each time step. Perhaps surprisingly, we prove that this small modification can overcome the problem of poor curvature in the Gaussian example. In general, we show the number of steps for NGD to converge to a good solution for the NCE loss depends on the *condition number* κ of the Hessian of the loss at the optimum—the growth of this condition number is unclear for P^* and Q when they belong to an exponential family.

To address this, we propose the *eNCE* loss, a variant to NCE that replaces the log loss in NCE with an exponential loss, and we show that the resulting condition number is polynomial in the dimension and the parameter distance between P^* and Q when they belong to an exponential family. Our proposed change of loss and optimization algorithm *together* form the first solution that provides a provable polynomial rate for learning the parameters of the ground truth distribution. Theoretically, both NCE and eNCE can potentially suffer from numerical issues during optimization when P^* and Q are far—this is an interesting direction for future work. Nonetheless, we find this to be a simple and effective fix to the flatness of the loss landscape in many settings, as evidenced by experimental results on synthetic and MNIST dataset.

Related Work NCE and its variants have inspired a large volume of research in NLP [Mnih and Teh, 2012, Mnih and Kavukcuoglu, 2013, Dyer, 2014, Kong et al., 2020] as well as computer vision [Oord et al., 2018, Hjelm et al., 2018, Henaff, 2020, Tian et al., 2020a]. It has been observed empirically that NCE with a fixed noise Q is often insufficient for learning good generative models. The predominant class of approaches that have been proposed to overcome this issue aim to do so by not using a fixed Q but by iteratively solving multiple NCE problems with an updated Q, or equivalently updated discriminators. This includes the famous generative adversarial network (GAN) by Goodfellow et al. [2014], which uses a separate discriminator network updated throughout training. In a similar vein, Gao et al. [2020] also aimed to increase the discriminative power as the density estimator improves, and parameterize Q explicitly with a flow model. More recently, Rhodes et al. [2020] proposed the telescoping density ratio estimation, or TRE, which sidesteps the chasm by expanding p_*/q into a series of intermediate density ratios, each of which is easier to estimate, leading to strong empirical performance—though their work carries no formal guarantees.

With respect to a fixed Q, it remains an open question about what formally are the nature of the challenges posed by a poorly chosen Q, which could be statistical and/or algorithmic. Various previous works have analyzed the asymptotic behavior of NCE and its variants [Gutmann and Hyvärinen, 2012, Riou-Durand et al., 2018, Uehara et al., 2020], but these do not provide guidance on the finite sample behavior of NCE or its common variants. The improvements to NCE in prior works are all borne out by the empirical observations of NCE practitioners, rather than motivated by theory, which is precisely the aim of this work.

4.1.1 Preliminaries

4.1.1.1 The NCE objective

Let P_* denote an unknown distribution in a parametric family $\{P_\theta\}_{\theta\in\Theta}$, for some bounded convex set Θ, with $P_* = P_{\theta_*}$. Our goal is to estimate P_* via P_θ for some $\theta \in \Theta$ by solving a noise contrastive estimation task. The noise distribution Q belongs to the same parametric family with parameters $\theta_q \in \Theta$, so that $Q = P_{\theta_q}$. We use p_θ, p_*, q to denote the probability density functions (pdfs) of P_θ, P_*, and Q; we may omit θ in P_θ, p_θ when it is clear from the context and write P, p instead. Given P_* and Q, the NCE loss of P is defined as follows:

Definition 19 (NCE Loss)**.** *The NCE loss of P_θ w.r.t. data distribution P_* and noise Q is:*

$$L(P_\theta) = -\frac{1}{2}\mathbb{E}_{P_*} \log \frac{p_\theta}{p_\theta + q} - \frac{1}{2}\mathbb{E}_Q \log \frac{q}{p_\theta + q} \tag{4.1}$$

Note that the NCE loss can be interpreted as the binary cross-entropy loss for the binary classification task of distinguishing the data samples from the noise samples. Moreover, the NCE loss has a unique minimizer:

Lemma 44 (Gutmann and Hyvärinen 2012)**.** *The NCE objective in Definition 19 is uniquely minimized at $P = P_*$.*

4.1.1.2 Exponential family.

We focus our attention on the exponential family, where the pdf for a distribution with parameter θ is $p_\theta(x) = \exp\left(\theta^\top \tilde{T}(x) - A(\theta)\right)$, with $\tilde{T}(x)$ denoting the sufficient statistics and $A(\theta)$ the log partition function. [1] The partition function is treated as a parameter in NCE, so we use τ to denote the extended parameter, i.e. $\tau := [\theta, \alpha]$ where α is the estimate for the log partition function. We accordingly extend the sufficient statistics as $T(x) = [\tilde{T}(x), -1]$ to account for the log partition function. The pdf with the extended representation is now simply $p_\tau(x) = \exp(\tau^\top T(x))$. We will use the notation P_θ and P_τ interchangeably. We will also use $\tau(\theta)$ to denote the log-partition extended parameterization when the log partition function α properly normalizes the distribution specified by θ.

[1] Another common format of the exponential family PDF is $p_\theta(x) = h(x)\exp\left(\theta^\top T(x) - A(\theta)\right)$ where $h(x)$ is a non-negative function. Such $h(x)$ could be absorbed into $\tilde{T}(x)$ and θ with corresponding coordinates $\log(h(x))$ and 1.

A compelling reason for focusing on the exponential family is the observation that the NCE loss is convex in the parameter τ:

Lemma 45 (NCE convexity)**.** *For exponential family $p_{\theta,\alpha}(x) = h(x)\exp(\theta^\top \tilde{T}(x) - \alpha)$, the NCE loss is convex in parameter $\tau := [\theta, \alpha]$.*

Proof. We note that Lemma 45 has been stated under more general settings by Uehara et al. [2020]. The following is an alternative proof for completeness. The gradient and Hessian of the NCE loss are:

$$\begin{aligned}
\nabla_\tau p(x) =& p(x) \cdot T(x), \\
\nabla L(\tau) =& \frac{1}{2}\nabla\left[\mathbb{E}_* \log\frac{p+q}{p} + \mathbb{E}_Q \log\frac{p+q}{q}\right] \\
=& \frac{1}{2}\left[\mathbb{E}_* \frac{p}{p+q}\frac{p-p-q}{p^2}\nabla_\tau p + \mathbb{E}_Q \frac{q}{p+q}\frac{1}{q}\nabla_\tau p\right] = \frac{1}{2}\int_x \frac{q}{p+q}(p-p_*)T(x)dx, \qquad (4.2) \\
\nabla^2 L(\tau) =& \frac{1}{2}\int_x \left(-\frac{q(p-p_*)}{(p+q)^2}\nabla_\tau p + \frac{q}{p+q}\nabla_\tau p\right) T(x)dx \\
=& \frac{1}{2}\int_x \frac{q}{p+q}\cdot\frac{p_*+q}{p+q}\cdot p \cdot T(x)T(x)^\top dx = \frac{1}{2}\int_x \frac{(p_*+q)pq}{(p+q)^2} T(x)T(x)^\top dx.
\end{aligned}$$

Hence the Hessian is PSD at any τ. □

Recall that Θ denotes the set of parameters without the extended coordinate for the log partition function. We assume the following on distributions supported on Θ:

Assumption 11 (Bounded parameter norm)**.** $\|\theta\|_2 \le \omega, \forall \theta \in \Theta$.

Assumption 12 (Lipschitz log partition function)**.** *Assume the log partition function is β_Z-Lipschitz, that is, $\forall \theta_1, \theta_2 \in \Theta$, $|\log Z(\theta_1) - \log Z(\theta_2)| \le \beta_Z \|\theta_1 - \theta_2\|$.*

Assumption 13 (Bounded singular values of the population Fisher matrix)**.** *There exist $\lambda_{\max}, \lambda_{\min} > 0$, such that $\forall \theta \in \Theta$, we have $\sigma_{\max}(\mathbb{E}_\theta[T(x)T(x)^\top]) \le \lambda_{\max}$, and $\sigma_{\min}(\mathbb{E}_\theta[T(x)T(x)^\top]) \ge \lambda_{\min}$.*

Assumption 14 (Smooth change in the Fisher matrix)**.** *Assume the maximum and minimum singular values of the Fisher matrix change smoothly. Namely, there exist constants $\gamma_{\max}, \gamma_{\min} > 0$ s.t.*

$$\|\nabla_\theta \sigma_{\max}(\mathbb{E}_\theta[T(x)T(x)^\top])\| \le \gamma_{\max}, \ \|\nabla_\theta \sigma_{\min}(\mathbb{E}_\theta[T(x)T(x)^\top])\| \le \gamma_{\min}$$

We note that Assumptions 12-14 can be viewed as smoothness assumptions on the first, second and third order derivatives of the log partition function. In particular, Assumption 13 says the singular values of the Fisher matrix $\mathbb{E}_\theta[T(x)T(x)^\top]$ should be bounded from above and below. It can be shown that the Fisher matrix is proportional to the Hessian of the NCE objective when using $Q = P_*$, which means Assumption 13 can be interpreted as saying the NCE task can be solved efficiently under the optimal choice of Q.

4.1.2 Overview of results

We first provide an informal overview of our results, focusing on learning of exponential families.

Flatness of population landscape: Our first contribution is a negative result identifying a key source of difficulty for NCE optimization to be an ill-behaved *population* landscape. We show that due to an extremely flat landscape, gradient descent or Newton's method with *standard choices* of step sizes will need to take an exponential number of steps to find a reasonable parameter estimate.

We emphasize that though Gaussian mean estimation is a trivial task, its simplicity *strengthens the results above*: we are proving a *negative* result so that failures with a simpler setup means a stronger result. Moreover, the results only apply to *standard* choices of step sizes, such as inversely proportional to the smoothness for gradient descent, or to the ratio between the smoothness and strong convexity for Newton's method. This does not rule out the possibility that a cleverly designed learning rate schedule or a different algorithm would work efficiently; the results are however still meaningful since gradient descent with standard step sizes is the most common choice in practice.

Overcoming flatness using normalized gradient descent: Our second contribution is to show that the flatness problem can be solved by a simple modification to gradient descent if the loss is well-conditioned. Specifically, we show that the convergence rate for *normalized gradient descent* is polynomial in the parameter distance and κ_*, the *condition number* of the Hessian at the optimum. One immediate consequence is that for Gaussian mean estimation, NCE optimized with NGD achieves a rate of $O(\frac{1}{\delta^2})$, which is the same as the optimal rate achieved by MLE.

The remaining question is then whether κ_* is polynomial in the parameters of interests. We show that κ_* can be related to the Bhattacharyya coefficient between P_* and Q, which indeed grows polynomially in parameter distance under certain assumptions as detailed in Section 4.1.4.2.

Polynomial condition number for the eNCE loss: Our third and final contribution is that if we modify the NCE objective slightly—namely, use the exponential loss in place of the log loss—then the condition number at the optimum is guaranteed to be polynomial. We call this new objective *eNCE* . Combined with the NGD result, we get that running NGD on the eNCE objective achieves a polynomial convergence guarantee.

We then provide empirical evidence on synthetic and MNIST dataset that eNCE with NGD performs comparatively with NGD on the original NCE loss, and both outperform gradient descent.

4.1.3 A challenge in NCE optimization: a flat loss landscape

In this section, we study the challenges posed to NCE when using a badly chosen fixed Q. The main thrust of the results is to show that both algorithmic and statistical challenges can arise because the NCE loss is *poorly behaved*, particularly for first- and second-order optimization algorithms: when P_*, Q are far, the loss landscape is extremely flat near the optimum. In particular, the gradient has exponentially small norm and the strong convexity constant decreases exponentially fast, limiting the convergence rate of the excess risk. We further show that when moving from $P = Q$ to $P = P_*$, the loss drops from $\Theta(1)$ to a value that is exponentially small in terms of the distance between P_* and Q. Consequently, common gradient-based and second order methods will take exponential number of steps to converge.

An important note is that our analysis is at the population level, implying that the hardness comes from the landscape itself regardless of the statistical estimators used.

Setup: Gaussian mean estimation For the negative results in this section, let's consider an exceedingly simple scenario of 1-dimensional, fixed-variance Gaussian mean estimation. We will demonstrate the difficulty of achieving a good parameter estimate, even for such a simple problem—this bodes ill for NCE objectives corresponding to more complex models in practice, which certainly pose a much more difficult challenge. In particular, let P_*, Q, P be Gaussians with identity variance. Let $\theta_*, \theta_q, \theta$ denote the respective means, with θ_* being the target mean that NCE aims to estimate. When the covariance is known to be 1, we can denote $h(x) := \exp\left(-\frac{x^2}{2}\right)$, and parametrize the pdf of a 1d Gaussian with mean θ as $p(x) = h(x)\exp\left(\langle\tau(\theta), T(x)\rangle\right)$,[2] where the parameter is $\tau(\theta) := [\theta, \frac{\theta^2}{2} + \log\sqrt{2\pi}]$ and the sufficient statistics are $T(x) := [x, -1]$. [3] We will shorthand $\tau(\theta)$ when it is clear from the context. In particular, $\tau_* := \tau(\theta_*) = [R, -\frac{R^2}{2} - \log\sqrt{2\pi}]$, and $\tau_q := \tau(\theta_q) = [0, \log\sqrt{2\pi}]$.

Without loss of generality, we will assume $\theta_q = 0$, and $\theta_* > 0$. As a clarification, the results stated in this section will be in terms of $R := \theta_* - \theta_q$, hence the asymptotic notations Ω, O never hide dominating dependency on R. [4]

4.1.3.1 Properties of the NCE loss

We first describe several properties of the NCE loss that will be useful in the analysis of first- and second-order algorithms.

To start, we show that the dynamic range of the loss is large: that is, the optimal NCE loss is exponentially small as a function of R; on the other hand, if θ is initialized close to θ_q, the initial loss would be on the order of a constant. Precisely:

Proposition 7 (Range of NCE loss)**.** *Consider the 1d Gaussian mean estimation task with mean $\theta_*, \theta_q \in \mathbb{R}$, and a known variance of 1. Denote $R := |\theta_q - \theta_*|$ where $R \gg 1$, Then, the loss at $\theta = \theta_q$ is $\log 2$, while the minimal loss L_* is $L_*(R) = c\exp(-R^2/8)$ for some $c \in [\frac{1}{2}, 2]$.*

The next shows we *need to* decrease the loss to be on an order comparable to the optimum value. Namely, the loss is very flat close to θ_*, thus in order to recover a θ close to θ_*, we have to reach a very small value for the loss. Precisely:

Proposition 8. *Under the same setup as Proposition 7, for a given $\delta \in (0, 1)$, if the learned parameter τ satisfies $\|\tau - \tau^*\|_2 \le \delta$, then $L(\tau) - L(\tau^*) = R\exp(-R^2/8)\,\delta^2$.*

The way we will leverage Propositions 7 and 8 to prove lower bounds is to say that if the updates of an iterative algorithm are too small, the convergence will take an exponential number of steps.

Proposition 8 is proven via the Taylor expansion at θ^*: since the gradient is 0 at θ_*, we just need to bound the Hessian at θ_*. We show:

[2]Thus, we are setting h to be the base measure for the exponential family we are considering.

[3]Recall that the last coordinate -1 acts as a sufficient statistic for the log partition function.

[4]For example, for $R \gg 1$, $R\exp(R^2) = O(\exp(R^2))$, but the constant in $O(1)$ will not depend on R.

Lemma 46 (Smoothness at $P = P^*$)**.** *Under the same setup as Proposition 7, the smoothness at $P = P_*$ is upper bounded as $\sigma_{\max}(\nabla^2 L(\tau_*)) \leq \frac{R}{\sqrt{2\pi}} \exp(-R^2/8)$.*

We will also need a bound on the strong convexity constant (i.e. smallest singular value) at $P = P^*$:

Lemma 47 (Strong convexity at $P = P^*$)**.** *Under the same setup as Proposition 7, the minimum singular value at $P = P_*$ is $\sigma^*_{\min}(\nabla^2 L(\tau_*)) = \Theta\left(\frac{1}{R} \exp\left(-\frac{R^2}{8}\right)\right)$.*

Finally, in order to estimate the choice of the step size for standard optimization methods, we will also need a bound of the smoothness at $P = Q$:

Lemma 48 (Smoothness at $P = Q$)**.** *Under the same setup as Proposition 46, the smoothness at $P = Q$ is lower bounded as $\sigma_{\max}(\nabla^2 L(\tau_q)) \geq \frac{R^2}{2}$.*

The proofs of Lemma 46, 47 are included in Appendix 4.1.8, and the proof of Lemma 48 is in Appendix 4.1.9.

4.1.3.2 Lower bounds on first- and second-order methods

With the landscape properties at hand, we are now ready to provide lower bounds for both first-order and second-order methods. For first-order methods, we show that:

Theorem 17 (Lower bound for gradient-based methods)**.** *Let P_*, Q, P be 1d Gaussian with variance 1. Assume $\theta_q = 0, \theta_* > 0$ without loss of generality, and assume $R := \theta_* - \theta_q \gg 1$. Then, gradient descent with any step size $\eta = o(1)$ from an initialization $\tau = \tau_q$ will need an exponential number of steps to reach some τ' that is $O(1)$ close to τ_*.*

Note, the maximum step size $\eta = o(1)$ the theorem applies to is actually a loose bound: the standard setting of step size for gradient descent is $\eta \leq 1/\lambda_M$ for $\lambda_M := \max_{\theta \in \Theta} \sigma_{\max}(\nabla^2 L(\tau(\theta)))$, which is $\Omega(R^2)$ by Lemma 48. Theorem 17 helps explain why NCE with a far-away Q fails in practice, if we set the budget for the number of updates to be polynomial.

The idea behind the proof is to first show that there exists an annulus $\mathcal{A}$ around the target τ_* such that τ_q, τ_* lie in the outer and inner side of $\mathcal{A}$ (see Figure 4.1), and that gradient descent needs to cross a distance of at least $0.05R$ inside $\mathcal{A}$. Then, due to the choice of step size and the magnitude of the gradients, the number of steps required to do so is exponentially large.

Proof of Theorem 17. The key lemma to prove Theorem 17 is as follows, which upper bounds the decrease in parameter distance from each gradient step:

Lemma 49. *Consider the annulus $\mathcal{A} := \{(b, c) : (c - \frac{R^2}{2})^2 + (b - R)^2 \in [(0.1R)^2, (0.2R)^2]\}$. Then, for any $(b, c) \in \mathcal{A}$, it satisfies that*

$$\left| \langle \nabla L(\tau), \frac{\tau_* - \tau}{\|\tau_* - \tau\|} \rangle \right| = O(1) \cdot \exp\left(-\frac{\kappa(b, c) \cdot R^2}{8}\right) \tag{4.3}$$

where $\kappa(b, c) \in [\frac{3}{4}, \frac{5}{4}]$ is a small constant.

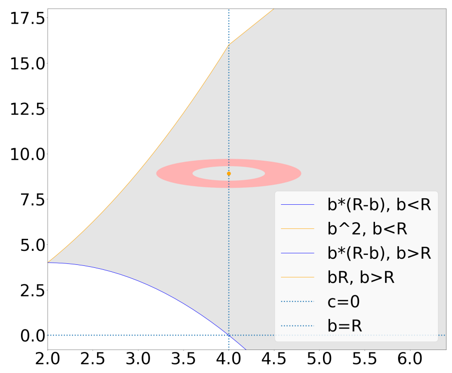


Figure 4.1: The gray-shaded area is the region where certain conditions (see equation 4.17) are satisfied. The orange dot marks τ_*, which is enclosed in the green-shaded area. Moreover, the red-shaded area centered at τ_* corresponds the width-$0.1R$ annulus $\mathcal{A}$, within which the gradient is exponentially small.

Given Lemma 49, to prove Theorem 17, we will first show that the lemma gives an upper bound for the decrease in parameter distance, that is, we show $\eta \left|\langle \nabla L(\tau), \frac{\tau_*-\tau}{\|\tau_*-\tau\|}\rangle\right| \geq \|\tau_t - \tau_*\| - \|\tau_{t+1} - \tau_*\|$. Towards this claim, we write τ_{t+1} as:

$$\tau_{t+1} = \tau_t - \eta \nabla L(\tau_t) = \tau_t - \eta \left\langle \nabla L(\tau_t), \frac{\tau_* - \tau_t}{\|\tau_* - \tau_t\|} \right\rangle \cdot \frac{\tau_* - \tau_t}{\|\tau_* - \tau_t\|} - \eta \boldsymbol{v} \tag{4.4}$$

where $\boldsymbol{v} := \nabla L(\tau_t) - \langle \nabla L(\tau_t), \frac{\tau_*-\tau_t}{\|\tau_*-\tau_t\|}\rangle \cdot \frac{\tau_*-\tau_t}{\|\tau_*-\tau_t\|}$ is orthogonal to $\tau_* - \tau_t$. Hence

$$\|\tau_{t+1} - \tau_*\| = \left(1 - \frac{\eta}{\|\tau_* - \tau_t\|} \left\langle \nabla L(\tau_t), \frac{\tau_* - \tau_t}{\|\tau_* - \tau_t\|} \right\rangle\right) \cdot \|\tau_t - \tau_*\| + \eta \|\boldsymbol{v}\|, \tag{4.5}$$

From this, we can conclude

$$\|\tau_t - \tau_*\| - \|\tau_{t+1} - \tau_*\| = \eta \left\langle \nabla L(\tau_t), \frac{\tau_* - \tau_t}{\|\tau_* - \tau_t\|} \right\rangle - \eta \|\boldsymbol{v}\| \leq \eta \left| \left\langle \nabla L(\tau_t), \frac{\tau_* - \tau_t}{\|\tau_* - \tau_t\|} \right\rangle \right|. \tag{4.6}$$

The next step is to show that there is a path lying in $\mathcal{A}$ of length at least $0.01R$ that gradient descent has to go through. We have the following lemma:

Lemma 50. *Let $\eta = o(1)$. For any τ s.t. $\|\tau - \tau_*\| \geq 0.2R$, let τ' denote the point after one step of gradient descent from τ, then $\|\tau' - \tau_*\| > 0.15R$.*

From any such τ', the shortest way to exit the annulus $\mathcal{A}$ is to project onto the inner circle defining $\mathcal{A}$, i.e. the circle centered at τ_* with radius $0.1R$ which is a convex set. Denote this inner circle as $\mathcal{B}(\tau_*, 0.1R)$ whose projection is $\Pi_{\mathcal{B}(\tau_*,0.1R)}$, then the shortest path is the line segment $\tau' - \Pi_{\mathcal{B}(\tau_*,0.1R)}(\tau')$. Further, this line segment is of length $0.05R$ since $\|\tau' - \tau_*\| > 0.15R$ by Lemma 50, while the decrease of the parameter distance (i.e. $\|\tau - \tau_*\|$) is exponentially small at any point in $\mathcal{A}$ by Lemma 49 and equation 4.6. Hence the number of steps to exit $\mathcal{A}$ is lower bounded by $\frac{0.05R}{\eta \cdot O(1) \cdot \exp\left(-\frac{\kappa R^2}{8}\right)} = \omega(R) \exp\left(\frac{\kappa R^2}{8}\right)$. □

Proofs for Lemma 49 and Lemma 50 are provided in Section 4.1.7.1.

Next, we proceed to second order methods, which are a natural guess for a remedy to the drastically changing norms of the gradients, as they can precondition the gradient. Unfortunately, standard second-order approaches are again of no help, and the number of steps required to converge remains exponential. Consider Newton's method with updates of the form $\eta(\nabla^2 L)^{-1}\nabla L$. At first glance, this looks like it may solve the issue of a flat gradient, since the Hessian $\nabla^2 L$ may also be exponentially small hence canceling out with the exponentially small gradient. However, the flatness of the landscape forces us to take an exponentially small step size η, resulting in the following claim:

Theorem 18 (Lower bound for Newton's method)**.** *Let P_*, Q, P satisfy the same conditions as in Theorem 17. Let $\lambda_\rho := \min_{\theta\in\Theta}\sigma_{\min}(\nabla^2 L(\tau_\theta))$, $\lambda_M := \max_{\theta\in\Theta}\sigma_{\max}(\nabla^2 L(\tau_\theta))$. Then, running the Newton's method with step size $\eta = O(\frac{\lambda_\rho}{\lambda_M})$ from an initialization $\tau = \tau_q$ will need an exponential number of steps to reach some τ' that is $O(1)$ close to τ_*.*

Again, the condition $\eta = O\left(\frac{\lambda_\rho}{\lambda_M}\right)$ follows the typical step size choice for Newton's method, i.e. the step size should be upper bounded by the ratio between the *global* strong convexity constant and the *global* smoothness of the function, which is exponentially small for this setup by Lemma 47, 48. The proof of Theorem 18 is deferred to Appendix 4.1.9.1.

4.1.4 Normalized gradient descent for well-conditioned losses

We have seen that due to an ill-behaved landscape, NCE optimized with standard gradient descent or Newton's method will fail to reach a good parameter estimate efficiently, even on a problem as simple as Gaussian mean estimation, and even with access to the population gradient.

In this section, we will show that a close relative of gradient descent, *normalized gradient descent* (NGD), despite its simplicity, provides a fix to the flatness problem to exponential family distributions when the Hessian of the loss is well-conditioned close to the optimum.

Precisely, recall that the NGD updates for a loss function L is $\tau_{t+1} = \tau_t - \eta\frac{\nabla L(\tau_t)}{\|\nabla L(\tau_t)\|_2}$. We assume that in a neighborhood around τ_*, the change in the shape of the Hessian $\boldsymbol{H}$ is moderate: [5]

Assumption 15 (Hessian in a neighborhood of τ_*)**.** *Under Assumption 12 with constant β_Z, assume that for any τ such that $\|\tau-\tau_*\|_2 \le \frac{1}{\beta_Z}$, it holds that $\sigma_{\max}(\boldsymbol{H}(\tau)) \le \beta_u \cdot \sigma_{\max}(\boldsymbol{H}(\tau_*))$, and $\sigma_{\min}(\boldsymbol{H}(\tau)) \ge \beta_l \cdot \sigma_{\min}(\boldsymbol{H}(\tau_*))$, for some constant $\beta_u, \beta_l > 0$.*

The main result of this section states that NGD can find a parameter estimate efficiently for exponential families, where the number of steps required is polynomial in the distance between the initial estimate and the optimum:

Theorem 19. *Let L be any loss function that is convex in the exponential family parameter and satisfies Assumptions 15 and 11 - 13. Furthermore, let P_*, Q be exponential family distributions with parameters τ_*, τ_q and let κ_* be the condition number of the Hessian at $P = P_*$. Then, for any $0 < \delta \le \frac{1}{\beta_Z}$ and parameter initialization τ_0, with step size $\eta \le \sqrt{\frac{\beta_l}{\beta_u\kappa_*}}\delta$, performing NGD on the population objective L guarantees that after $T \le \frac{\beta_u\kappa_*}{\beta_l}\cdot\frac{\|\tau_0-\tau_*\|^2}{\delta^2}$ steps, there exists an iterate $t \le T$ such that $\|\tau_t - \tau_*\|_2 \le \delta$.*

[5]As a concrete example, we will show in the next section that a variant of NCE satisfies both conditions.

The main technical ingredient for proving Theorem 19 is the following Lemma:

Lemma 51. *Suppose Assumptions 12 and 15 hold with constants β_Z, β_u and β_l. Let L be a convex function with minimizer τ_*, and let $g := \nabla L(\tau)$. For any $\delta \le \frac{1}{\beta_Z}$, let $\gamma = \sqrt{\frac{\beta_l}{\beta_u \kappa_*}}\delta$. Then for all τ s.t. $\|\tau - \tau_*\|_2 \ge \delta$, we have $L(\tau_* + \gamma\frac{g}{\|g\|}) \le L(\tau)$.*

We will first prove Theorem 19 then return to the proof of this lemma.

Proof of Theorem 19. Denote $g_t := \nabla L(\tau_t)$ and $R := \|\tau_* - \tau_q\|_2$ for notation convenience. Recall that the NGD update with step size η is $\tau_{t+1} = \tau_t - \eta \cdot \frac{g_t}{\|g_t\|_2}$. Then, $\|\tau_t - \tau_*\|^2$ can be rewritten as:

$$\|\tau_{t+1} - \tau_*\|^2 = \|\tau_t - \tau_*\|^2 - 2\gamma\eta + \eta^2 + 2\eta \frac{g_t^\top}{\|g_t\|}\left(\tau_* + \gamma\frac{g_t}{\|g_t\|} - \tau_t\right) \tag{4.7}$$

If we set γ s.t. the last term is smaller than 0 for all τ that are not within distance δ to τ_*, setting $\eta = \gamma$ gives:

$$\|\tau_{t+1} - \tau_*\|^2 \le \|\tau_t - \tau_*\|^2 - 2\gamma\eta + \eta^2 = \|\tau_t - \tau_*\|^2 - \gamma^2 \tag{4.8}$$

Hence the number of steps required to find a τ s.t. $\|\tau - \tau_*\|_2 \le \delta$ is at most $T \le \frac{\|\tau_0 - \tau_*\|^2}{\gamma^2}$.

By Lemma 51, setting $\gamma = \sqrt{\frac{\beta_l}{\beta_u \kappa_*}}\delta$ ensures $L(\tau_* + \gamma\frac{g_t}{\|g_t\|}) \le L(\tau_t)$ for any τ_t that is at least δ away from τ_*. It then follows from the convexity of L that

$$g_t^\top\left(\tau_* + \gamma\frac{g_t}{\|g_t\|} - \tau_t\right) \le L\left(\tau_* + \gamma\frac{g_t}{\|g_t\|}\right) - L(\tau_t) \le 0. \tag{4.9}$$

Substituting this choice of γ back to the bound for T gives $T \le \frac{\beta_u \kappa_*}{\beta_l} \cdot \frac{\|\tau_0 - \tau_*\|^2}{\delta^2}$ □

Finally, we return to proving Lemma 51:

Proof of Lemma 51. The proof follows from the Taylor expansion around τ_*: for any unit vector $\boldsymbol{v}$ and any constant $c \le \gamma$, the Taylor remainder theorem states that there exists some constant $c' < c$ and unit vector $\boldsymbol{v}'$ such that $L(\tau_* + c\boldsymbol{v}) - L(\tau_*) = \frac{c^2}{2}\boldsymbol{v}^\top \boldsymbol{H}(\tau_* + c'\boldsymbol{v}')\boldsymbol{v}$.

For any unit vector $\boldsymbol{v}_1, \boldsymbol{v}_2$ and constants $c_1, c_2 \le \delta$ such that $L(\tau_* + c_1\boldsymbol{v}_1) = L(\tau_* + c_2\boldsymbol{v}_2)$, we have

$$\begin{aligned} &L(\tau_* + c_1\boldsymbol{v}_1) - L(\tau_*) = \frac{c_1^2}{2}\boldsymbol{v}_1^\top \boldsymbol{H}(\tau_* + c_1'\boldsymbol{v}_1')\boldsymbol{v}_1 = \frac{c_2^2}{2}\boldsymbol{v}_2^\top \boldsymbol{H}(\tau_* + c_2'\boldsymbol{v}_2')\boldsymbol{v}_2 = L(\tau_* + c_2\boldsymbol{v}_2) - L(\tau_*) \\ &\Rightarrow \frac{c_1}{c_2} \le \sqrt{\frac{\sigma_{\max}(\boldsymbol{H}(\tau_* + c_1'\boldsymbol{v}_1'))}{\sigma_{\min}(\boldsymbol{H}(\tau_* + c_2'\boldsymbol{v}_2'))}} \le \sqrt{\frac{\beta_u}{\beta_l}\kappa_*} \end{aligned} \tag{4.10}$$

This means for any two points with the same loss, the ratio between their distances to τ_* will be at most $\sqrt{\frac{\beta_u}{\beta_l}\kappa_*}$. Therefore setting $\gamma = \sqrt{\frac{\beta_l}{\beta_u \kappa_*}}\delta$ guarantees that for any τ that is at least δ away from τ_*, τ will have a larger loss than any point that is γ away from τ_*. In other words, $L(\tau_1) \le L(\tau_2)$ holds for any $\tau_1 \in \mathcal{B}(\tau_*, \gamma)$, $\tau_2 \notin \mathcal{B}(\tau_*, \delta)$. □

4.1.4.1 Example: 1d Gaussian mean estimation

It is relatively straightforward to check that NGD addresses the flatness problem faced by Gaussian mean estimation we considered in Section 4.1.3:

Corollary 2. *Let P_*, Q be 1d Gaussian with covariance 1 and mean $\theta_* = R$ where $R \ll 1$, and $\theta_q = 0$. For any given $\delta \leq \frac{1}{R}$ and initial estimate $\tau_0 = \tau_q$, NGD can find an estimate τ such that $\|\tau - \tau_*\|_2 \leq \delta$, with at most $O(\frac{R^6}{\delta^2})$ steps.*

Intuitively, the effectiveness of NGD comes from the crucial observation that though the magnitude for the loss and derivatives can be exponentially small, they share the same exponential factor, making normalization effective. Formally, it can be shown that $\frac{\beta_u}{\beta_l} = O(1)$ (Appendix 4.1.9.1). Corollary 2 then follows from Theorem 19 and the curvature and strong convexity from Lemma 46, 47.

4.1.4.2 Bounds on the condition number of NCE

The convergence rate in Theorem 19 depends on κ_*, the condition number of the NCE Hessian at the optimum, and Hessian-related constants β_u, β_l in Assumption 15. We now show that under the setup of Theorem 19, κ_* and β_u, β_l can be related to the *Bhattacharyya coefficient* between P_* and Q, which is a similarity measure defined as $\text{BC}(P_*, Q) := \int_x \sqrt{p_*(x)q(x)}dx$. As a result, we get the following convergence guarantee:

Theorem 20. *Suppose Assumptions 11- 14 hold with constants ω, β_Z, $\lambda_{\max}$ and $\lambda_{\min}$, $\gamma_{\max}$ and $\gamma_{\min}$. Consider a NCE task with data distribution P_1 and noise distribution P_2, parameterized by $\theta_1, \theta_2 \in \Theta$ respectively. Define constant $C := 18 \exp\left(\frac{2}{\beta_Z}\right) \cdot \left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)^3 \cdot \min\left\{\frac{2\lambda_{\max}^2}{\lambda_{\min}^2}, \frac{2\lambda_{\min}+\gamma_{\max}\|\delta\|}{\lambda_{\min}-\gamma_{\min}\|\delta\|}\right\}$. Then, for any $0 < \delta \leq \frac{1}{\beta_Z}$ and parameter initialization τ_0, with step size $\eta \leq \sqrt{\frac{\beta_l}{\beta_u \kappa_*}}\delta$, performing NGD on the population objective L guarantees that after $T \leq C \cdot \frac{1}{\text{BC}(P_*,Q)^3} \frac{\|\tau_0 - \tau_*\|^2}{\delta^2}$ steps, there exists an iterate $t \leq T$ such that $\|\tau_t - \tau_*\|_2 \leq \delta$.*

In particular, when P_*, Q are not too far, we can further show a lower bound on $\text{BC}(P_*, Q)$:

Lemma 52. *For P_1, P_2 parameterized by $\theta_1, \theta_2 \in \Theta$, if $\|\theta_1 - \theta_2\|_2^2 \leq \frac{4}{\lambda_{\max}}$, then $BC(P_1, P_2) \geq \frac{1}{2}$.*

The proofs of Theorem 20 and Lemma 52 rely on analyzing the geodesic on the manifold of square root densities $\sqrt{p}$ equipped with the Hellinger distance as a metric; the details are deferred to Section 4.1.10. It is also worth noting that Theorem 20 only requires $\|\theta_1 - \theta_2\|$ to be smaller than a constant, rather than tending to zero as usually required for analyses using Taylor expansions.

Finally, we would like to note that although our analysis can be tightened, it is unlikely to remove such dependency since NGD only uses first-order information. [6] Moreover, the condition number κ_* also affects the practical use of Newton-like methods, since matrix inversion is widely known to be sensitive to numerical issues when the matrix is extremely ill-conditioned. It is an interesting open question whether a non-standard preconditioning approach might be amenable to this setting.

[6]In the next section, we will that the condition number is provably polynomial in $\|\theta_* - \theta_q\|$ for a variant of the NCE loss.

4.1.5 Analyzing eNCE : NCE with an exponential loss

The previous section proved that NGD can serve as a simple fix to overcome the flatness problem of NCE for well-conditioned losses. However, though we showed κ_* has a polynomial growth when the distributions P, Q^* are sufficiently close —it is unclear how κ_* behaves beyond this threshold.

In this section, we introduce a slight modification to the NCE objective, which we call the *eNCE* objective, for which κ_* depends *polynomially* on all the exponential family-related constants. This means though eNCE may still suffer from the flatness problem, *eNCE and NGD together provide a solution that guarantees a polynomial convergence rate.*

Towards formalizing this, the eNCE loss is defined as:

Definition 20 (eNCE Loss)**.** *Let* $\varphi(x) := \log\sqrt{\frac{p(x)}{q(x)}}$*, and* $l(x,y) := \exp(-y\varphi(x))$ *for* $y \in \{\pm 1\}$*. The eNCE loss of* P_θ *w.r.t. data distribution* P_* *and noise* Q *is:*

$$L_{\exp}(P_\theta) = \frac{1}{2}\mathbb{E}_{x\sim P_*}[l(x,1)] + \frac{1}{2}\mathbb{E}_{x\sim Q}[l(x,-1)] = \frac{1}{2}\int_x p_*\sqrt{\frac{q(x)}{p(x)}} + \frac{1}{2}\int_x q\sqrt{\frac{p(x)}{q(x)}} \tag{4.11}$$

It can be checked easily that the minimizing φ is $\varphi(x) = \frac{1}{2}\log\frac{p_*}{q}$. Moreover, each φ is associated with an induced distribution p, defined by $p(x) = \exp(2\varphi(x))q(x)$.

Relation to NCE: Same as the original NCE loss (referred to as "NCE" below), eNCE learns to solve a distinguishing task between samples from P_* or Q. The difference lies only in the losses, which have analogous forms: the NCE loss described in Definition 19 can be rewritten in the same form with $l(x,y) := \log\frac{1}{1+\exp(-y\psi(x))}$ and $\psi(x) := \log\frac{p(x)}{q(x)}$.

The main advantage of the exponential loss is that the Hessian at the optimum is now guaranteed to be well-conditioned. Namely, the crucial technical lemma is the following result:

Lemma 53 (Polynomial condition number for eNCE loss)**.** *Under Assumption 13 with constants* $\lambda_{\max}, \lambda_{\min}$*, the condition number of the eNCE Hessian at the optimum is bounded by* $\kappa_* \le \frac{\lambda_{\max}}{\lambda_{\min}}$*.*

We can also show that eNCE satisfies part (ii) of Assumption 15, whose proof is deferred to Appendix 4.1.11.

Lemma 54. *Under Assumption 12, 13 with constant* β_Z*,* $\lambda_{\max}$ *and* $\lambda_{\min}$*, for any unit vector* $\boldsymbol{u}$ *and constant* $c \in [0, \frac{1}{\beta_Z}]$*, the maximum and minimum singular values of* $\boldsymbol{H}(\tau_* + c\boldsymbol{u})$ *satisfy Assumption 15 with constants* $\beta_u = 2e \cdot \frac{\lambda_{\max}}{\lambda_{\min}}$*,* $\beta_l = \frac{1}{2e}\cdot\frac{\lambda_{\min}}{\lambda_{\max}}$*.*

Lemma 53 and Lemma 54 together imply the Hessian is well-conditioned around the optimum. Combined with Theorem 19, we have the main result of this section:

Theorem 21. *Let* P_*, Q *be exponential family distributions with parameters* τ_*, τ_q *under Assumption 11-13. Let* β_Z *be the constant for Assumption 12, and let* $\lambda_{\max}, \lambda_{\min}$ *be constants for Assumption 13. For any given* $\delta \le \frac{1}{\beta_Z}$ *and parameter initialization* τ_0*, performing NGD on the eNCE objective guarantees that when taking* $T \le 4e^2 \cdot \frac{\lambda_{\max}^3}{\lambda_{\min}^3} \cdot \frac{\|\tau_0 - \tau_*\|^2}{\delta^2}$ *steps, there exists an iterate* $t \le T$ *such that* $\|\tau_t - \tau_*\|_2 \le \delta$*.*

Proof. Theorem 21 follows directly from Theorem 19, using the condition number bound from Lemma 53 and constants from 54. □

We now return to proving Lemma 53:

Proof of Lemma 53. Let's first write out the Hessian for the eNCE objective:

$$\begin{aligned}
L_{\exp}(P) =& \frac{1}{2}\int_x p_*\sqrt{\frac{q}{p}} + \frac{1}{2}\int_x q\sqrt{\frac{p}{q}} \\
\nabla L_{\exp}(P) =& \frac{1}{4}\int_x \sqrt{q}\left(\sqrt{p} - \frac{p_*}{\sqrt{p}}\right)\nabla \log p \\
\nabla^2 L_{\exp}(P) =& \frac{1}{4}\int_x \sqrt{q}\left(\sqrt{p} - \frac{p_*}{\sqrt{p}}\right)\cdot \nabla^2 \log p + \frac{1}{8}\int_x \sqrt{q}\left(\sqrt{p} + \frac{p_*}{\sqrt{p}}\right)\nabla \log p (\nabla \log p)^\top \\
=& \frac{1}{8}\int_x \sqrt{q}\left(\sqrt{p} + \frac{p_*}{\sqrt{p}}\right)\nabla \log p (\nabla \log p)^\top \\
=& \frac{1}{8}\int_x p_*\sqrt{\frac{q}{p}}T(x)T(x)^\top + \frac{1}{8}\int_x q\sqrt{\frac{p}{q}}T(x)T(x)^\top
\end{aligned} \tag{4.12}$$

Note that this Hessian is always PSD, which means $L_{\exp}$ is convex in the parameters of exponential families.

Recall that $\theta_*, \theta_q, \tilde{T}$ denote the parameters and sufficient statistics without the partition function coordinate, and τ_*, τ_q, T denote the extended version with the partition function, e.g. $\tau_* = [\theta_*, \log Z(\theta_*)]$, $T(x) = [\tilde{T}(x), -1]$. Then, we can rewrite $\boldsymbol{H}_*$ as:

$$\begin{aligned}
\boldsymbol{H}_* =& \frac{1}{4}\int_x \sqrt{p_* q}T(x)T(x)^\top = \frac{1}{4}\int_x \exp\left(\frac{(\tau_* + \tau_q)^\top}{2}T(x)\right)T(x)T(x)^\top \\
=& \frac{1}{4}\int_x \exp\left(\frac{(\theta_* + \theta_q)^\top}{2}\tilde{T}(x) - \frac{1}{2}\log Z(\theta_*) - \frac{1}{2}\log Z(\theta_q)\right)T(x)T(x)^\top \\
=& \frac{1}{4}\underbrace{\frac{Z\left(\frac{\theta_*+\theta_q}{2}\right)}{\sqrt{Z(\theta_*)Z(\theta_q)}}}_{B(P_*,Q)}\int_x \frac{\exp\left(\left(\frac{\theta_*+\theta_q}{2}\right)^\top \tilde{T}(x)\right)}{Z\left(\frac{\theta_*+\theta_q}{2}\right)}T(x)T(x)^\top dx = \frac{B(P_*,Q)}{4}\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[TT^\top]
\end{aligned} \tag{4.13}$$

Since $\frac{\theta_*+\theta_q}{2} \in \Theta$, we have $\lambda_{\min}\boldsymbol{I} \preceq \mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[TT^\top] \preceq \lambda_{\max}\boldsymbol{I}$ by Assumption 13. The Lemma hence follows. □

4.1.6 Empirical verification

To corroborate our theory, we verify the effectiveness of NGD and eNCE on Gaussian mean estimation and the MNIST dataset. For MNIST, we use a ResNet-18 to model the log density ratio $\log(p/q)$, following the setup in TRE [Rhodes et al., 2020].

Results: For Gaussian data, we run gradient descent (GD) and normalized gradient descent (NGD) on the NCE loss and eNCE loss. Figure 4.2 compares the best runs under each setup given a fixed

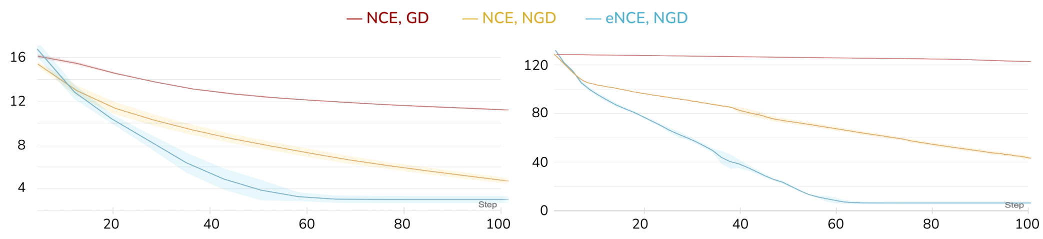


Figure 4.2: Results for estimating 1d (left) and 16d (right) Gaussians, plotting the best parameter distance $\|\tau_* - \tau\|_2$ (y-axis) against the number of updates (x-axis). In both cases, when using NCE, normalized gradient descent ("NCE, NGD", yellow curve) largely outperforms gradient descent ("NCE, GD", red curve). When using NGD, the proposed eNCE ("eNCE, NGD", blue curve) decays faster than the original NCE loss. The results are averaged over 5 runs, with shaded areas showing the standard deviation.

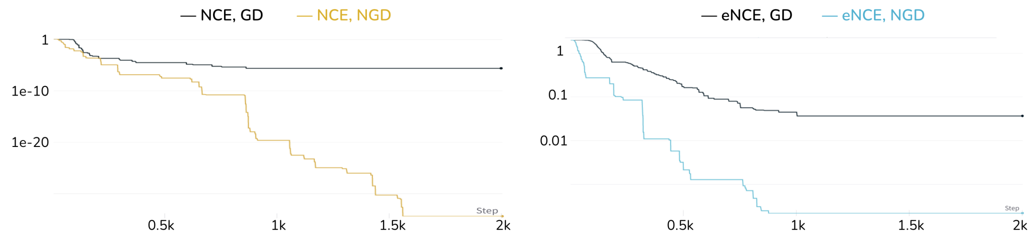


Figure 4.3: Results on MNIST, plotting loss value (y-axis, log scale) against update steps (x-axis). The left plot shows NCE optimized by GD (black) and NGD (yellow), and the right shows eNCE optimized by GD (black) and NGD (blue). It can be seen that NGD outperforms GD in both cases.

computation budget (100 update steps), where "best" is defined to be the run with the lowest loss on fresh samples. The plots show the minimum parameter distance $\|\tau_* - \tau\|_2$ up to each step. We find that NGD indeed outperforms GD, and that the proposed eNCE sees a further improvement over NCE while additionally enjoying provable polynomial convergence guarantees.

For MNIST, we can no longer compare parameter distances since τ_* is unknown. Instead, we compare the result of optimization directly in terms of loss achieved, again under a fixed computation budget (2K steps). The results are shown in Figure 4.3, with NGD converging significantly faster for both NCE and eNCE.

Implementation details We note that eNCE can be numerically unstable, especially when P_*, Q are well separated. Below we introduce implementation details for preventing numerical issues.

Parameterization: For the 1-dimensional Gaussian, we take P_*, Q to have mean $\mu_* = 16$, $\mu_q = 0$, and unit variance $\sigma_*^2 = \sigma_q^2 = 1$. We use $h(x) := \exp(-\frac{x^2}{2})$, $T(x) := [x, -1]$ to be consistent with the notation in Section 4.1.3. For the 16-dimensional Gaussian, P_*, Q share the same mean $\mu_* = \mu_q = 0$ but have different covariance with $\text{Cov}_q = \boldsymbol{I}_d$ and $\text{Cov}_p = \text{diag}([s_1, ..., s_d])$, where $s_i = \text{Uniform}[8 \times 0.75, 8 \times 1.5]$. [7]

[7]Generally, for d-dimensional Gaussian with mean μ and a diagonal covariance matrix $\Sigma := \text{diag}([\sigma_1^2, ..., \sigma_d^2])$, the exponential parametrization is $\tau = [\frac{1}{\sigma_1^2}, ..., \frac{1}{\sigma_d^2}, \frac{\mu_1}{\sigma_1^2}, ... \frac{\mu_d}{\sigma_d^2}, \frac{\mu^\top \Sigma^{-1} \mu}{2} + \frac{1}{2}\log((2\pi)^d \det(\Sigma))]$.

For MNIST, we adapt the TRE implementation by Rhodes et al. [2020]. We model the log density ratio $\log(p/q)$ by a quadratic of the form $g(x) := -f(x)^\top \boldsymbol{W} f(x) - \boldsymbol{b}^\top f(x) - c$, where f is ResNet-18, and $\boldsymbol{W}, \boldsymbol{b}, c$ are trainable parameters with $\boldsymbol{W}$ constrained to be positive definite.

In addition, we find the following tricks helpful in improving numerical stability:

- *Calculation in log space*: instead of dividing two pdfs, we found it more numerically stable to use subtraction between the log pdfs and then exponentiate.
- *Removing common additive factors*: the empirical loss is the average loss over a batch of samples where overflow can happen. [8] We found it more stable to calculate the mean by first subtract the largest value of the batch, calculate the mean of the remaining values, then add back the large value—akin to the usual log-sum-exp trick. For example, $\text{mean}([a,b]) = \max(a,b) + \text{mean}([a - \max(a,b), b - \max(a,b)])$.
- *Per-sample gradient clipping*: it is sometimes helpful to limit the amount of gradient contributed by any data point in a batch. We ensure this by limiting the norm of the gradient, that is, the gradient from a sample x is now $\min\{1, \frac{K}{\|\nabla \ell(x)\|}\}\nabla \ell(x)$ for some prespecified constant K [Tsai et al., 2021].
- *Per-sample log ratio clipping*: an alternative to per-sample gradient clipping is to upper threshold the absolute value of the log density ratio on each sample, before passing it to the loss function. Setting a proper threshold prevents the loss from growing too large, and consequently prevents a large gradient update.

4.1.7 Proofs: Flatness of NCE (Section 4.1.3)

This section provides proofs for the negative results in Section 4.1.3, that is, the NCE landscape is ill-behaved with exponentially flat loss, gradient, and curvature. We start with proving the helper lemmas (Lemma 49, 50, used for the proof of Theorem 17) in Section 4.1.7.1. Results related to second-order quantities are proved in Section 4.1.8 (Lemma 46, Lemma 47), Section 4.1.9 (Lemma 48), and Section 4.1.9.1 (Theorem 18).

As a note on the notation, in the following, we will use $a \lesssim b$ to denote $a = O(b)$ with O hiding a constant less than 2. Similarly, $a \gtrsim b$ denotes $a = \Omega(b)$ where Ω hides a constant greater than $\frac{1}{2}$.

4.1.7.1 Proof of Lemma 49, 50: bounding progress from 1 gradient descent step

4.1.7.1.1 Proof of Lemma 49

Lemma (Lemma 49, restated)**.** *Consider the annulus* $\mathcal{A} := \{(b,c) : (c - \frac{R^2}{2})^2 + (b-R)^2 \in [(0.1R)^2, (0.2R)^2]\}$. *Then, for any* $(b,c) \in \mathcal{A}$, *it satisfies that*

$$\left|\langle \nabla L(\tau), \frac{\tau_* - \tau}{\|\tau_* - \tau\|}\rangle\right| = O(1) \cdot \exp\left(-\frac{\kappa(b,c)\cdot R^2}{8}\right), \tag{4.14}$$

where $\kappa(b,c) \in [\frac{3}{4}, \frac{5}{4}]$ *is a small constant.*

[8] This is because the mean function is internally implemented as the sum of all entries divided by the batch size, and the sum of a large batch size where each value is also large can lead to overflow.

Proof. Recall that for 1d Gaussian with a known unit covariance, we can use parameter $\tau := [b, c]$ and sufficient statistics $T(x) := [x, -1]$, with pdf $p(x) = \exp\left(-\frac{x^2}{2}\right) \cdot \exp\left(\langle \tau, T(x)\rangle\right)$.

For any τ such that $\|\tau_* - \tau\| \geq 1$, $\left|\langle \nabla L(\tau), \frac{\tau_* - \tau}{\|\tau_* - \tau\|}\rangle\right|$ can be upper bounded as:

$$\begin{aligned}
&2\left|\langle \nabla L(\tau), \frac{\tau_* - \tau}{\|\tau_* - \tau\|}\rangle\right| \leq 2\left|\langle \nabla L(\tau), \tau_* - \tau\rangle\right| = \left|\int_x \frac{p - p_*}{\frac{p}{q} + 1}\langle T(x), \tau_* - \tau\rangle\right| \\
=&\left|\int_x \frac{p - p_*}{\frac{p}{q} + 1}\left[(R - b)x - \frac{R^2}{2} - \log\sqrt{2\pi} + c\right]\right| \\
\leq&(R - b)\left|\int_x \frac{p - p_*}{\frac{p}{q} + 1} x\right| + \left|\frac{R^2}{2} + \log\sqrt{2\pi} - c\right| \cdot \left|\int_x \frac{p - p_*}{\frac{p}{q} + 1}\right|.
\end{aligned} \tag{4.15}$$

Let $a \simeq b$ denote $a = kb$ for a constant $k = \Theta(1)$. We first show the calculations with $b > 0$ for cleaner presentation; the $b < 0$ case is analogous and deferred to the end of Section 4.1.7.1.2.

Bounding $\left|\int_x \frac{p - p_*}{\frac{p}{q} + 1}\right|$:

$$\begin{aligned}
&\left|\int_x \frac{p - p_*}{\frac{p}{q} + 1}\right| = \left|\int_x \frac{\exp\left(-\frac{x^2}{2} + bx - c\right) - \exp\left(-\frac{(x-R)^2}{2} - \log\sqrt{2\pi}\right)}{\exp\left(bx - c + \log\sqrt{2\pi}\right) + 1}\right| \\
\leq&\underbrace{\int_{x < \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + bx - c\right)}_{T_1^{(0)}} + \underbrace{\int_{x \geq \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} - \log\sqrt{2\pi}\right)}_{T_2^{(0)}} \\
&+ \underbrace{\int_{x < \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x - R)^2}{2} - \log\sqrt{2\pi}\right)}_{T_3^{(0)}} + \underbrace{\int_{x \geq \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + (R - b)x + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)}_{T_4^{(0)}} \\
\overset{(i)}{\simeq}&\frac{1}{\sqrt{2\pi}} \frac{1}{b - \frac{c - \log\sqrt{2\pi}}{b}} \cdot \exp\left(-\frac{(c - \log\sqrt{2\pi})^2}{2b^2}\right) + \frac{1}{\sqrt{2\pi}} \frac{1}{\frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{(c - \log\sqrt{2\pi})^2}{2b^2}\right) \\
&+ \frac{1}{\sqrt{2\pi}} \frac{1}{R - \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{(\frac{c - \log\sqrt{2\pi}}{b} - R)^2}{2}\right) + \frac{1}{\sqrt{2\pi}} \frac{1}{\frac{c - \log\sqrt{2\pi}}{b} - (R - b)} \exp\left(-\frac{\left(\frac{c - \log\sqrt{2\pi}}{b} - R\right)^2}{2}\right) \\
\simeq&\frac{b}{b^2 - c} \cdot \exp\left(-\frac{c^2}{2b^2}\right) + \frac{b}{c}\exp\left(-\frac{c^2}{2b^2}\right) + \frac{b}{bR - c}\exp\left(-\frac{(c - bR)^2}{2b^2}\right) + \frac{b}{c - b(R - b)}\exp\left(-\frac{(c - bR)^2}{2b^2}\right) \\
\overset{(ii)}{=}&O(R^{-1}) \cdot \exp\left(-\frac{\kappa(b, c) \cdot R^2}{8}\right),
\end{aligned} \tag{4.16}$$

where $\kappa(b, c) \in [\frac{3}{4}, \frac{5}{4}]$. Step (i) uses calculations in equation 4.27-4.30 (deferred to subsection 4.1.7.1.2 for cleaner presentation), and assumes (b, c) belongs to the set $\mathcal{V} := \{(b, c) : c \in [b(R - b), b \cdot \min\{b, R\}]\}$. In particular, the annulus $\mathcal{A} := \{(b, c) : (c - \frac{R^2}{2})^2 + (b - R)^2 \in [(0.1R)^2, (0.2R)^2]\}$ is a subset of $\mathcal{V}$ when $R \gg 1$. Step (ii) considers $(b, c) \in \mathcal{A}$.

We can choose b, c s.t. $b \geq \frac{R}{2}$, $c \in [b(R - b), b \cdot \min\{b, R\}]$, so that we pick up the tails in $T_1^{(0)}$ to $T_4^{(0)}$.

This means:

$$\begin{cases} c \in \left[b(R-b), b^2\right], & b \in [\frac{R}{2}, R] \\ c \in [-b(b-R), bR], & b \in [R, \infty] \end{cases} \tag{4.17}$$

Bounding $\left|\int_x \frac{p-p_*}{\frac{p}{q}+1} x\right|$*:* Using similar calculations as before, we have that when $c - \log\sqrt{2\pi} > 0$ (which is the case for $\tau = [b, c] \in \mathcal{A}$),

$$\begin{aligned} &\left|\int_x \frac{p-p_*}{\frac{p}{q}+1} x\right| \le \left|\int_x \frac{p}{\frac{p}{q}+1} x\right| + \left|\int_x \frac{p_*}{\frac{p}{q}+1} x\right| \\ \le &\max\left\{\int_{x>0} \frac{p}{\frac{p}{q}+1} x,\ -\int_{x<0} \frac{p}{\frac{p}{q}+1} x\right\} + \max\left\{\int_{x>0} \frac{p_*}{\frac{p}{q}+1} x,\ -\int_{x<0} \frac{p_*}{\frac{p}{q}+1} x\right\} \end{aligned} \tag{4.18}$$

Below we bound the case where $x > 0$; the other case (i.e. $x < 0$) has an upper bound of the same order following similar calculations and is hence omitted.

$$\begin{aligned} &\int_{x>0} \frac{p}{\frac{p}{q}+1} x + \int_{x>0} \frac{p_*}{\frac{p}{q}+1} x \\ \le &\underbrace{\int_{x\in[0, \frac{c-\log\sqrt{2\pi}}{b}]} \exp\left(-\frac{x^2}{2} + bx - c\right) x}_{T_1^{(1)}} + \underbrace{\int_{x \ge \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} - \log\sqrt{2\pi}\right) x}_{T_2^{(1)}} \\ &+ \underbrace{\int_{x\in[0, \frac{c-\log\sqrt{2\pi}}{b}]} \exp\left(-\frac{(x-R)^2}{2} - \log\sqrt{2\pi}\right) x}_{T_3^{(1)}} \\ &+ \underbrace{\int_{x \ge \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + (R-b)x + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right) x}_{T_4^{(1)}} \\ \overset{(i)}{\simeq} &\exp(-c) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right) + bT_1^{(0)} + \frac{1}{\sqrt{2\pi}} \frac{1}{\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right) \\ &+ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{R^2}{2}\right) - \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(c-\log\sqrt{2\pi}-bR)^2}{2b^2}\right) + RT_3^{(0)} \\ &+ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(c-\log\sqrt{2\pi}-bR)^2}{2b^2}\right) + (R-b)T_4^{(0)}, \end{aligned} \tag{4.19}$$

where step (i) uses calculations in equation 4.31-4.34. Ignoring small constants $\log\sqrt{2\pi}$ in $c - \log\sqrt{2\pi}$, and denoting $E_1 := \exp\left(-\frac{c^2}{2b^2}\right)$, $E_2 := \exp\left(-\frac{(c-bR)^2}{2b^2}\right)$ for notation convenience, we can substitute equation 4.16 and 4.18 into equation 4.15 as:

$$\begin{aligned}
&(R-b)\left|\int_x \frac{p-p_*}{\frac{p}{q}+1}x\right| + \left|\frac{R^2}{2}+\log\sqrt{2}-c\right|\cdot\left|\int_x \frac{p-p_*}{\frac{p}{q}+1}\right| \\
\le&(R-b)\cdot(T_1^{(1)}+T_2^{(1)}+T_3^{(1)}+T_4^{(1)}) + \left|\frac{R^2}{2}+\log\sqrt{2}-c\right|\cdot(T_1^{(0)}+T_2^{()}+T_3^{(0)}+T_4^{(0)}) \\
=&O(R)\left[\exp(-c)-E_1+bT_1^{(0)}+E_1+\exp\left(-\frac{R^2}{2}\right)-E_2+RT_3^{(0)}+E_2+(R-b)T_4^{(0)}\right] \\
&\quad+\Theta(R^2)\cdot(T_1^{(0)}+T_2^{(0)}+T_3^{(0)}+T_4^{(0)}) \\
=&O(R)\left[\exp(-c)+\exp\left(-\frac{R^2}{2}\right)\right]+\Theta(R^2)\cdot O(R^{-1})\exp\left(-\frac{\kappa(b,c)R^2}{8}\right) \\
=&O(R)\exp\left(-\frac{\kappa(b,c)\cdot R^2}{8}\right),
\end{aligned} \tag{4.20}$$

where $\kappa(b,c)\in[\frac{3}{4},\frac{5}{4}]$ is the constant defined in equation 4.16.

Since $\tau\in\mathcal{R}$, $\|\tau_*-\tau\|=\Theta(R)$, and the proof is completed by:

$$\left|\langle\nabla L(\tau),\frac{\tau_*-\tau}{\|\tau_*-\tau\|}\rangle\right| = \frac{O(R)\exp\left(-\frac{\kappa(b,c)\cdot R^2}{8}\right)}{\Theta(R)} = O(1)\exp\left(-\frac{\kappa(b,c)\cdot R^2}{8}\right). \tag{4.21}$$

□

4.1.7.1.2 Proof of Lemma 50

We first show the following claim before proving Lemma 50:

Claim 6. *For any $\tau=[b,c]\in\mathbb{R}^2$, the gradient norm at τ is $\|\nabla L(\tau)\|_2\le 32\max\{R,|b|\}$.*

Proof. For parameter $\tau=[b,c]$ where $b>0$, $c-\log\sqrt{2\pi}>0$,

$$\begin{aligned}
&\|\nabla L(\tau)\|_2\le\|\nabla L(\tau)\|_1=\left|\int_x\frac{p-p_*}{\frac{p}{q}+1}x\right|+\left|\int_x\frac{p-p_*}{\frac{p}{q}+1}\right| \\
\overset{(i)}{\le}&\exp(-c)-\exp\left(-\frac{c^2}{2b^2}\right)+bT_1^{(0)}+\exp\left(-\frac{c^2}{2b^2}\right)+\exp\left(-\frac{R^2}{2}\right)-\exp\left(-\frac{(c-bR)^2}{2b^2}\right) \\
&+RT_3^{(0)}+\exp\left(-\frac{(c-bR)^2}{2b^2}\right)+(R-b)T_4^{(0)}+T_1^{(0)}+T_2^{(0)}+T_3^{(0)}+T_4^{(0)} \\
\overset{(ii)}{\simeq}&(b+1)T_1^{(0)}+T_2^{(0)}+(R+1)T_3^{(0)}+(R-b+1)T_4^{(0)} \\
\le&4+b+R+\max\{R-b,0\}\lesssim 2\max\{R,b\},
\end{aligned} \tag{4.22}$$

where step (i) and (ii) use equation 4.31-4.34 and equation 4.27-4.30. Moreover, step (ii) increases the value by at most 16. Hence overall we have $\|\nabla_\tau L\|_2\le 32\max\{R,b\}$.

When $b>0$, $c-\log\sqrt{2\pi}<0$:

$$\left|\int_x\frac{p-p_*}{\frac{p}{q}+1}x\right|\le\left|\int_x\frac{p}{\frac{p}{q}+1}x\right|+\left|\int_x\frac{p_*}{\frac{p}{q}+1}x\right|\le\max\left\{\int_{x>0}\frac{p}{\frac{p}{q}+1}x+\int_{x>0}\frac{p_*}{\frac{p}{q}+1}x,\ -\int_{x<0}\frac{p}{\frac{p}{q}+1}x-\int_{x<0}\frac{p_*}{\frac{p}{q}+1}x\right\}. \tag{4.23}$$

Let's bound the first term (i.e. $x > 0$); the bound for the second term (i.e. $x < 0$) follows from similar calculations and is on the same order.

$$\begin{aligned}
&\int_{x>0} \frac{p}{\frac{p}{q}+1}x + \int_{x>0} \frac{p_*}{\frac{p}{q}+1}x \leq \int_{x>0} qx + \int_{x>0} \frac{p_* q}{p} x \\
=&\frac{1}{\sqrt{2\pi}}\int_{x>0}\exp\left(-\frac{x^2}{2}\right)x + \frac{1}{\sqrt{2\pi}}\int_{x>0}\exp\left(-\frac{x^2}{2}+(R-b)x+c-\frac{R^2}{2}-\log\sqrt{2\pi}\right)x \\
\overset{(i)}{\underline{\underline{\ }}}&\begin{cases} 1+(R-b)\exp\left(\frac{b^2}{2}-Rb+c-\log\sqrt{2\pi}\right)+\frac{1}{(b-R)^2+1}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right), & R-b>0 \\ 1+\frac{1}{(b-R)^2+1}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right), & R-b<0 \end{cases} \\
=&O(1).
\end{aligned} \tag{4.24}$$

Step (i) omits a factor of $\frac{1}{\sqrt{2\pi}}$ and uses:

$$\begin{aligned}
&\int_{x>0}\exp\left(-\frac{x^2}{2}+(R-b)x+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)x \\
=&\exp\left(\frac{(R-b)^2}{2}-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)\int_{x>0}\exp\left(-\frac{(x-(R-b))^2}{2}\right)x \\
=&\exp\left(\frac{(R-b)^2}{2}-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)\left[\int_{x>-(R-b)}\exp\left(-\frac{x^2}{2}\right)x+(R-b)\int_{x>-(R-b)}\exp\left(-\frac{x^2}{2}\right)\right] \\
\simeq&\begin{cases} (R-b)\exp\left(\frac{b^2}{2}-Rb+c-\log\sqrt{2\pi}\right)+\frac{1}{(b-R)^2+1}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right) & R-b>0 \\ \frac{1}{(b-R)^2+1}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right), & R-b<0 \end{cases}
\end{aligned} \tag{4.25}$$

For $b < 0$, we similarly have $\|\nabla L(\tau)\|_2 = O(\max\{R, -b\})$. The calculations are similar to the $b > 0$ case and hence omitted.

□

We are now ready to prove Lemma 50, which we restate below.

Lemma 55 (Lemma 50, restated)**.** *Let $\eta = o(1)$. For any τ s.t. $\|\tau - \tau_*\| \geq 0.2R$, let τ' denote the point after one step of gradient descent from τ, then $\|\tau' - \tau_*\| > 0.15R$.*

Proof of Lemma 55. We will prove by contradiction. First assume that we can go from τ where $\|\tau - \tau_*\|_2 \geq 0.2R$ to some τ' where $\|\tau' - \tau_*\|_2 \leq 0.15R$. Then $\|\tau' - \tau_*\|$ is lower bounded as:

$$\|\tau'-\tau_*\| \geq \|\tau-\tau_*\| - \eta\|\nabla L(\tau)\| \overset{(i)}{\geq} |b-R| - \eta\|\nabla L(\tau)\| \overset{(ii)}{\geq} |b-R| - 32\eta b = \left(\left|1-\frac{R}{b}\right| - 32\eta\right)b, \tag{4.26}$$

where step (i) uses $\|\tau - \tau^*\| \geq |\tau[1] - \tau^*[1]| \geq \big||\tau_1| - |\tau_1^*|\big| = |b - R|$, and step (ii) is by Claim 6.

On the other hand, we have $\|\tau' - \tau_*\| \leq 0.15R$ by assumption, which when combined with equation 4.26 gives $b \leq \frac{0.15R}{|1-\frac{R}{b}|-32\eta}$, or $b = O(R)$. This means $\|\tau - \tau_*\| - \|\tau' - \tau_*\| \leq \eta\|\nabla L(\tau)\| = o(1) \cdot O(\max\{R, |b|\}) = o(R)$. However, we also have $\|\tau - \tau_*\| - \|\tau' - \tau_*\| \geq 0.05R = \Theta(R)$ by assumption. This is a contradiction, which means the assumption must be false, i.e. τ' cannot satisfy $\|\tau' - \tau_*\|_2 \leq 0.15R$. □

Let's now finish the calculations in the proof of Lemma 49.

Calculation details for Equation 4.16 and 4.18 We now calculate term $T_i^{(0}$ and $T_i^{(1)}$ used in equation 4.16 and 4.18.

$$
\begin{aligned}
T_1^{(0)} &= \int_{x<\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2}+bx-c\right) = \exp\left(\frac{b^2}{2}-c\right)\int_{x<\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-b)^2}{2}\right) \\
&= \exp\left(\frac{b^2}{2}-c\right)\int_{x<\frac{c-\log\sqrt{2\pi}}{b}-b} \exp\left(-\frac{x^2}{2}\right) \\
&\simeq \begin{cases} \exp\left(\frac{b^2}{2}-c\right)\cdot \frac{1}{b-\frac{c-\log\sqrt{2\pi}}{b}}\cdot \exp\left(-\frac{1}{2}\left(\frac{c-\log\sqrt{2\pi}}{b}-b\right)^2\right), & c-\log\sqrt{2\pi}<b^2 \\ \exp\left(\frac{b^2}{2}-c\right)\cdot\left[1-\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-b}\cdot\exp\left(-\frac{1}{2}\left(\frac{c-\log\sqrt{2\pi}}{b}-b\right)^2\right)\right], & c-\log\sqrt{2\pi}\ge b^2 \end{cases} \qquad (4.27) \\
&= \begin{cases} \frac{1}{\sqrt{2\pi}}\frac{1}{b-\frac{c-\log\sqrt{2\pi}}{b}}\cdot\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & c-\log\sqrt{2\pi}<b^2, \\ \exp\left(\frac{b^2}{2}-c\right)-\frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-b}\cdot\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & c-\log\sqrt{2\pi}\ge b^2. \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
T_2^{(0)} &= \int_{x\ge\frac{c-\log\sqrt{2\pi}}{b}} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \\
&\simeq \begin{cases} \frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & c-\log\sqrt{2\pi}>0, \\ 1-\frac{1}{\sqrt{2\pi}}\frac{1}{\frac{|c-\log\sqrt{2\pi}|}{b}}\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & c-\log\sqrt{2\pi}<0. \end{cases} \qquad (4.28)
\end{aligned}
$$

$$
\begin{aligned}
T_3^{(0)} &= \int_{x<\frac{c-\log\sqrt{2\pi}}{b}} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-R)^2}{2}\right) = \int_{x<\frac{c-\log\sqrt{2\pi}}{b}-R}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \\
&= \begin{cases} \frac{1}{\sqrt{2\pi}}\frac{1}{R-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right), & c-\log\sqrt{2\pi}<bR, \\ 1-\frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-R}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right), & c-\log\sqrt{2\pi}\ge bR. \end{cases} \qquad (4.29)
\end{aligned}
$$

$$
\begin{aligned}
T_4^{(0)} &= \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\int_{x\ge\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(x-(R-b))^2}{2}\right) \\
&= \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\int_{x\ge\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{x^2}{2}\right) \\
&= \begin{cases} \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\left[1-\frac{1}{R-b-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(R-b-\frac{c-\log\sqrt{2\pi}}{b})^2}{2}\right)\right], & c-\log\sqrt{2\pi}<b(R-b) \\ \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{(R-b-\frac{c-\log\sqrt{2\pi}}{b})^2}{2}\right), & c-\log\sqrt{2\pi}\ge b(R-b) \end{cases} \\
&= \begin{cases} \frac{1}{2\pi}\exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}\right)-\frac{1}{\sqrt{2\pi}}\frac{1}{R-b-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{\left(\frac{c-\log\sqrt{2\pi}}{b}-R\right)^2}{2}\right) & c-\log\sqrt{2\pi}<b(R-b), \\ \frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{\left(\frac{c-\log\sqrt{2\pi}}{b}-R\right)^2}{2}\right), & c-\log\sqrt{2\pi}\ge b(R-b). \end{cases}
\end{aligned}
$$

(4.30)

$$
\begin{aligned}
T_1^{(1)} &= \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \exp\left(-\frac{x^2}{2}+bx-c\right)x \\
&= \exp\left(\frac{b^2}{2}-c\right)\int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \exp\left(-\frac{(x-b)^2}{2}\right)(x-b) + b\cdot\int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \exp\left(-\frac{(x-b)^2}{2}\right) \\
&\leq \exp\left(\frac{b^2}{2}-c\right)\int_{x\in[-b,\frac{c-\log\sqrt{2\pi}}{b}-b]} \exp\left(-\frac{x^2}{2}\right)x + bT_1^{(0)} \\
&= \exp\left(\frac{b^2}{2}-c\right)\left[-\exp\left(-\frac{x^2}{2}\right)\right]_{-b}^{\frac{c-\log\sqrt{2\pi}}{b}-b} + bT_1^{(0)} \\
&= \exp\left(\frac{b^2}{2}-c\right)\left(\exp\left(-\frac{b^2}{2}\right)-\exp\left(-\frac{(c-\log\sqrt{2\pi}-b^2)^2}{2b^2}\right)\right) + bT_1^{(0)} \\
&= \exp(-c) - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right) + bT_1^{(0)}.
\end{aligned} \tag{4.31}
$$

$$
\begin{aligned}
T_2^{(1)} &= \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)x = \frac{1}{\sqrt{2\pi}}\left[-\exp\left(-\frac{x^2}{2}\right)\right]_{\frac{c-\log\sqrt{2\pi}}{b}}^{\infty} \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right).
\end{aligned} \tag{4.32}
$$

$$
\begin{aligned}
T_3^{(1)} &= \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-R)^2}{2}\right)x \\
&\leq \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-R)^2}{2}\right)(x-R) + RT_3^{(0)} \\
&= \int_{x\in[-R,\frac{c-\log\sqrt{2\pi}}{b}-R]} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)x + RT_3^{(0)} = \frac{1}{\sqrt{2\pi}}\left[-\exp\left(-\frac{x^2}{2}\right)\right]_{-R}^{\frac{c-\log\sqrt{2\pi}}{b}-R} + RT_3^{(0)} \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{2}\right) - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(c-\log\sqrt{2\pi}-bR)^2}{2b^2}\right) + RT_3^{(0)}.
\end{aligned} \tag{4.33}
$$

$$
\begin{aligned}
T_4^{(1)} &= \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-(R-b))^2}{2}\right)(x-(R-b)) + (R-b)T_4^{(0)} \\
&= \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-(R-b)} \exp\left(-\frac{x^2}{2}\right)x + (R-b)T_4^{(0)} \\
&= \exp\left(\frac{(R-b)^2}{2}+c-\frac{R^2}{2}-2\log\sqrt{2\pi}\right)\left[-\exp\left(-\frac{x^2}{2}\right)\right]_{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}^{\infty} + (R-b)T_4^{(0)} \\
&= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(c-\log\sqrt{2\pi}-bR)^2}{2b^2}\right) + (R-b)T_4^{(0)}.
\end{aligned} \tag{4.34}
$$

Calculations for $b < 0$ We now calculate the gradient norm bound for the case where $b < 0$. Recall that:

$$\|\nabla L(\tau)\|_2 \leq \|\nabla L(\tau)\|_1 = \left| \int_x \frac{p - p_*}{\frac{p}{q} + 1} x \right| + \left| \int_x \frac{p - p_*}{\frac{p}{q} + 1} \right|. \tag{4.35}$$

Let's bound each term individually.

Bounding $\left| \int_x \frac{p-p_*}{\frac{p}{q}+1} \right|$*:*

$$\begin{aligned}
&\left| \int_x \frac{p - p_*}{\frac{p}{q} + 1} \right| = \left| \int_x \frac{\exp\left(-\frac{x^2}{2} + bx - c\right) - \exp\left(-\frac{(x-R)^2}{2} - \log\sqrt{2\pi}\right)}{\exp\left(bx - c + \log\sqrt{2\pi}\right) + 1} \right| \\
&\leq \underbrace{\int_{x < \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} - \log\sqrt{2\pi}\right)}_{T_{1,-}^{(0)}} + \underbrace{\int_{x \geq \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + bx - c\right)}_{T_{2,-}^{(0)}} \\
&\quad + \underbrace{\int_{x < \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + (R-b)x + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)}_{T_{3,-}^{(0)}} + \underbrace{\int_{x \geq \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-R)^2}{2} - \log\sqrt{2\pi}\right)}_{T_{4,-}^{(0)}} \\
&= O(1),
\end{aligned} \tag{4.36}$$

where $T_{i,-}^{(0)}$ terms are calculated as:

$$\begin{aligned}
T_{1,-}^{(0)} &= \int_{x < \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} - \log\sqrt{2\pi}\right) \\
&\simeq \begin{cases} \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{-\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & \frac{c-\log\sqrt{2\pi}}{b} < 0, \\ 1 - \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & \frac{c-\log\sqrt{2\pi}}{b} > 0, \end{cases}
\end{aligned} \tag{4.37}$$

$$\begin{aligned}
T_{2,-}^{(0)} &= \int_{x \geq \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{x^2}{2} + bx - c\right) = \exp\left(\frac{b^2}{2} - c\right) \int_{x \geq \frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-b)^2}{2}\right) \\
&= \exp\left(\frac{b^2}{2} - c\right) \int_{x \geq \frac{c-\log\sqrt{2\pi}}{b} - b} \exp\left(-\frac{x^2}{2}\right) \\
&\simeq \begin{cases} \exp\left(\frac{b^2}{2} - c\right) \cdot \left[1 - \frac{1}{b - \frac{c-\log\sqrt{2\pi}}{b}} \cdot \exp\left(-\frac{1}{2}\left(\frac{c-\log\sqrt{2\pi}}{b} - b\right)^2\right)\right], & \frac{c-\log\sqrt{2\pi}}{b} - b < 0 \\ \exp\left(\frac{b^2}{2} - c\right) \cdot \frac{1}{\frac{c-\log\sqrt{2\pi}}{b} - b} \cdot \exp\left(-\frac{1}{2}\left(\frac{c-\log\sqrt{2\pi}}{b} - b\right)^2\right), & \frac{c-\log\sqrt{2\pi}}{b} - b > 0 \end{cases} \\
&= \begin{cases} \exp\left(\frac{b^2}{2} - c\right) - \frac{1}{\sqrt{2\pi}} \frac{1}{b - \frac{c-\log\sqrt{2\pi}}{b}} \cdot \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & \frac{c-\log\sqrt{2\pi}}{b} - b < 0, \\ \frac{1}{\sqrt{2\pi}} \frac{1}{\frac{c-\log\sqrt{2\pi}}{b} - b} \cdot \exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right), & \frac{c-\log\sqrt{2\pi}}{b} - b > 0, \end{cases}
\end{aligned} \tag{4.38}$$

$$T^{(0)}_{3,-} = \exp\left(\frac{(R-b)^2}{2} + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)\int_{x<\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-(R-b))^2}{2}\right)$$
$$= \exp\left(\frac{(R-b)^2}{2} + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)\int_{x<\frac{c-\log\sqrt{2\pi}}{b}-(R-b)} \exp\left(-\frac{x^2}{2}\right)$$
$$= \begin{cases} \exp\left(\frac{(R-b)^2}{2} + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)\frac{1}{R-b-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(R-b-\frac{c-\log\sqrt{2\pi}}{b})^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b}-(R-b)<0 \\ \exp\left(\frac{(R-b)^2}{2} + c - \frac{R^2}{2} - 2\log\sqrt{2\pi}\right)\left[1 - \frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{(R-b-\frac{c-\log\sqrt{2\pi}}{b})^2}{2}\right)\right], & \frac{c-\log\sqrt{2\pi}}{b}-(R-b)>0 \end{cases}$$
$$= \begin{cases} \frac{1}{2\pi}\exp\left(\frac{(R-b)^2}{2} + c - \frac{R^2}{2}\right) - \frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{\left(\frac{c-\log\sqrt{2\pi}}{b}-R\right)^2}{2}\right) & \frac{c-\log\sqrt{2\pi}}{b}-(R-b)>0, \\ \frac{1}{\sqrt{2\pi}}\frac{1}{R-b-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{\left(\frac{c-\log\sqrt{2\pi}}{b}-R\right)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b}-(R-b)<0, \end{cases} \tag{4.39}$$

$$T^{(0)}_{4,-} = \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-R)^2}{2}\right) = \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-R}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$$
$$= \begin{cases} \frac{1}{\sqrt{2\pi}}\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-R}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b}-R>0, \\ 1-\frac{1}{\sqrt{2\pi}}\frac{1}{R-\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b}-R<0. \end{cases} \tag{4.40}$$

Bounding $\left|\int_x \frac{p-p_*}{\frac{p}{q}+1}x\right|$*:*

$$\left|\int_x \frac{p-p_*}{\frac{p}{q}+1}x\right| \leq \left|\int_x \frac{p}{\frac{p}{q}+1}x\right| + \left|\int_x \frac{p_*}{\frac{p}{q}+1}x\right|$$
$$\leq \max\left\{\int_{x>0}\frac{p}{\frac{p}{q}+1}x,\ -\int_{x<0}\frac{p}{\frac{p}{q}+1}x\right\} + \max\left\{\int_{x>0}\frac{p_*}{\frac{p}{q}+1}x,\ -\int_{x<0}\frac{p_*}{\frac{p}{q}+1}x\right\}. \tag{4.41}$$

As before, we will show the bound for the case where $x>0$; the other case (i.e. $x<0$) follows a similar calculation and has an upper bound on the same order.

First consider $b<0$, $c-\log\sqrt{2\pi}>0$:

$$\int_{x>0}\frac{p}{\frac{p}{q}+1}x + \int_{x>0}\frac{p_*}{\frac{p}{q}+1}x \leq \int_{x>0}px + \int_{x>0}p_*x \overset{(i)}{\simeq} \frac{1}{b^2+1}\exp(-c) + 1 + \frac{1}{1+R^2}\exp\left(-\frac{R^2}{2}\right), = O(1) \tag{4.42}$$

where step (i) uses the following:

$$\int_{x>0}\exp\left(-\frac{x^2}{2}+bx-c\right)x = \exp\left(\frac{b^2}{2}-c\right)\int_{x>0}\exp\left(-\frac{(x-b)^2}{2}\right)(x-b+b)$$
$$= \exp\left(\frac{b^2}{2}-c\right)\left[\int_{x>-b}\exp\left(-\frac{x^2}{2}\right)x + b\int_{x>-b}\exp\left(-\frac{x^2}{2}\right)\right]$$
$$= \exp\left(\frac{b^2}{2}-c\right)\left[\exp\left(-\frac{b^2}{2}\right) - \frac{b^2}{b^2+1}\exp\left(-\frac{b^2}{2}\right)\right] = \frac{1}{b^2+1}\exp(-c)$$
$$\int_{x>0}\exp\left(-\frac{(x-R)^2}{2}\right)x \leq \exp\left(-\frac{R^2}{2}\right) + 1 - \frac{R^2}{1+R^2}\exp\left(-\frac{R^2}{2}\right) = 1 + \frac{1}{1+R^2}\exp\left(-\frac{R^2}{2}\right). \tag{4.43}$$

When $b < 0$, $c - \log\sqrt{2\pi} < 0$,

$$\int_{x>0} \frac{p}{\frac{p}{q}+1}x + \int_{x>0} \frac{p_*}{\frac{p}{q}+1}x = \underbrace{\int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} qx}_{T^{(1)}_{1,-}} + \underbrace{\int_{x\geq \frac{c-\log\sqrt{2\pi}}{b}} px}_{T^{(1)}_{2,-}} + \underbrace{\int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} \frac{p_* q}{p}}_{T^{(1)}_{3,-}} + \underbrace{\int_{x\geq \frac{c-\log\sqrt{2\pi}}{b}} p_*}_{T^{(1)}_{4,-}}$$

$$\leq 16\max\{R, |b|\}, \tag{4.44}$$

where $T^{(1)}_{i,}$ terms are calculated as:

$$T^{(1)}_{1,-} = \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]} qx = \left[-\exp\left(-\frac{x^2}{2}\right)\right]_0^{\frac{c-\log\sqrt{2\pi}}{b}} = 1-\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right). \tag{4.45}$$

$$\begin{aligned} T^{(1)}_{2,-} &= \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} px = \exp\left(\frac{b^2}{2}-c\right)\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} \exp\left(-\frac{(x-b)^2}{2}\right)x \\ &= \exp\left(\frac{b^2}{2}-c\right)\left[\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-b}\exp\left(-\frac{x^2}{2}\right)x + b\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-b}\exp\left(-\frac{x^2}{2}\right)\right] \\ &\simeq \left(1-\frac{1}{1-\frac{c-\log\sqrt{2\pi}}{b^2}}\right)\cdot\exp\left(-\frac{(c-\log\sqrt{2\pi})^2}{2b^2}\right). \end{aligned} \tag{4.46}$$

$$\begin{aligned} T^{(1)}_{3,-} &= \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]}\frac{p_* q}{p} \simeq \int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]}\exp\left(-\frac{(x-R)^2}{2}-bx+c-\log\sqrt{2\pi}\right) \\ &= \exp\left(\frac{(R-b)^2}{2}-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)\int_{x\in[0,\frac{c-\log\sqrt{2\pi}}{b}]}\exp\left(-\frac{(x-(R-b))^2}{2}\right) \\ &= \exp\left(\frac{(R-b)^2}{2}-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)\int_{x\in[-(R-b),\frac{c-\log\sqrt{2\pi}}{b}-(R-b)]}\exp\left(-\frac{x^2}{2}\right)(x+R-b) \\ &= \exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)-\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right)+(R-b)\cdot\beta^{(1)}_{3,-}, \end{aligned} \tag{4.47}$$

where $\beta^{(1)}_{3,-} = O(1)$ is:

$$\beta^{(1)}_{3,-} = \begin{cases} 2-\frac{1}{R-b}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right)-\frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b}\geq R-b, \\ \frac{1}{\frac{c-\log\sqrt{2\pi}}{b}-(R-b)}\exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right)-\frac{1}{R-b}\exp\left(-\frac{R^2}{2}+c-\log\sqrt{2\pi}\right), & \frac{c-\log\sqrt{2\pi}}{b}<R-b. \end{cases} \tag{4.48}$$

$$\begin{aligned} T^{(1)}_{4,-} &= \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}} p_* = \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}}\exp\left(-\frac{(x-R)^2}{2}\right)(x-R+R) \\ &= \int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-R}\exp\left(-\frac{x^2}{2}\right)x + R\int_{x\geq\frac{c-\log\sqrt{2\pi}}{b}-R}\exp\left(-\frac{x^2}{2}\right) \\ &= \exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b}-R)^2}{2}\right)+R\cdot\beta^{(1)}_{4,-}, \end{aligned} \tag{4.49}$$

where $\beta_{4,-}^{(1)} = O(1)$ is:

$$\beta_{4,-}^{(1)} = \begin{cases} 1 - \frac{1}{R - \frac{c - \log\sqrt{2\pi}}{b}} \exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b} - R)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b} < R, \\ \frac{1}{\frac{c-\log\sqrt{2\pi}}{b} - R} \exp\left(-\frac{(\frac{c-\log\sqrt{2\pi}}{b} - R)^2}{2}\right), & \frac{c-\log\sqrt{2\pi}}{b} > R. \end{cases} \tag{4.50}$$

Combining equation 4.36, 4.42, and 4.41 we have that $\|\nabla L([b,c])\|_2 \leq 32 \max\{R, |b|\}$ for $b < 0$.

4.1.8 Proof of Lemma 46, 47: smoothness and strong convexity at the optimum

We prove Lemmas 46 and 47 in this section. First recall the lemma statements:

Lemma 56 (Smoothness at $P = P^*$, Lemma 46 restated)**.** *Consider the 1d Gaussian mean estimation task with $R := |\theta_* - \theta_q| \gg 1$. Then the smoothness at $P = P_*$ is upper bounded as:*

$$\sigma^*_{\max} := \sigma_{\max}(\nabla^2 L(\tau_*)) \leq \frac{R}{\sqrt{2\pi}} \exp(-R^2/8). \tag{4.51}$$

We will also need a bound on the strong convexity constant (i.e. smallest singular value) at $P = P^*$:

Lemma 57 (Strong convexity at $P = P^*$, Lemma 47 restated)**.** *Under the same setup as lemma 46, the minimum singular value at $P = P_*$ is $\sigma^*_{\min}(\nabla^2 L(\tau_*)) = \Theta\left(\frac{1}{R}\exp\left(-\frac{R^2}{8}\right)\right)$.*

Proof of Lemma 46 (smoothness at $P = P_*$) We will show the smoothness constant (i.e. $\sigma_{\max}(\nabla^2 L)$) is exponentially small at the optimum, i.e. when $P = P_*$. The Hessian at the optimum is:

$$\nabla^2 L(\tau) = \frac{1}{2}\int_x \frac{p_* q}{p_* + q} T(x)T(x)^\top dx = \frac{1}{2}\int_x \frac{p_* q}{p_* + q} [x, -1]^\top [x, -1]\, dx. \tag{4.52}$$

Recall that $\theta_q = 0$ w.l.o.g, and assume $\theta_* = R \gg 1$. Then

$$\begin{aligned} \nabla^2 L(\tau) =& \frac{1}{2}\int_{x \leq R/2} \frac{p_* q}{p_* + q} T(x)T(x)^\top dx + \frac{1}{2}\int_{x > R/2} \frac{p_* q}{p_* + q} T(x)T(x)^\top dx \\ \lesssim& \frac{1}{2}\int_{x \leq R/2} p_* T(x)T(x)^\top dx + \frac{1}{2}\int_{x > R/2} q T(x)T(x)^\top dx. \end{aligned} \tag{4.53}$$

Let $\mathcal{S}^1 \subset \mathbb{R}^2$ denote the circle centered as the origin with radius 1. The maximum singular value is upper bounded by

$$\begin{aligned}
&\sigma^*_{\max} := \max_{[a_1,a_2]\in\mathcal{S}^1} \frac{1}{2}\int_x \frac{p_* q}{p_*+q}\left(a_1 x - a_2\right)^2 dx \\
\le&\frac{1}{2}\max_{[a_1,a_2]\in\mathcal{S}^1}\left[\int_{x\le R/2} p_*\left(a_1x-a_2\right)^2 dx + \int_{x>R/2} q\left(a_1x-a_2\right)^2 dx\right] \\
=&\frac{1}{2}\max_{[a_1,a_2]\in\mathcal{S}^1}\left[\int_{x\le R/2} p^*\left(a_1^2x^2-2a_1a_2x+a_2^2\right)dx+\int_{x>R/2} q\left(a_1^2x^2-2a_1a_2x+a_2^2\right)dx\right] \\
=&\frac{1}{2}\max_{[a_1,a_2]\in\mathcal{S}^1}\left[\underbrace{\left(\int_{x\le\frac{R}{2}} p_*x^2+\int_{x>\frac{R}{2}} qx^2\right)}_{T_2}\cdot a_1^2-\underbrace{\left(\int_{x\le\frac{R}{2}} p_*x+\int_{x>\frac{R}{2}} qx\right)}_{T_1}2a_1a_2+\underbrace{\left(\int_{x\le\frac{R}{2}} p_*+\int_{x>\frac{R}{2}} q\right)}_{T_0}a_3^2\right] \\
\overset{(i)}{\le}&\frac{1}{2}T_2+T_1+\frac{T_0}{2}\overset{(ii)}{\le}\left(\frac{R}{2}+\frac{1}{R}+2+\frac{2}{R}\right)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right) \\
=&\left(\frac{R}{2}+2+\frac{3}{R}\right)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\le\frac{R}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right),
\end{aligned} \tag{4.54}$$

where (i) substitutes in 1 or -1 for a_1, a_2 and uses the fact that the upper bounds for T_0, T_1, T_2 are positive. (ii) uses the calculations on T_0 to T_2 shown below. We note that these calculations rely on properties of Gaussian and do not extend to general exponential families.

$$\begin{aligned}
T_0 =&2\int_{x>R/2} q \le \frac{4}{R}\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right), \\
T_1 =&\int_{x\le R/2}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-\theta)^2}{2}\right)xdx+\int_{x>R/2}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)xdx \\
=&\int_{x'\ge R/2}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x')^2}{2}\right)(R-x')dx+\int_{x>R/2}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)xdx \\
=&R\int_{x\ge R/2} qdx\le\frac{2}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right).
\end{aligned} \tag{4.55}$$

For T_2, denote $P_{R/2} := P_*\left(\left\{x : x \le \frac{R}{2}\right\}\right) = P_Q\left(\left\{x : x \ge \frac{R}{2}\right\}\right)$; Gaussian tail bound gives $P_{R/2} \le$

$\frac{2}{\sqrt{2\pi}}\frac{1}{R}\exp\left(-\frac{R^2}{8}\right)$. Then we can calculate each term in T_2 as:

$$
\begin{aligned}
\int_{x\le\frac{R}{2}} p_*(x)x^2dx &= \int_{x\le\frac{R}{2}} \frac{1}{\sqrt{2\pi}}\exp\Big(-\frac{(x-R)^2}{2}\Big)x^2dx\\
&= \int_{x\le\frac{R}{2}} \frac{1}{\sqrt{2\pi}}\exp\Big(-\frac{(x-R)^2}{2}\Big)(x-R)\cdot xdx + R\int_{x\le\frac{R}{2}}\frac{1}{\sqrt{2\pi}}\exp\Big(-\frac{(x-R)^2}{2}\Big)xdx\\
&= \left[-\frac{\exp\Big(-\frac{(x-R)^2}{2}\Big)x}{\sqrt{2\pi}}\right]_{-\infty}^{\frac{R}{2}} + \int_{x\le\frac{R}{2}}\frac{\exp\Big(-\frac{(x-R)^2}{2}\Big)}{\sqrt{2\pi}}(x-R)dx + R\left(R\cdot P_{R/2} - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\right)\\
&= -\left(\frac{R}{2}+1\right)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right) + R\left(R\cdot P_{R/2} - \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\right)\\
&= -\left(\frac{3R}{2}+1\right)\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right) + R^2\cdot P_{R/2}\\
&\le -\frac{3R}{2}\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right) + R^2\cdot P_{R/2},\\
\int_{x\ge\frac{R}{2}} q(x)x^2dx &= \int_{x\ge\frac{R}{2}}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)x^2dx = \left[-\frac{\exp(-\frac{x^2}{2})x}{\sqrt{2\pi}}\right]_{\frac{R}{2}}^{\infty} + P_{R/2} = \frac{1}{\sqrt{2\pi}}\frac{R}{2}\exp\left(-\frac{R^2}{8}\right) + P_{R/2}.
\end{aligned}
\tag{4.56}
$$

Combining both terms gives $T_2 \le -R\cdot\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right) + (R^2+1)P_{R/2} \le \left(R+\frac{2}{R}\right)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)$.

Proof of Lemma 47 (strong convexity at $P = P_*$) Lower bounding $\sigma^*_{\min}$ follows a similar calculation as for upper bounding $\sigma^*_{\max}$:

$$
\begin{aligned}
\sigma^*_{\min} &:= \min_{[a_1,a_2]\in\mathcal{S}^1}\frac{1}{2}\int_x\frac{p_*q}{p_*+q}(a_1x-a_2)^2dx \gtrsim \min_{[a_1,a_2]\in\mathcal{S}^1}\frac{1}{4}\int_x\frac{p_*q}{\max\{p_*,q\}}(a_1x-a_2)^2dx\\
=&\frac{1}{4}\min_{[a_1,a_2]\in\mathcal{S}^1}\left[\int_{x\le R/2}p_*(a_1x-a_2)^2dx + \int_{x>R/2}q(a_1x-a_2)^2dx\right]\\
=&\frac{1}{4}\min_{[a_1,a_2]\in\mathcal{S}^1}\left[\int_{x\le R/2}p^*\left(a_1x^2-2a_1a_2x+a_2^2\right)dx + \int_{x>R/2}q\left(a_1x^2-2a_1a_2x+a_2^2\right)dx\right]\\
=&\frac{1}{4}\min_{[a_1,a_2]\in\mathcal{S}^1}\left[\underbrace{\left(\int_{x\le\frac{R}{2}}p_*x^2+\int_{x>\frac{R}{2}}qx^2\right)}_{T_2}\cdot a_1^2 - \underbrace{\left(\int_{x\le\frac{R}{2}}p_*x+\int_{x>\frac{R}{2}}qx\right)}_{T_1}2a_1a_2 + \underbrace{\left(\int_{x\le\frac{R}{2}}p_*+\int_{x>\frac{R}{2}}q\right)}_{T_0}a_2^2\right]\\
\overset{(i)}{\ge}&\frac{1}{4}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\min_{[a_1,a_2]\in\mathcal{S}^1}\left[\left(\frac{R}{2}+\frac{1}{R}\right)a_1^2 - 4a_1a_2 + \frac{1}{R}a_2^2\right]\\
=&\frac{1}{4}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\min_{a\in[0,1]}\left[\left(\frac{R}{2}+\frac{1}{R}\right)a^2 - 4a\sqrt{1-a^2} + \frac{1}{R}(1-a^2)\right]\\
=&\frac{1}{4}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\min_{a\in[0,1]}\left[\frac{R}{2}a^2 - 4a\sqrt{1-a^2}+\frac{1}{R}\right] = \frac{1}{4}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right)\min_{a\in[0,1]}\left[a\left(\frac{R}{2}a-4\sqrt{1-a^2}\right)+\frac{1}{R}\right]\\
\overset{(ii)}{\ge}&\frac{1}{4R}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{R^2}{8}\right),
\end{aligned}
\tag{4.57}
$$

where (i) uses the calculations on T_0 to T_2 stated in equation 4.55 and 4.56. Step (ii) replaces $a = 0$ to remove the $O(R)$ term.

4.1.9 Proof of Lemma 48: curvature at $P = Q$

Lemma 58 (Smoothness at $P = Q$, Lemma 48 restated)**.** *Under the same setup as Lemma 46, the smoothness at $P = Q$ is lower bounded as $\sigma_{\max}(\nabla^2 L(\tau_q)) \geq \frac{R^2}{2}$.*

Proof. The result follows from direct calculation of the Hessian at $P = Q$:

$$
\begin{aligned}
\nabla^2 L(\tau) =& \frac{1}{2}\int_x \frac{p_* + q}{4} T(x)T(x)^\top dx = \frac{1}{8}\Big(\mathbb{E}_*(T(x)T(x)^\top) + \mathbb{E}_Q(T(x)T(x)^\top)\Big) \\
=& \frac{1}{8}\left(\mathbb{E}_*\left(\begin{bmatrix} x \\ -1 \end{bmatrix}[x, -1]\right) + \mathbb{E}_Q\left(\begin{bmatrix} x \\ -1 \end{bmatrix}[x, -1]\right)\right) = \frac{1}{8}\begin{bmatrix} \mathbb{E}_* x^2 + \mathbb{E}_Q x^2 & 0 \\ 0 & 2 \end{bmatrix} \\
=& \frac{1}{8}\begin{bmatrix} R^2 + 2 & 0 \\ 0 & 2 \end{bmatrix}.
\end{aligned}
\tag{4.58}
$$

Hence $\sigma_{\max}(\nabla_\tau^2 L) \geq \boldsymbol{e}_1^\top \nabla^2 L(\tau)\boldsymbol{e}_1 \geq \frac{R^2}{2}$. □

4.1.9.1 Proof of Theorem 18: lower bound for second-order methods

The proof of Theorem 18 is similar to that of Theorem 17, where we show that there is a ring of width $\Theta(R)$ in which the amount of progress at each step is exponentially small, hence the number of steps required to cross this ring is exponential.

We show that starting from $\tau_0 = \tau_q$, the optimization path will necessarily steps into $\mathcal{A}$:

Lemma 59. *Let $\eta := O(\frac{\lambda_\rho}{\lambda_M})$, where $\lambda_\rho := \min_{\theta\in\Theta} \sigma_{\min}(\nabla^2 L(\tau_\theta))$, $\lambda_M := \max_{\theta\in\Theta} \sigma_{\max}(\nabla^2 L(\tau_\theta))$ as defined in Section 4.1.3. For any τ s.t. $\|\tau - \tau_*\| \geq 0.2R$, let τ' denote the point after one step of gradient descent from τ, then $\|\tau' - \tau_*\|_2 > 0.15R$.*

Proof. First note that $\forall \tau$, the next point after one step of Newton update is:

$$
\tau_{t'} = \tau - \eta(\nabla^2 L(\tau))^{-1}\nabla L(\tau) = \tau - \eta\Big[\Big\langle(\nabla^2 L(\tau))^{-1}\nabla L(\tau), \frac{\tau - \tau_*}{\|\tau - \tau_*\|_2}\Big\rangle \cdot \frac{\tau - \tau_*}{\|\tau - \tau_*\|_2} + \boldsymbol{v}\Big], \tag{4.59}
$$

where $\boldsymbol{v} := (\nabla^2 L(\tau))^{-1}\nabla L(\tau) - \big\langle(\nabla^2 L(\tau))^{-1}\nabla L(\tau), \frac{\tau-\tau_*}{\|\tau-\tau_*\|_2}\big\rangle \cdot \frac{\tau-\tau_*}{\|\tau-\tau_*\|_2}$ is orthogonal to $\tau - \tau_*$. This means

$$
\begin{aligned}
&\|\tau - \tau_*\| - \|\tau' - \tau_*\| = \eta\Big\langle(\nabla^2 L(\tau))^{-1}\nabla L(\tau), \frac{\tau - \tau_*}{\|\tau - \tau_*\|_2}\Big\rangle - \eta\|\boldsymbol{v}\| \\
\leq& \frac{\eta}{\sigma_{\min}(\nabla^2 L(\tau))} \cdot \left|\Big\langle \nabla L(\tau), \frac{\tau - \tau_*}{\|\tau - \tau_*\|_2}\Big\rangle\right| \leq \frac{\eta\|\nabla L(\tau)\|_2}{\sigma_{\min}(\nabla^2 L(\tau))} \overset{(i)}{\leq} \frac{32\eta \max\{R, |b|\}}{\sigma_{\min}(\nabla^2 L(\tau))} \\
\overset{(ii)}{\leq}& 32\frac{\lambda_\rho}{\sigma_{\min}(\nabla^2 L(\tau))}\frac{\max\{R, |b|\}}{\lambda_M} \leq \frac{32\max\{R, |b|\}}{\lambda_M} \leq \frac{64\max\{R, |b|\}}{R^2},
\end{aligned}
\tag{4.60}
$$

where step (i) uses Claim 6, and step (ii) follows from the choice of η.

Suppose $\|\tau' - \tau_*\|_2 < 0.15R$, then

$$0.05R \leq \|\tau - \tau_*\| - \|\tau' - \tau_*\| \leq \frac{64\max\{R, |b|\}}{R^2} \Rightarrow b = \Omega(R^3). \tag{4.61}$$

However, $\|\tau' - \tau_*\|_2$ entails $b = \Theta(R)$, which is a contradiction. Hence it must be that $\|\tau' - \tau_*\|_2 > 0.15R$. ☐

Proof of Theorem 18. By Lemma 59, the optimization path will go to a point $\tau' \in \mathcal{A}$ s.t. $\|\tau' - \tau_*\|_2 > 0.15R$. From any such τ', the shortest way to exit the annulus $\mathcal{A}$ is to project onto the inner circle defining $\mathcal{A}$, i.e. the circle centered at τ_* with radius $0.1R$ which is a convex set. Denote this inner circle as $\mathcal{B}(\tau_*, 0.1R)$ whose projection is $\Pi_{\mathcal{B}(\tau_*,0.1R)}$, then the shortest path is the line segment $\tau' - \Pi_{\mathcal{B}(\tau_*,0.1R)}(\tau')$. Further, this line segment is of length $0.05R$ since $\|\tau' - \tau_*\| > 0.15R$ by Lemma 59.

However, the decrease of the parameter distance (i.e. $\|\tau - \tau_*\|$) is exponentially small at any point in $\mathcal{A}$:

$$\begin{aligned} \|\tau_t - \tau_*\| - \|\tau_{t+1} - \tau_*\| &\overset{(i)}{\leq} \frac{\eta}{\sigma_{\min}(\nabla^2 L(\tau_t))} \left|\left\langle \nabla L(\tau_t), \frac{\tau_t - \tau_*}{\|\tau_t - \tau_*\|_2} \right\rangle\right| \\ \overset{(ii)}{\leq} \frac{\left|\left\langle \nabla L(\tau_t), \frac{\tau_t - \tau_*}{\|\tau_t - \tau_*\|_2} \right\rangle\right|}{\lambda_M} &\overset{(iii)}{\leq} O\Big(\frac{\exp\left(-\frac{\kappa(b,c)R^2}{8}\right)}{R^3}\Big), \end{aligned} \tag{4.62}$$

where step (i) uses the calculations in equation 4.60; step (ii) use the choice of η; and step (iii) uses Lemma 49.

Hence the number of steps to exit $\mathcal{A}$ is lower bounded by $\frac{0.05R}{O(\frac{2}{R^2}\exp\left(-\frac{R^2}{8}\right))} = \Omega\left(R^3 \exp\left(\frac{R^2}{8}\right)\right)$.

☐

4.1.10 Proofs: NGD convergence in terms of Bhattacharyya coefficient (Section 4.1.4.2)

This section provides proofs for results in Section 4.1.4.2 relating to the Bhattacharyya coefficient. Recall that the *Bhattacharyya coefficient* of P_*, Q is defined as $\text{BC}(P_*, Q) := \int_x \sqrt{p_*(x)q(x)}dx$. We start by proving the convergence rate stated in terms of the Bhattacharyya coefficient (Theorem 20), and then prove the bound on Bhattacharyya coefficient (Lemma 52). The helper lemmas used in the proof of Theorem 20 are provided in the end.

Theorem (Convergence rate in terms of Bhattacharyya coefficient (Theorem 20, restated))**.** *Suppose Assumptions 11- 14 hold with constants ω, β_Z, $\lambda_{\max}$ and $\lambda_{\min}$, $\gamma_{\max}$ and $\gamma_{\min}$. Consider a NCE task with data distribution P_* and noise distribution Q, parameterized by $\theta_*, \theta_q \in \Theta$ respectively. Then for any given $\delta \leq \frac{1}{R}$ and initial estimate $\tau_0 = \tau_q$, NGD finds an estimate τ such that $\|\tau - \tau_*\|_2 \leq \delta$ within $T \leq C \cdot \frac{1}{BC(P_*,Q)^3}\frac{\|\tau_0 - \tau_*\|^2}{\delta^2}$ steps, where $C := 18\exp\left(\frac{2}{\beta_Z}\right) \cdot \left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)^3 \cdot \min\left\{\frac{2\lambda_{\max}^2}{\lambda_{\min}^2}, \frac{2\lambda_{\min} + \gamma_{\max}\|\bar{\delta}\|}{\lambda_{\min} - \gamma_{\min}\|\bar{\delta}\|}\right\}$.*

Proof. Proving Theorem 20 requires bounding the condition number κ_* and the Hessian-related constants β_u, β_l. The proof follows from the following two lemmas, which we prove in the end of this section.

The first lemma shows that κ_* is inversely related to $\text{BC}(P_*, Q)$:

Lemma 60. *Let Θ be the set of parameters for an exponential family satisfying Assumption 11-12. Then, for any pair of P_*, Q parameterized by $\theta_*, \theta_q \in \Theta$, the NCE problem defined with P_*, Q has $\kappa_* \leq \frac{\lambda_{\max}}{2\lambda_{\min}} \frac{1}{BC(P_*,Q)}$.*

The second lemma estimates the Hessian-related constants in Assumption 15:

Lemma 61. *Let $\bar{\delta} := \tau - \tau_*$. Let $BC(P_*, Q)$ denote the Bhattacharyya coefficient between P_* and Q, then for any τ such that $\|\bar{\delta}\| \leq \frac{1}{\beta_Z}$, we have:*

$$\begin{aligned}\frac{\sigma_{\max}(\nabla^2 L(\tau))}{\sigma_{\max}(\nabla^2 L(\tau_*))} &\leq \frac{1}{BC(P_*,Q)} \cdot 8\exp\left(\frac{3}{2} + \frac{1}{\beta_Z}\right) \cdot \frac{\lambda_{\max}}{\lambda_{\min}} \cdot \min\left\{\frac{2\lambda_{\max}}{\lambda_{\min}}, 2 + \frac{\gamma_{\max}\|\bar{\delta}\|}{\lambda_{\min}}\right\}, \\ \frac{\sigma_{\min}(\nabla^2 L(\tau))}{\sigma_{\min}(\nabla^2 L(\tau_*))} &\geq BC(P_*,Q) \cdot 16\exp\left(-2 - \frac{1}{\beta_Z}\right) \cdot \frac{\lambda_{\min}}{\lambda_{\max}} \cdot \max\left\{\frac{\lambda_{\min}}{\lambda_{\max}}, 1 - \frac{\gamma_{\min}\|\bar{\delta}\|}{\lambda_{\min}}\right\}.\end{aligned}$$

Hence Assumption 15 is satisfied with constants β_u, β_l equal to the respective right hand sides.

The factor C in the theorem statement is then chosen such that $\frac{C}{\mathrm{BC}(P_*,Q)^3} \geq \frac{\beta_u}{\beta_l}$, and the proof of Theorem 20 follows by applying Theorem 19 and the above lemmas. □

Next, we show that the Bhattacharyya coefficient $\mathrm{BC}(P_*, Q)$ can be lower bounded when the parameters θ_*, θ_q are close:

Lemma (Lemma 52 restated)**.** *For P_1, P_2 parameterized by $\theta_1, \theta_2 \in \Theta$, if $\|\theta_1 - \theta_2\|_2^2 \leq \frac{4}{\lambda_{\max}}$, then $BC(P_1, P_2) \geq \frac{1}{2}$.*

Proof. The proof relies on analyzing the geodesic on the manifold of square root densities $\sqrt{p}$ equipped with the Hellinger distance as a metric. Given $\theta_1, \theta_2 \in \Theta$, define a map ϕ from $[0,1]$ to a function $\sqrt{p}$, where p is the PDF for a distribution parameterized by some $\theta \in \Theta$: let $Z(\theta)$ denote the partition function for parameter $\theta \in \Theta$, and let $\delta := \theta_2 - \theta_1$, then $\phi(t)$ is a function of x defined as:

$$\phi(t)(x) = \sqrt{h(x)\exp\left((\theta_1 + t\delta)^\top x - \log Z(\theta_1 + t\delta)\right)}. \tag{4.63}$$

Denote $\phi_t(x) := \phi(t)(x)$ and $\theta_t := \theta_1 + t\delta$ for notation convenience. Then

$$\begin{aligned}\frac{\partial \phi_t(x)}{\partial t} =& \frac{\partial}{\partial t}\left(\frac{\sqrt{h}\exp\left(\frac{1}{2}\theta_t^\top x\right)}{\sqrt{Z(\theta_t)}}\right) = \frac{\sqrt{h}}{2}\exp\left(\frac{1}{2}\theta_t^\top x\right) \frac{\delta^\top x \cdot \sqrt{Z(\theta_t)} - \frac{1}{\sqrt{Z(\theta_t)}} \frac{\partial Z(\theta_t)}{\partial t}}{Z(\theta_t)} \\ \overset{(*)}{=}& \frac{\sqrt{h}}{2}\exp\left(\frac{1}{2}\theta_t^\top x\right) \frac{\delta^\top x - \mathbb{E}_{\theta_t}[\delta^\top x]}{\sqrt{Z(\theta_t)}} = \frac{1}{2}\sqrt{p_{\theta_t}(x)}(\delta^\top x - \mathbb{E}_{\theta_t}[\delta^\top x]),\end{aligned} \tag{4.64}$$

where step $(*)$ used

$$\frac{\partial Z(\theta_t)}{\partial t} = \frac{\partial}{\partial t}\int_x h(x)\exp\left(\theta_t^\top x\right) = \int_x h(x)\exp\left(\theta_t^\top x\right)\delta^\top x = Z(\theta_t)\mathbb{E}_{\theta_t}[\delta^\top x]. \tag{4.65}$$

Hence

$$\begin{aligned}\left\|\frac{\partial \phi_t}{\partial t}\right\|_{L_2} :=& \int_x \left(\frac{\partial \phi_t(x)}{\partial t}\right)^2 = \frac{\int_x p_{\theta_t}(x)\left(\delta^\top x - \mathbb{E}_{\theta_t}[\delta^\top x]\right)^2}{4} \\ =& \frac{\mathrm{Var}_{\theta_t}(\delta^\top x)}{4} = \frac{\delta^\top \mathbb{E}_{\theta_t}[xx^\top]\delta^\top}{4} \leq \frac{\lambda_{\max}}{4}\|\delta\|_2^2.\end{aligned} \tag{4.66}$$

Using the fundamental theorem of calculus, we get

$$\|\sqrt{p_{\theta_1}} - \sqrt{p_{\theta_2}}\|_{L_2} = \|\phi(1) - \phi(0)\|_{L_2} = \int_{t=0}^{1} \frac{\partial \phi_t(x)}{\partial t} dt \le \int_{t=0}^{1} \left\| \frac{\partial \phi_t(x)}{\partial t} \right\| dt \le \frac{\lambda_{\max}}{4} \|\delta\|_2^2. \tag{4.67}$$

It follows that $\int_x \sqrt{p_{\theta_1}}\sqrt{p_{\theta_2}} \ge 1 - \frac{\lambda_{\max}}{8}\|\delta\|^2$, or $\frac{1}{\int_x \sqrt{p_{\theta_1} p_{\theta_2}}} \le \frac{1}{1-\frac{\lambda_{\max}}{8}\|\delta\|^2}$ for $\|\delta\|^2 < \frac{8}{\lambda_{\max}}$. In particular, for any θ_1, θ_2 satisfying $\|\delta\|^2 := \|\theta_1 - \theta_2\|^2 \le \frac{4}{\lambda_{\max}}$, $\frac{1}{\int_x \sqrt{p_{\theta_1} p_{\theta_2}}} = \frac{1}{\mathrm{BC}(P,Q)} \le 2$, i.e. $\mathrm{BC}(P,Q) \ge \frac{1}{2}$. □

4.1.10.1 Proof of helper lemmas

We now return to proving the helper lemmas used in the proof of Theorem 20, i.e. Lemma 60 and Lemma 61.

Proof of Lemma 60. For exponential family with pdf $p(x) = h(x)\exp\left(\theta^\top x - \log Z(\theta)\right)$, the Hessian at the optimum is:

$$\boldsymbol{H}_* = \int_x \frac{p_* q}{p_* + q} T(x)T(x)^\top dx \preceq \int_x \min\{p_*, q\} T(x)T(x)^\top dx := \boldsymbol{M}. \tag{4.68}$$

We also have $\boldsymbol{H}_* \succeq \frac{1}{2}\boldsymbol{M}$ by noting that $p_* + q \le 2\max\{p_*, q\}$. Therefore in order to bound κ_*, it suffices to analyze the condition number of $\boldsymbol{M}$.

For any pair of distributions parameterized by $\theta, \theta_q \in \Theta$ with PDFs p, q, and for any unit vector $\boldsymbol{v}$, we have

$$\begin{aligned}
&\left(\int_x \sqrt{p}\sqrt{q}(\boldsymbol{v}^\top T(x))^2\right)^2 = \left(\int_x \min\{\sqrt{p},\sqrt{q}\}\max\{\sqrt{p},\sqrt{q}\}(\boldsymbol{v}^\top T(x))^2\right)^2 \\
&\overset{(i)}{\le} \left(\int_x (\min\{\sqrt{p},\sqrt{q}\})^2(\boldsymbol{v}^\top T(x))^2\right) \cdot \left(\int_x (\max\{\sqrt{p},\sqrt{q}\})^2(\boldsymbol{v}^\top T(x))^2\right) \\
&\le \left(\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2\right) \cdot \left(\int_x (p+q)(\boldsymbol{v}^\top T(x))^2\right) \overset{(ii}{\le} 2\lambda_{\max}\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2,
\end{aligned} \tag{4.69}$$

where (i) uses Cauchy-Schwarz, and (ii) uses assumption 13.

Denote $B := \frac{\sqrt{Z(\theta)Z(\theta_q)}}{Z\left(\frac{\theta+\theta_q}{2}\right)}$. We have:

$$\left(\int_x \sqrt{p}\sqrt{q}(\boldsymbol{v}^\top T(x))^2\right)^2 = \frac{Z\left(\frac{\theta+\theta_q}{2}\right)^2}{Z(\theta)Z(\theta_q)}\left(\int_x p_{\frac{\theta+\theta_q}{2}}(x)(\boldsymbol{v}^\top T(x))^2\right)^2 = \frac{1}{B^2}\left(\mathbb{E}_{\frac{\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\right)^2. \tag{4.70}$$

Combining equation 4.69, 4.70 gives a lower bound of $\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2$:

$$\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2 \ge \frac{1}{2\lambda_{\max}}\frac{1}{B^2}\left(\mathbb{E}_{\frac{\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\right)^2. \tag{4.71}$$

On the other hand, $\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2$ can also be upper bounded as:

$$\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2 \le \int_x \sqrt{p}\sqrt{q}(\boldsymbol{v}^\top T(x))^2 \le \frac{1}{B}\mathbb{E}_{\frac{\theta+\theta_q}{2}}\left[(\boldsymbol{v}^\top T(x))^2\right]. \tag{4.72}$$

Hence the condition number of $\boldsymbol{M}$ is bounded as:

$$\kappa(\boldsymbol{M}) := \frac{\max_v \int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2}{\min_v \int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2} \le \frac{\lambda_{\max} B}{2\min_{\boldsymbol{v}} \mathbb{E}_{\frac{\theta+\theta_q}{2}}\left[(\boldsymbol{v}^\top T(x))^2\right]} \le \frac{\lambda_{\max}}{2\lambda_{\min}} \cdot B. \tag{4.73}$$

It is left to determine the value of B. We claim that $B = \frac{1}{\mathrm{BC}(P,Q)}$, where $\mathrm{BC}(P,Q)$ is the Bhattacharyya coefficient of P and Q defined as $\mathrm{BC}(P,Q) := \int_x \sqrt{p(x)q(x)}dx$. To see this, note that it holds for any x that $\log Z_\theta = \theta^\top x + \log h(x) - \log p_\theta(x)$. Hence for any x,

$$B^{-1} = \exp\left(\log Z_{\frac{\theta+\theta_q}{2}} - \frac{1}{2}\log Z_\theta - \frac{1}{2}\log Z_{\theta_q}\right) = \frac{\sqrt{p_\theta(x)p_{\theta_q}(x)}}{p_{\frac{\theta+\theta_q}{2}}(x)}. \tag{4.74}$$

Therefore $B^{-1} = \left(\int_x p_{\frac{\theta+\theta_q}{2}}(x)\right) \cdot B^{-1} = \int_x \sqrt{p_\theta(x)p_{\theta_q}(x)} = \mathrm{BC}(P,Q)$. □

Proof for Lemma 61. For notational convenience, write $\bar{\delta} = [\bar{\theta}, \bar{\alpha}]$, where $\bar{\alpha} = \log Z(\theta_*) - \log Z(\theta)$ is the difference in the coordinate for the log partition function.

Upper bounding $\frac{\sigma_{\max}(\nabla^2 L(\tau))}{\sigma_{\max}(\nabla^2 L(\tau_*))}$: We proceed by splitting $\boldsymbol{v}^\top \nabla^2 L(\tau)\boldsymbol{v}$ into two terms:

$$\boldsymbol{v}^\top \nabla^2 L(\tau)\boldsymbol{v} = \int_{\bar{\delta}^\top T(x)<0} (p_* + q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2 dx + \int_{\bar{\delta}^\top T(x)>0} (p_* + q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2 dx. \tag{4.75}$$

The first term is bounded as:

$$
\begin{aligned}
&\int_{\bar\delta^\top T(x)<0}(p_*+q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2dx = \int_{\bar\delta^\top T(x)<0}(p_*+q)\frac{1}{\frac{p}{q}+\frac{q}{p}+2}(\boldsymbol{v}^\top T(x))^2dx\\
&\le \int_{\bar\delta^\top T(x)<0}(p_*+q)\frac{1}{\frac{p}{q}+\frac{q}{p}}(\boldsymbol{v}^\top T(x))^2dx \le \int_{\bar\delta^\top T(x)<0}(p_*+q)\cdot\min\left\{\frac{q}{p},\frac{p}{q}\right\}(\boldsymbol{v}^\top T(x))^2dx\\
&= \int_{\bar\delta^\top T(x)<0}(p_*+q)\cdot\min\left\{\frac{q}{p_*\exp\left(\bar\delta^\top T(x)\right)},\ \frac{p_*\exp(\bar\delta^\top T(x))}{q}\right\}(\boldsymbol{v}^\top T(x))^2dx\\
&= \int_{\bar\delta^\top T(x)<0}\exp\left(-\bar\delta^\top T(x)\right)(p_*+q)\min\left\{\frac{q}{p_*},\ \frac{p_*\exp(2\bar\delta^\top T(x))}{q}\right\}(\boldsymbol{v}^\top T(x))^2dx\\
&\overset{(i)}{\le} \int_{\bar\delta^\top T(x)<0}\exp\left(-\bar\delta^\top T(x)\right)(p_*+q)\min\left\{\frac{q}{p_*},\frac{p_*}{q}\right\}(\boldsymbol{v}^\top T(x))^2dx\\
&\le 2\int_{\bar\delta^\top T(x)<0}\exp\left(-\bar\delta^\top T(x)\right)\min\{q,p_*\}(\boldsymbol{v}^\top T(x))^2dx\\
&\overset{(ii)}{\le} 2\int_x \exp\left(-\bar\delta^\top T(x)\right)\min\{q,p_*\}(\boldsymbol{v}^\top T(x))^2dx \le 2\int_x\exp\left(-\bar\delta^\top T(x)\right)\sqrt{p_*q}(\boldsymbol{v}^\top T(x))^2dx\\
&=2\frac{Z(\frac{\theta_*+\theta_q}{2}-\bar\theta)\exp(-\bar\alpha)}{\sqrt{Z(\theta_*)Z(\theta_q)}}\int_x p_{\frac{\theta_*+\theta_q}{2}-\bar\theta}\cdot(\boldsymbol{v}^\top T(x))^2dx \le 2\frac{Z(\frac{\theta_*+\theta_q}{2}-\bar\theta)\exp(-\bar\alpha)}{\sqrt{Z(\theta_*)Z(\theta_q)}}\mathbb{E}_{\frac{\theta_*+\theta_q}{2}-\bar\theta}(\boldsymbol{v}^\top T(x))^2\\
&\overset{(iii)}{\le} 2\underbrace{\frac{Z(\frac{\theta_*+\theta_q}{2})}{\sqrt{Z(\theta_*)Z(\theta_q)}}}_{:=1/B}\cdot\exp\left(\beta_Z\bar\theta-\bar\alpha\right)\cdot\mathbb{E}_{\frac{\theta_*+\theta_q}{2}-\bar\theta}(\boldsymbol{v}^\top T(x))^2\\
&\overset{(iv)}{\le}\frac{2}{B}\cdot\exp\left(1+\frac{1}{\beta_Z}\right)\cdot\mathbb{E}_{\frac{\theta_*+\theta_q}{2}-\bar\theta}(\boldsymbol{v}^\top T(x))^2,
\end{aligned}
\tag{4.76}
$$

where step (i) is because $\bar\delta^\top T(x) < 0$; step (ii) increases the value by integrating over all x; step (iii) uses Assumption 12 on Lipschitz log partition function; and step (iv) follows from the choice of $\bar\delta = [\bar\theta,\bar\alpha]$ that $\|\bar\delta\| \le \frac{1}{\beta_Z}$.

The second term can be bounded as:

$$
\begin{aligned}
&\int_{\bar\delta^\top T(x)>0}(p_*+q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2dx \le \int_{\bar\delta^\top T(x)>0}\frac{p_*+q}{p+q}\min\{p,q\}(\boldsymbol{v}^\top T(x))^2dx\\
&\le\int_{\bar\delta^\top T(x)>0}\min\{p,q\}(\boldsymbol{v}^\top T(x))^2dx \le \int_x\sqrt{pq}(\boldsymbol{v}^\top T(x))^2dx = \frac{Z(\frac{\theta_*+\bar\theta+\theta_q}{2})\exp(-\bar\alpha)}{\sqrt{Z(\theta_*+\bar\theta)Z(\theta_q)}}\mathbb{E}_{\frac{\theta_*+\bar\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\\
&\overset{(i)}{\le}\frac{Z(\frac{\theta_*+\theta_q}{2})}{\sqrt{Z(\theta_*)Z(\theta_q)}}\mathbb{E}_{\frac{\theta_*+\bar\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\cdot\exp\left(\frac{3}{2}\beta_Z\|\bar\theta\|_2-\bar\alpha\right)\le\frac{1}{B}\exp\left(\frac{3}{2}+\frac{1}{\beta_Z}\right)\cdot\mathbb{E}_{\frac{\theta_*+\bar\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2.
\end{aligned}
\tag{4.77}
$$

where step (i) uses Assumption 12 about Lipschitzness of the log partition function, and step (ii) is because we have chosen that $\|\bar\delta\|_2 \le \frac{1}{\beta_Z}$.

Substituting back to equation 4.75 gives:

$$\begin{aligned}\boldsymbol{v}^\top \nabla^2 L(\tau)\boldsymbol{v} \leq& \frac{1}{B}\left[2\exp\left(1+\frac{1}{\beta_Z}\right)\cdot \mathbb{E}_{\frac{\theta_*+\theta_q}{2}-\bar{\theta}}(\boldsymbol{v}^\top T(x))^2 + \exp\left(\frac{3}{2}+\frac{1}{\beta_Z}\right)\cdot \mathbb{E}_{\frac{\theta_*+\theta_q+\bar{\theta}}{2}}(\boldsymbol{v}^\top T(x))^2\right]\\ \leq& \frac{2\exp(\frac{3}{2}+\frac{1}{\beta_Z})}{B}\cdot \min\left\{\lambda_{\max},\ \sigma_{\max}(\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[T(x)T(x)^\top]) + \gamma_{\max}\|\bar{\delta}\|\right\},\end{aligned} \tag{4.78}$$

where the second inequality uses Assumption 13 and Assumption 14 for the first and second term respectively.

Recall that $\boldsymbol{v}^\top \nabla^2 L(\tau_*)\boldsymbol{v} \geq \frac{1}{4B^2}\frac{1}{\lambda_{\max}}\left(\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\right)^2$. Hence:

$$\begin{aligned}&\frac{\sigma_{\max}(\nabla^2 L(\tau))}{\sigma_{\max}(\nabla^2 L(\tau_*))} = \frac{\max_{\boldsymbol{v}} \boldsymbol{v}^\top \nabla^2 L(\tau)\boldsymbol{v}}{\max_{\tilde{\boldsymbol{v}}'} \tilde{\boldsymbol{v}}^\top \nabla^2 L(\tau_*)\tilde{\boldsymbol{v}}}\\ &\leq 8\lambda_{\max} B\exp\left(\frac{3}{2}+\frac{1}{\beta_Z}\right)\frac{\mathbb{E}_{\frac{\theta_*+\theta_q}{2}-\bar{\theta}}(\boldsymbol{v}^\top T(x))^2 + \mathbb{E}_{\frac{\theta_*+\theta_q+\bar{\theta}}{2}}(\boldsymbol{v}^\top T(x))^2}{\max_{\tilde{\boldsymbol{v}}}\left(\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}(\tilde{\boldsymbol{v}}^\top T(x))^2\right)^2}\\ &\leq 8\frac{\lambda_{\max}}{\lambda_{\min}} B\exp\left(\frac{3}{2}+\frac{1}{\beta_Z}\right)\cdot\min\left\{\frac{2\lambda_{\max}}{\lambda_{\min}}, 2+\frac{\gamma_{\max}\|\bar{\delta}\|}{\sigma_{\max}(\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[T(x)T(x)^\top])}\right\}\\ &\leq 8\frac{\lambda_{\max}}{\lambda_{\min}} B\exp\left(\frac{3}{2}+\frac{1}{\beta_Z}\right)\cdot\min\left\{\frac{2\lambda_{\max}}{\lambda_{\min}}, 2+\frac{\gamma_{\max}\|\bar{\delta}\|}{\lambda_{\min}}\right\}.\end{aligned} \tag{4.79}$$

Lower bounding $\frac{\sigma_{\min}(\nabla^2 L(\tau))}{\sigma_{\min}(\nabla^2 L(\tau_*))}$: Let us denote $\mathcal{S}_1 := \{x : \bar{\delta}^\top T(x) > 0\}$ and $\mathcal{S}_{-1} := \{x : \bar{\delta}^\top T(x) \leq 0\}$. The goal is to lower bound:

$$\begin{aligned}\boldsymbol{v}^\top \nabla^2 L(\tau)\boldsymbol{v} =& \int_{x\in\mathcal{S}_1}(p_*+q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2dx + \int_{x\in\mathcal{S}_{-1}}(p_*+q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2dx\\ :=& T_1 + T_{-1}.\end{aligned} \tag{4.80}$$

Let's lower bound T_1, T_{-1} in each of the following two cases.

The first case is when $T_{-1} \geq T_1$. Let $\mathcal{S}_{\frac{1}{2}}(\boldsymbol{v}) \subset \mathcal{S}_{-1}$ denote a set s.t.

$$\int_{x\in\mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})}\min\{p,q\}(\boldsymbol{v}^\top T(x))^2 \geq \frac{1}{2}\int_x \min\{p,q\}(\boldsymbol{v}^\top T(x))^2.$$

Write $\bar{\delta} = [\bar{\theta}, \bar{\alpha}]$ as before, then

$$\begin{aligned}T_1 \geq& 0,\\ T_{-1} =& \int_{\bar{\delta}^\top T(x)<0}(p_*+q)\frac{pq}{(p+q)^2}(\boldsymbol{v}^\top T(x))^2dx \overset{(i)}{\geq} \int_{\bar{\delta}^\top T(x)<0}\frac{pq}{p+q}(\boldsymbol{v}^\top T(x))^2dx\\ \geq& \frac{1}{2}\int_{\bar{\delta}^\top T(x)<0}\min\{p,q\}(\boldsymbol{v}^\top T(x))^2dx \overset{(ii)}{\geq} \frac{1}{2}\int_{\mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})}\min\{p,q\}(\boldsymbol{v}^\top T(x))^2dx\\ \overset{(iii)}{\geq}& \frac{1}{4}\int \min\{p,q\}(\boldsymbol{v}^\top T(x))^2dx \overset{(iv)}{\geq} \frac{\exp(-\bar{\alpha})}{8\lambda_{\max}}\cdot\frac{Z(\frac{\theta+\theta_q}{2})^2}{Z(\theta)Z(\theta_q)}\left(\mathbb{E}_{\frac{\theta+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2\right)^2\\ \overset{(v)}{\geq}& \frac{\exp(-\bar{\alpha})}{8\lambda_{\max}}\cdot\frac{1}{B^2}\exp(-2\beta_Z\|\bar{\delta}\|)\cdot\left(\mathbb{E}_{\frac{\theta_*+\theta_q+\bar{\theta}}{2}}(\boldsymbol{v}^\top T(x))^2\right)^2,\end{aligned} \tag{4.81}$$

where step (i) uses $\frac{p_*+q}{p+q} < 1$ since $\bar{\delta}^\top T(x) < 0$; step $(ii), (iii)$ follows from the definition of $\mathcal{S}_{-1}$; step (iv) uses equation 4.71; and step (v) uses Assumption 12 that the log partition function is Lipschitz.

The second case is when $T_1 \geq T_{-1}$. Let $\mathcal{S}_{\frac{1}{2}}(\boldsymbol{v}) \subset \mathcal{S}_1$ denote a set s.t.

$$\int_{x \in \mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})} \min\{p, q\}(\boldsymbol{v}^\top T(x))^2 \geq \frac{1}{2} \int_x \min\{p, q\}(\boldsymbol{v}^\top T(x))^2.$$

Then T_{-1}, T_1 can be lower bounded as:

$$\begin{aligned}
T_{-1} \geq & 0 \\
T_1 = & \int_{x \in \mathcal{S}_1} (p_* + q) \frac{pq}{(p+q)^2} (\boldsymbol{v}^\top T(x))^2 dx \\
\geq & \frac{1}{2} \int_{x \in \mathcal{S}_1} \frac{p_* + q}{p + q} \cdot \min\{p, q\}(\boldsymbol{v}^\top T(x))^2 dx \geq \frac{1}{2} \int_{x \in \mathcal{S}_1} \frac{p_*}{p} \cdot \min\{p, q\}(\boldsymbol{v}^\top T(x))^2 dx \\
= & \frac{\exp(\bar{\alpha})}{2} \int_{x \in \mathcal{S}_1} \min\{p_{\theta_*}, p_{\theta_q - \bar{\theta}}\}(\boldsymbol{v}^\top T(x))^2 dx \\
\geq & \frac{\exp(\bar{\alpha})}{2} \int_{x \in \mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})} \min\{p_{\theta_*}, p_{\theta_q - \bar{\theta}}\}(\boldsymbol{v}^\top T(x))^2 dx \geq \frac{\exp(\bar{\alpha})}{4} \int_x \min\{p_{\theta_*}, p_{\theta_q - \bar{\theta}}\}(\boldsymbol{v}^\top T(x))^2 dx \\
\geq & \frac{\exp(\bar{\alpha})}{8\lambda_{\max}} \cdot \frac{Z(\frac{\theta_* + \theta_q - \bar{\theta}}{2})^2}{Z(\theta_*) Z(\theta_q - \bar{\theta})} \left(\mathbb{E}_{\frac{\theta_* + \theta_q - \bar{\theta}}{2}} (\boldsymbol{v}^\top T(x))^2 \right)^2 \\
\geq & \frac{\exp(\bar{\alpha})}{8\lambda_{\max}} \cdot \frac{1}{B^2} \exp(-2\beta_Z \|\bar{\delta}\|) \cdot \left(\mathbb{E}_{\frac{\theta_* + \theta_q - \bar{\theta}}{2}} (\boldsymbol{v}^\top T(x))^2 \right)^2.
\end{aligned} \tag{4.82}$$

Combining both cases and using $\|\bar{\delta}\| \leq \frac{1}{\beta_Z}$, we get:

$$\begin{aligned}
& \boldsymbol{v}^\top \nabla^2 L(\tau) \boldsymbol{v} = T_1 + T_{-1} \\
\geq & \frac{\exp(-2 - \frac{1}{\beta_Z})}{8\lambda_{\max}} \cdot \frac{1}{B^2} \cdot \min \left\{ \left(\mathbb{E}_{\frac{\theta_* + \theta_q + \bar{\theta}}{2}} (\boldsymbol{v}^\top T(x))^2 \right)^2, \left(\mathbb{E}_{\frac{\theta_* + \theta_q - \bar{\theta}}{2}} (\boldsymbol{v}^\top T(x))^2 \right)^2 \right\}.
\end{aligned} \tag{4.83}$$

Recall that $\boldsymbol{v}^\top \nabla^2 L(\tau_*) \boldsymbol{v} \leq \frac{2}{B} \mathbb{E}_{\frac{\theta_* + \theta_q}{2}} [(\boldsymbol{v}^\top T(x))^2]$. Hence

$$\begin{aligned}
& \frac{\sigma_{\min}(\nabla^2 L(\tau))}{\sigma_{\min}(\nabla^2 L(\tau_*))} = \frac{\min_{\boldsymbol{v}} \boldsymbol{v}^\top \nabla^2 L(\tau) \boldsymbol{v}}{\min_{\tilde{\boldsymbol{v}}'} \tilde{\boldsymbol{v}}^\top \nabla^2 L(\tau_*) \tilde{\boldsymbol{v}}} \\
\geq & \frac{16 \exp(-2 - \frac{1}{\beta_Z})}{B} \frac{\lambda_{\min}}{\lambda_{\max}} \cdot \max \left\{ \frac{\lambda_{\min}}{\lambda_{\max}}, \min \left\{ \frac{\sigma_{\min}(\mathbb{E}_{\frac{\theta_* + \theta_q + \bar{\theta}}{2}} TT^\top)}{\sigma_{\min}(\mathbb{E}_{\frac{\theta_* + \theta_q}{2}} [TT^\top])}, \frac{\sigma_{\min}(\mathbb{E}_{\frac{\theta_* + \theta_q - \bar{\theta}}{2}} TT^\top)}{\sigma_{\min}(\mathbb{E}_{\frac{\theta_* + \theta_q}{2}} [TT^\top])} \right\} \right\} \\
\geq & \frac{16 \exp(-2 - \frac{1}{\beta_Z})}{B} \frac{\lambda_{\min}}{\lambda_{\max}} \cdot \max \left\{ \frac{\lambda_{\min}}{\lambda_{\max}}, 1 - \frac{\gamma_{\min} \|\bar{\delta}\|}{\lambda_{\min}} \right\}.
\end{aligned} \tag{4.84}$$

□

4.1.11 Proofs: eNCE satisfies Assumption 15 (Lemma 54)

Lemma 62 (Lemma 54, restated). *Under Assumption 12, 13 with constant β_Z, $\lambda_{\max}$ and $\lambda_{\min}$, for any unit vector $\boldsymbol{u}$ and constant $c \in [0, \frac{1}{\beta_Z}]$, the maximum and minimum singular values of $\boldsymbol{H}(\tau_* + c\boldsymbol{u})$ satisfy Assumption 15 with constants $\beta_u = 2e \cdot \frac{\lambda_{\max}}{\lambda_{\min}}$, $\beta_l = \frac{1}{2e} \cdot \frac{\lambda_{\min}}{\lambda_{\max}}$.*

Proof. We directly calculate the Hessian at some $\tilde{\tau} := \tau_* + c\boldsymbol{u}$ for some $c \leq \frac{1}{\beta_Z}$ and $\|\boldsymbol{u}\|_2 = 1$, using the expression in equation 4.12:

$$\begin{aligned}
&\nabla^2 L_{\exp}(\tilde{\tau}) = \int_x \left(p_* \sqrt{\frac{q}{\tilde{p}}} + q\sqrt{\frac{\tilde{p}}{q}} \right) T(x)T(x)^\top \\
&= \int_x \left[\exp\left(\langle \tau_* + \frac{\tau_q - \tilde{\tau}}{2}, T(x) \rangle \right) + \exp\left(\langle \frac{\tau_q + \tilde{\tau}}{2}, T(x) \rangle \right) \right] T(x)T(x)^\top \\
&= \int_x \left[\exp\left(\langle \frac{\tau_q + \tau_*}{2} - \frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) + \exp\left(\langle \frac{\tau_q + \tau_*}{2} + \frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) \right] T(x)T(x)^\top \\
&= \int_x \left[\exp\left(\langle -\frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) + \exp\left(\langle \frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) \right] \exp\left(\langle \frac{\tau_q + \tau_*}{2}, T(x) \rangle \right) T(x)T(x)^\top \\
&= \underbrace{\frac{Z(\frac{\theta_* + \theta_q}{2})}{\sqrt{Z(\theta_q)Z(\theta_*)}}}_{B(P_*, Q)} \int_x \left[\exp\left(\langle -\frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) + \exp\left(\langle \frac{c}{2}\boldsymbol{u}, T(x) \rangle \right) \right] \exp\left(\langle \tau(\frac{\theta_q + \theta_*}{2}), T(x) \rangle \right) T(x)T(x)^\top.
\end{aligned} \tag{4.85}$$

Note that without the term in the square brackets, the integration is exactly the same as the one for $\boldsymbol{H}_*$.

We would like to bound the ratio $\frac{\boldsymbol{v}^\top \nabla^2 L_{\exp}(\tilde{\tau})\boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{H}_* \boldsymbol{v}}$ for any unit vector $\boldsymbol{v}$. Denote $\bar{\delta} := \frac{c\boldsymbol{u}}{2}$, $\bar{\tau} := \tau\left(\frac{\theta_q + \theta_*}{2}\right)$ for notation convenience, and denote $\mathcal{S}_1 := \{x : \bar{\delta}^\top T(x) > 0\}$, $\mathcal{S}_{-1} := \{x : \bar{\delta}^\top T(x) \leq 0\}$. We have:

$$\begin{aligned}
\frac{\boldsymbol{v}^\top \nabla^2 L_{\exp}(\tilde{\tau})\boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{H}_* \boldsymbol{v}} \simeq & \frac{\int_{x \in \mathcal{S}_1} \exp\left(\bar{\delta}^\top T(x)\right) \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2}{\int_x \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2} \\
& + \frac{\int_{x \in \mathcal{S}_{-1}} \exp\left(\bar{\delta}^\top T(x)\right) \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2}{\int_x \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2} := T_1 + T_{-1}.
\end{aligned} \tag{4.86}$$

Recall that $f \simeq g$ means functions f, g differ only by a constant factor. This equation will be used to calculate both the upper and the lower bound.

For the upper bound, let $\chi \in \{\pm 1\}$, we have

$$\begin{aligned}
T_\chi =& \frac{\int_{x:\chi\bar{\delta}^\top T(x) > 0} \exp\left(\chi\bar{\delta}^\top T(x)\right) \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2}{\int_x \exp\left(\bar{\tau}^\top T(x)\right) (\boldsymbol{v}^\top T(x))^2} \\
=& \frac{Z(\chi\bar{\theta} + \frac{\theta_q + \theta_*}{2})}{Z(\frac{\theta_q + \theta_*}{2}) \cdot \exp(\chi\bar{\alpha})} \cdot \frac{\int_{x:\bar{\delta}^\top T(x) > 0} p_{\chi\bar{\theta} + \frac{\theta_* + \theta_q}{2}}(x) (\boldsymbol{v}^\top T(x))^2}{\int_x p_{\frac{\theta_* + \theta_q}{2}} (\boldsymbol{v}^\top T(x))^2} \\
\leq & \frac{Z(\chi\bar{\theta} + \frac{\theta_q + \theta_*}{2})}{Z(\frac{\theta_q + \theta_*}{2}) \cdot \exp(\chi\bar{\alpha})} \cdot \frac{\mathbb{E}_{\chi\bar{\theta} + \frac{\theta_* + \theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_* + \theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]} \overset{(i)}{\leq} \exp\left(\beta_Z \|\bar{\theta}\| - \chi\bar{\alpha}\right) \frac{\mathbb{E}_{\chi\bar{\theta} + \frac{\theta_* + \theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_* + \theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]},
\end{aligned} \tag{4.87}$$

where step (i) uses the Lipschitz property of the log partition function in assumption 12.

For the lower bound, let $\chi^* := \arg\max_{\chi \in \{\pm 1\}} T_\chi$. Write $\bar{\delta} = [\bar{\theta}, \bar{\alpha}]$ (i.e. separating out $\bar{\alpha}$ which is the normalizing constant), let $\mathcal{S}_{\frac{1}{2}}(\boldsymbol{v}) \subset \mathcal{S}_{\chi^*}$ denote a set s.t.

$$\int_{x \in \mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})} p_{\chi^*\bar{\theta} + \frac{\theta_* + \theta_q}{2}}(x) (\boldsymbol{v}^\top T(x))^2 \geq \frac{1}{2} \int_x p_{\chi^*\bar{\theta} + \frac{\theta_* + \theta_q}{2}}(x) (\boldsymbol{v}^\top T(x))^2.$$

Then T_χ for $\chi \in \{\pm 1\}$ can be lower bounded as:

$$
\begin{aligned}
T_{\chi^*} \geq & \frac{Z(\chi\bar{\theta} + \frac{\theta_q+\theta_*}{2})}{Z(\frac{\theta_q+\theta_*}{2}) \cdot \exp(\bar{\alpha})} \cdot \frac{\int_{x \in \mathcal{S}_{\frac{1}{2}}(\boldsymbol{v})} p_{\chi^*\bar{\theta}+\frac{\theta_*+\theta_q}{2}}(x)(\boldsymbol{v}^\top T(x))^2}{\int_x p_{\frac{\theta_*+\theta_q}{2}}(\boldsymbol{v}^\top T(x))^2} \\
\geq & \frac{1}{2} \frac{Z(\chi^*\bar{\theta} + \frac{\theta_q+\theta_*}{2})}{Z(\frac{\theta_q+\theta_*}{2}) \cdot \exp(\bar{\alpha})} \cdot \frac{\mathbb{E}_{\chi^*\bar{\theta}+\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]} \\
\overset{(i)}{\geq} & \frac{1}{2} \exp\left(-\beta_Z \|\bar{\theta}\| - \chi^*\bar{\alpha}\right) \cdot \frac{\mathbb{E}_{\chi^*\bar{\theta}+\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}, \\
T_{-\chi^*} \geq & 0,
\end{aligned}
\tag{4.88}
$$

where step (i) uses the Lipschitz property of the log partition function in assumption 12.

This means for any unit vector $\boldsymbol{v}$, we have

$$
\begin{aligned}
\frac{\boldsymbol{v}^\top \nabla^2 L_{\exp}(\tilde{\tau})\boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{H}_* \boldsymbol{v}} =& T_1 + T_{-1} \leq \exp\left(\beta_Z\|\bar{\theta}\| + |\bar{\alpha}|\right) \cdot \left[\frac{\mathbb{E}_{\bar{\theta}+\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]} + \frac{\mathbb{E}_{-\bar{\theta}+\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}{\mathbb{E}_{\frac{\theta_*+\theta_q}{2}}[(\boldsymbol{v}^\top T(x))^2]}\right] \\
\overset{(i)}{\leq}& 2\exp\left(\beta_Z\|\bar{\theta}\| + |\bar{\alpha}|\right) \cdot \frac{\lambda_{\max}}{\lambda_{\min}} \leq 2\exp\left(\frac{c}{2}(1+\beta_Z)\right) \cdot \frac{\lambda_{\max}}{\lambda_{\min}} \leq 2e \cdot \frac{\lambda_{\max}}{\lambda_{\min}}, \\
\frac{\boldsymbol{v}^\top \nabla^2 L_{\exp}(\tilde{\tau})\boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{H}_* \boldsymbol{v}} =& T_1 + T_{-1} \overset{(ii)}{\geq} \frac{1}{2}\exp\left(-\beta_Z\|\bar{\theta}\| - |\bar{\alpha}|\right) \cdot \frac{\lambda_{\min}}{\lambda_{\max}} \geq \frac{1}{2e} \cdot \frac{\lambda_{\min}}{\lambda_{\max}},
\end{aligned}
\tag{4.89}
$$

where step $(i), (ii)$ follow from Assumption 13.

Hence the eNCE loss satisfies Assumption 15 with constants $\beta_u = 2e \cdot \frac{\lambda_{\max}}{\lambda_{\min}}$, $\beta_l = \frac{1}{2e} \cdot \frac{\lambda_{\min}}{\lambda_{\max}}$.

□

4.1.12 Conclusion and Discussions

We provided a theoretical analysis of the algorithmic difficulties that arise when optimizing the NCE objective with an uninformative noise distribution, stemming from an ill-behaved loss landscape. Our theoretical results are inspired by empirical observations in prior works [Rhodes et al., 2020, Gao et al., 2020, Goodfellow et al., 2014] and provide the first formal explanation on the nature of the optimization problems of NCE. Our negative results showed that even on the simple task of Gaussian mean estimation, and even assuming access to the population gradient, gradient descent and Newton's method with standard step size choice still require an exponential number of steps to reach a good solution.

We then proposed modifications to the NCE loss and optimization algorithm, whose combination results in the first provably polynomial convergence rate for NCE. The loss we propose, eNCE, can be efficiently optimized using normalized gradient descent and empirically outperforms existing methods. We hope these theoretical results will help identify promising new directions in the search for simple, effective, and practical improvements to noise-contrastive estimation.

4.2 Accelerating feature learning using progressive knowledge distillation

As the cost of training state-of-the-art models grows rapidly [Hoffmann et al., 2022], there is increased interest in using knowledge distillation [Hinton et al., 2015] to leverage existing capable models to train new models more efficiently and effectively. Knowledge distillation is an effective technique to train smaller vision [Jia et al., 2021, Touvron et al., 2021, Yu et al., 2022, Lin et al., 2023] and language models [Sanh et al., 2019, Gunasekar et al., 2023, Touvron et al., 2023, Reid et al., 2024] that permit faster inference with comparable performance. However, one curiously persistent phenomenon is that a better teacher does not always yield a stronger student. Prior works [Mirzadeh et al., 2019, Jin et al., 2019, Jafari et al., 2021, Harutyunyan et al., 2022, Anil et al., 2018] hypothesized that this is due to a capability gap between the teacher and the student. As such, they proposed *progressive distillation*, where the student is incrementally supervised by increasingly capable teachers. This technique has yielded strong empirical performance. One recent example is the training of Gemini-1.5 Flash from Gemini-1.5 Pro [Reid et al., 2024, Team et al., 2024]: Gemini-1.5 Flash achieves 95% of Gemini-1.5 Pro's performance on average and outperforms Gemini-1.0 Pro on 41 out of 50 text-based long-context benchmarks, while being substantially smaller. However, little is understood about progressive distillation in terms of the optimization or generalization benefits, compared to directly learning from the data or the final teacher checkpoint (i.e., *one-shot distillation*).

Most prior work hypothesizes that progressive distillation enables better generalization [Mirzadeh et al., 2019, Jafari et al., 2021, Harutyunyan et al., 2022]. In contrast, we identify a novel mechanism by which progressive distillation helps a student by *accelerating its optimization* (Figure 4.4). We define optimization acceleration as achieving improved performance with fewer training steps or samples. In this paper, we use fresh training samples in each training step; hence we use training steps and samples interchangeably to measure the optimization speed.

We study two tasks where learning the right features is believed to be important and show that the intermediate checkpoints provide signal towards these features. The first is learning sparse parity (Definition 23), which is a commonly studied setting to understand the feature learning dynamics of neural networks. The second is learning probabilistic context-free grammars (PCFGs), which we use as a sandbox for capturing certain aspects of language modeling. Theory and extensive experiments in these settings support the following claims.

1. **Progressive distillation accelerates student learning.** Our experiments in multiple settings demonstrate that progressive distillation accelerates training compared to standard one-shot distillation and learning from the data directly (Figure 4.4). More specifically, for sparse parity, progressive distillation can train a smaller MLP (or Transformer) at the same speed as a larger MLP (or Transformer). For PCFGs, progressive distillation improves the accuracy of a smaller BERT model [Devlin et al., 2018b] at masked prediction. Finally, we verify our findings on more realistic setups of training BERT on Wikipedia and Books dataset.
2. **An implicit curriculum drives faster learning.** We demonstrate theoretically and empirically that acceleration comes from an *implicit curriculum* of easy-to-learn subtasks provided by intermediate teacher checkpoints, which is not available from the final teacher checkpoint. For sparse parity, the easy-to-learn subtasks provide supervision for the coordinates which constitute the support

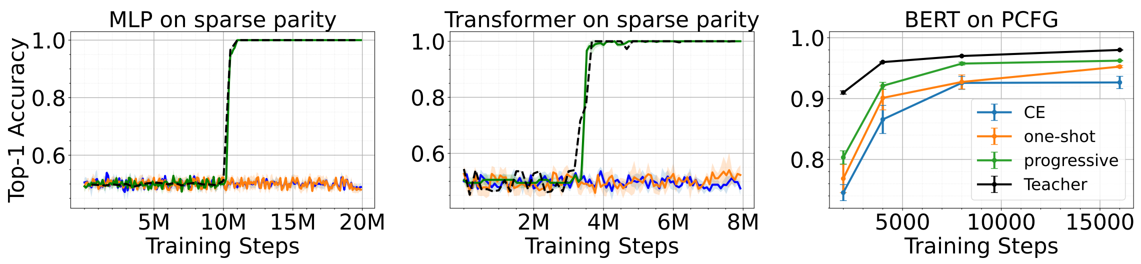


Figure 4.4: **Progressive distillation accelerates training.** *Left*: MLP on $(100, 6)$-sparse parity (Definition 23), with width-50k teachers and width-100 students. Progressive distillation checkpoints are at 100k-step intervals, and one-shot checkpoint uses the final (20M-step) checkpoint. *Middle*: Transformer on $(100, 6)$-sparse parity, with 32-head teachers and 4-head students. Progressive distillation checkpoints are at 10k-step intervals, and the one-shot checkpoint is at 250k steps. *Right*: Transformers on PCFG (Section 4.2.3), with 32-head teachers and 8-head students using BERT-style masked prediction. Progressive distillation uses 8 intermediate checkpoints.

of the sparse parity (Section 4.2.2). As a consequence, we show progressive distillation provably improves the sample complexity for sparse parity over one-shot distillation or learning directly from data (Theorem 22). For PCFGs, the implicit curriculum is defined in terms of learning features that increasingly capture larger n-gram contexts. Our results also provide guidance on how to select the intermediate teachers used during progressive distillation.

Related works[9].

One persistent surprise in knowledge distillation is that stronger teachers do not always lead to stronger students. Prior works have speculated that an overly large "teacher-student gap" is the cause, and accordingly proposed to bridge this gap by introducing supervision of intermediate difficulty [Mirzadeh et al., 2019, Cho and Hariharan, 2019, Harutyunyan et al., 2022, Jafari et al., 2021]. Mirzadeh et al. [2019] used multi-step distillation involving models of intermediate sizes, and Shi et al. [2021] proposed to directly inject teacher supervision into the student's trajectory using an approximation of mirror descent. Most related to our work, Harutyunyan et al. [2022] analyzed distillation for extremely wide networks and found it helpful to learn from the intermediate checkpoints of the teacher, a strategy also adopted by Jin et al. [2019]. They speculated that this is because neural networks learn progressively complex functions during training [Kalimeris et al., 2019]. In contrast to their focus on the generalization ability of the student, we study the optimization dynamics of distillation.

It is worth noting that there is also a rich body of work on understanding standard (one-shot) distillation, mostly regarding regularization effects. In particular, Menon et al. [2021] shows that learning from the teacher leads to a tighter generalization bound when the teacher is closer to the Bayes distribution over the class labels. However, such Bayes perspective cannot explain the training acceleration in the feature learning tasks considered in this work, whose the Bayes distributions are delta masses and hence are the same as the one-hot labels themselves. Our results fill this gap by providing an orthogonal view of implicit curriculum.

[9]We defer a detailed discussion of related work to Section 4.2.5.1.

The benefit of curriculum on sparse parity has also been explored in Abbe et al. [2024], where the curriculum also helps identify the support. The difference though is that their curriculum is defined by explicitly altering the distribution over the inputs, whereas our curriculum shows up implicitly in the teacher supervision. Moreover, our implicit curriculum emphasizes that a properly chosen intermediate checkpoint, while having a worse accuracy than the final checkpoint, can lead to a better-performing student. This can be seen as a plausible mechanism for *weak-to-strong generalization* [Burns et al., 2023].

Outline. Section 4.2.1 describes the distillation strategies. Section 4.2.2 introduces the implicit curriculum with a case study on sparse parity, presenting both empirical evidence and a provable benefit in sample complexity. Section 4.2.3 continues the empirical investigations on PCFG, and extends the observations to BERT's training on Wikipedia and Books dataset. Finally, Section 4.2.4 discusses open directions.

4.2.1 Preliminaries

We now outline the distillation strategies considered in this paper and their empirical instantiation. For ease of exposition, we discuss one-dimensional label classification tasks here and generalize to sequence-to-sequence functions in Section 4.2.3. Denote the teacher and student models operating on input domain $\mathcal{X}$ as $f_{\mathcal{T}} : \mathcal{X} \to \mathbb{R}^C$ and $f_{\mathcal{S}} : \mathcal{X} \to \mathbb{R}^C$, respectively. The outputs of a model f are logits that are transformed into a probability distribution over C classes using a softmax function with temperature τ, denoted as $p(x;\tau) := \text{softmax}(f(x)/\tau)$. We will use $p_{\mathcal{T}}$, $p_{\mathcal{S}}$ to denote the probability distributions of the teacher and the student, and will omit the subscript to denote a generic model. When $\tau = 1$, we omit τ from the notation for brevity. Following Zheng and Yang [2024], we set $\tau = 1$ for the student and vary the temperature of the teacher.

We compare two loss functions: ℓ, where the student $f_{\mathcal{S}}$ learns only from ground-truth labels , and ℓ_{D} , where the student $f_{\mathcal{S}}$ is supervised only with the logits of some teacher $f_{\mathcal{T}}$.[10]

$$\ell(x, y; f_{\mathcal{S}}) = \text{KL}(\boldsymbol{e}_y \| p_{\mathcal{S}}(x)), \tag{4.90}$$

$$\ell_{\text{D}}(x; f_{\mathcal{S}}, f_{\mathcal{T}}) = \text{KL}(p_{\mathcal{T}}(x;\tau) \| p_{\mathcal{S}}(x)), \tag{4.91}$$

where $\boldsymbol{e}_y$ is a one-hot vector whose y^{th} entry is 1. We consider two strategies for choosing the teacher. The first is *one-shot distillation*, where the student learns from a fixed $f_{\mathcal{T}}$ throughout the training, and the teacher is chosen as the final converged checkpoint. The second is *progressive distillation*, where the student learns from multiple intermediate checkpoints of the teacher's training run:

Definition 21 (($C_{\mathcal{T}}, \mathcal{D}$)-progressive distillation)**.** *Given a set of teacher checkpoints $C_{\mathcal{T}} = \{f_{\mathcal{T}_i}\}$ and a set of training durations $\mathcal{D}$, the student is trained with the logits of teacher checkpoint $f_{\mathcal{T}_i}$ for training length $\mathcal{D}_i$ with $i \in [|C_{\mathcal{T}}|] := \{1, \cdots, |C_{\mathcal{T}}|\}$.*

To simplify the presentation, the main paper tests a specific type of progressive distillation schemes, where $C_{\mathcal{T}}$ contains N equally-spaced checkpoints and the student is trained on each one for T steps:

[10]We note that prior papers generally use a combination of these objectives, but we use supervision from one source in order to isolate its effects on distillation.

Definition 22 ((N, T)-progressive distillation). *$C_{\mathcal{T}}$ contains $N-1$ equally-spaced intermediate teacher checkpoints and the final teacher checkpoint. The student is trained with each checkpoint for T training steps. After NT steps, the student is trained with the final teacher checkpoint.*

To study the effect of each teacher checkpoint, we will also consider an extreme version of progressive distillation with $N = 2$, where the student uses one intermediate teacher checkpoint.

Choice of temperature. We set $\tau = 10^{-4}$ for sparse parity and PCFG experiments (Section 4.2.3) where the vocabulary size is smaller than 5, and $\tau = 10^{-20}$ for natural language experiments (Section 4.2.3.2) whose vocabulary size is 30k.[11] Using such a small temperature makes the teacher's outputs close to one-hot labels. This removes potential regularization effects due to the softness of the labels [Yuan et al., 2020] which would otherwise be a confounding factor. Moreover, the supervision with nearly one-hot labels is more representative of the setting where the student learns directly from the teacher's *generations* instead of the logits. This method, often described as generating synthetic data in the language modeling setting, has generally yielded small yet highly performant students [Gunasekar et al., 2023, Liu et al., 2024]. For one-shot distillation, we report the best-performing temperature among $\tau = 1, 10^{-4}$ in the main paper and defer other results to Section 4.2.8.7.

4.2.2 The implicit curriculum: a case study with sparse parity

To elucidate the mechanism by which distillation accelerates training, we first focus on the well-studied task of learning sparse parity.[12] Sparse parity is a commonly used sandbox for understanding neural network optimization in the presence of feature learning [Barak et al., 2022b, Bhattamishra et al., 2022, Morwani et al., 2024, Edelman et al., 2023, Abbe et al., 2024].

Definition 23 ((d, k)-sparse parity task). *Let $S \subset [d]$ denote a fixed set of coordinates, with $|S| = k$ and $k < d$. Then, the sparse parity task is defined for any input $\mathbf{x} \in \{\pm 1\}^d$, whose label is computed as $y = 1$ if $\prod_{i \in S} \mathbf{x}_i > 0$ and 2 otherwise.*

We train the teacher and student models using 2-label classification, where $f_{\mathcal{T}}$ and $f_{\mathcal{S}}$ return logits in $\mathbb{R}^2$. The teacher and the student have the same number of layers but different sizes. We vary the model width for MLP, and vary the number of attention heads for Transformer, with a fixed per-head dimension. These choices not only affect the parameter counts, but also govern the learning speed[13].

Why can larger models learn faster? A natural way to learn sparse parity with gradient descent involves first identifying the support S and subsequently computing the product of variables in the support (i.e., $\prod_{i \in S} \mathbf{x}_i$). Empirically, the two stages of learning manifest as a long plateau period in the model's accuracy, followed by a sharp phase transition (Figure 4.4, left and middle). The search for the support is what makes learning problem difficult, as it depends on the input dimension d rather than the support size [Abbe et al., 2023b, Barak et al., 2022b]. The benefit of increasing the width or the number of heads comes from providing more "parallel search queries." For MLP, prior work has

[11] Figure 4.17 provides a comparison in temperature choices for sparse parity learning.

[12] We also experiment with a hierarchical generalization of sparse parity, which is deferred to Section 4.2.7.3.

[13] In terms of the number of samples or the number of training steps, which coincide in our experiments as we use freshly sampled batches.

shown that increasing the width accelerates training [Edelman et al., 2023], which we also observe in Figure 4.10 (left) in appendix. For Transformers though, we find that increasing the number of attention heads is the most effective for improving the convergence speed, as opposed to increasing the per-head dimension or the MLP width. A detailed comparison is provided in Section 4.2.7.2 (Figure 4.13). Given this finding, we will vary the number of attention heads between the teacher and the student, while keeping the per-head dimension fixed. The number of heads hence directly controls the parameter count. This choice also aligns with the practice in open-sourced models such as the Llama series [Touvron et al., 2023].

In the following, we first empirically verify that carefully chosen intermediate teacher checkpoints constitute an *implicit curriculum* for the student to learn from. Then, we show that this curriculum provably improves the speed of learning in the student by improving its training sample efficiency.

4.2.2.1 Accelerating learning with the implicit degree curriculum

The difficulty of the search problem suggests that we can accelerate student learning by providing direct supervision for what the support is [Abbe et al., 2023b]. We show that supplying the intermediate signal from a bigger teacher model accelerates the search process for the smaller model, as described by the following set of results.[14]

(R1) Intermediate teacher checkpoints constitute an implicit degree curriculum. We provide empirical evidence that the supervision from intermediate teacher checkpoints serves as an implicit curriculum supplying strong signals for certain degree-1 monomials, which require fewer samples to learn. In Figure 4.5, we report the correlation between degree-1 monomials and the prediction of the teacher logits at various checkpoints. The correlation for each monomial $x_j, j \in [d]$ is computed as $|\mathbb{E}_{\boldsymbol{x},y}([p_{\mathcal{T}}(\mathbf{x})]_1 \cdot x_j)|$ at each checkpoint $f_{\mathcal{T}}$. Here $[p_{\mathcal{T}}(\mathbf{x})]_1$ refers to the first output dimension of $f_{\mathcal{T}}$, which corresponds to $p(y = 1) = p(\prod_{i \in S} \mathbf{x}_i > 0) = 1 - p(y = 2)$ (recall Definition 23). We take the absolute value as we are only concerned with the magnitude of the correlation. Importantly, these strong correlations emerge when the teacher learns the sparse parity task (i.e., during the phase transition) but diminish with continued training.

Note that the monomials need not be strictly degree-1. While our theory (Section 4.2.2.2) will only focus on degree-1 monomials for the sake of mathematical analysis, low-degree polynomials can still provide acceleration, which we also observe in practice (see Figure 4.12 in the Appendix for such an example). This transient low-degree supervision, available only through intermediate teacher checkpoints, may explain the superior performance of progressive distillation over one-shot distillation (Figure 4.4). We will confirm the provable sample complexity benefit of this implicit low degree curriculum in Section 4.2.2.2. The importance of the implicit curriculum is further strengthened by the superior performance of $(2, T)$-progressive distillation:

(R2) Progressive distillation with a single intermediate checkpoint can outperform one-shot distillation. We consider the extreme version of progressive distillation where only a single intermediate checkpoint is used (in addition to the final checkpoint). Figure 4.5 shows the result for $(2, 1\text{M})$-progressive distillation. We consider 3 candidates for the intermediate teacher checkpoint, occurring

[14] We will mark our results with **(Ri)** throughout the paper for easy reference.

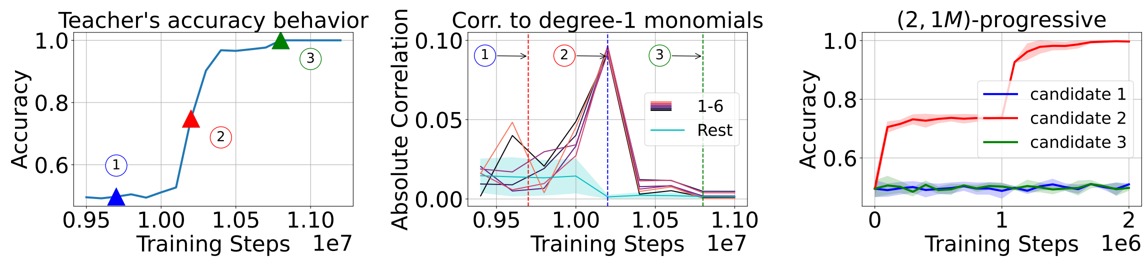


Figure 4.5: **Implicit curriculum for $(100, 6)$-sparse parity.** We compare **3 candidate intermediate checkpoints**, labeled as ①, ②, ③, corresponding to 9.7M, 10.2M, and 10.8M steps, or the beginning, middle, and end of the teacher's phase transition. *Left*: Teacher's accuracy throughout training. *Middle*: During the phase transition, $f_{\mathcal{T}}$ is much more strongly correlated with in-support variables ($x_1, \cdots, x_6$ in this case) than with off-support variables. *Right*: Only candidate ② (i.e., during phase transition) enables $(2, 1M)$-progressive distillation to reach 100% accuracy. We use width-50k teachers and width-100 students; Figure 4.11 shows similar results for width-1000 students.

respectively at the beginning, middle or the end of the teacher's phase transition. Our result demonstrates that the checkpoint selection is crucial, where only the checkpoint during the phase transition is useful in accelerating training.[15] This provides further evidence that the implicit degree curriculum is the key to faster training via progressive distillation.

More complex tasks may require more intermediate checkpoints, which we discuss in more depth in Section 4.2.7.3. Nevertheless, we find that progressive distillation can be run efficiently and effectively across tasks, and a small number of intermediate teacher checkpoints often suffice to accelerate training provided that the checkpoints are properly selected.

4.2.2.2 The low-degree curriculum reduces sample complexity

We now formalize the benefits of progressive distillation for (d, k)-sparse parity in terms of sample complexity. For the sake of mathematical analysis, we take the student $f_{\mathcal{S}}$ and the teacher $f_{\mathcal{T}}$ models to be 1-hidden-layer MLPs with ReLU activations and scalar outputs. Further, the labels y are given as ± 1, where 1 (or -1) corresponds to the class dimension 1 (or 2) in Definition 23.

Following previous works [Barak et al., 2022b, Abbe et al., 2023b, Edelman et al., 2023], we analyze a simplified two-stage training procedure and train the model using the hinge loss: $L_\alpha(\mathbf{x}, y; f_{\mathcal{S}}, f_{\mathcal{T}}) = \alpha \max(0, 1 - f_{\mathcal{S}}(\mathbf{x})y) + (1 - \alpha) \max(0, 1 - f_{\mathcal{S}}(\mathbf{x}) f_{\mathcal{T}}(\mathbf{x}))$.

Let's first recall the hardness of learning sparse parity.[16] For simplicity, we consider the case of MLPs of width $\tilde{O}(2^k)$ trained using online SGD. When learning from data alone, statistical query (SQ, Kearns [1998]) lower bound shows that learning the support for a (d, k)-sparse parity requires $\Omega(d^{k-1})$ samples [Abbe et al., 2023b, Edelman et al., 2023]. We will show that although this lower bound also applies to one-shot distillation from a strong teacher, it can be circumvented when learning from the implicit low-degree curriculum identified in the previous section.

[15] We show similar results for width-1000 students (Figure 4.11) and transformers (Figure 4.16).
[16] A more detailed discussion is provided in Section 4.2.5.1.

Specifically, we compare the sample complexity of one-shot distillation and $(2, T)$-progressive distillation (Section 4.2.1). Both strategies use a well-trained final checkpoint with an error of $\mathcal{O}(\epsilon)$ error for an arbitrarily small $\epsilon > 0$.

Progressive distillation additionally uses the teacher's intermediate checkpoint after its first phase of training, where we can provably show its predictions to have correlations at least $\Omega(1/k)$ to the monomials $x_i, \forall i \in S$. That is, progressive distillation first learns from the intermediate checkpoint and then switches to the final checkpoint, whereas one-shot distillation learns directly from the final checkpoint.

(R3) Progressive distillation reduces sample complexity. We formally demonstrate the sample complexity benefit of progressive distillation.

Theorem 22 (Informal version of Theorem 23)**.** *Consider learning (d, k)-sparse parity with a student model of size $\tilde{m} = \tilde{\Theta}(2^k)$, where $\tilde{\cdot}$ hides polylog factors in d, k. Suppose the teacher has a loss $\mathcal{O}(\epsilon)$ for some small $\epsilon > 0$. Then, the total sample complexity needed for the student to reach ϵ-loss using progressive distillation with 2 checkpoints is $\tilde{\Theta}(2^k d^2 \epsilon^{-2} + k^3)$. However, one-shot distillation requires at least $\Omega(d^{k-1}, \epsilon^{-2})$ samples.*

Proof sketch. We track the training behavior of the teacher model during its two-phase training. We show that at the end of the first phase, the teacher's predictions will have $\Omega(1/k)$ correlations to degree-1 monomials $x_i, \forall i \in S$. In contrast, the correlations are smaller for degree-1 monomials $x_i, \forall i \notin S$. Hence, the teacher's predictions can be written as $\sum_{i \in S} c_i x_i + \sum_{i \notin S} c_i x_i$, plus additional higher degree odd polynomials which can be controlled, with $|c_i| \geq \Omega(1/k)$ for $i \in S$, and $|c_i| = o(1/kd)$, if $i \notin S$.

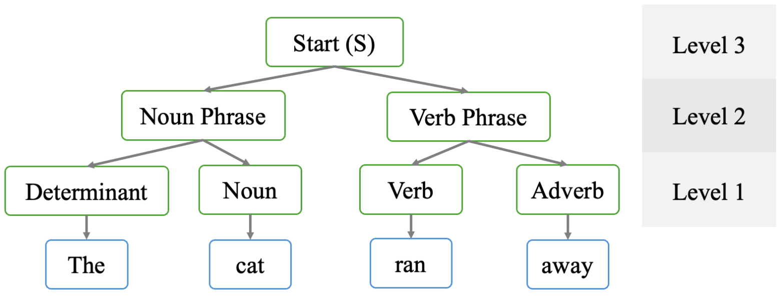
When training on the predictions from this intermediate teacher checkpoints, the correlation gap between in- and off-support degree-1 monomials will be reflected in the gradients of the student's weights. Namely, there is a $\Omega(1/k)$ gap between the support and non-support coordinates in the weight gradients. This gap allows the coordinates $i \in S$ in the student's weights to grow quickly with only $\mathcal{O}(k^2 \log(\tilde{m}))$ samples.

On the other hand, for a teacher that has loss $\mathcal{O}(\epsilon)$, a similar argument can show that the separation gap between the correlations of the teacher's predictions to degree-1 monomials on support and outside support can be at most $\mathcal{O}(\epsilon)$. So, harnessing this gap will require a sample size of at least $\Omega(\epsilon^{-2})$ by concentration inequalities. Learning directly from the labels will require $\Omega(d^{k-1})$ samples from the SQ lower bound as discussed above. This gives the sample complexity differences between one-shot and progressive distillation. The full proof is provided in Section 4.2.6.

□

Remark. One gap between our theory and experiments is that our analysis applies to large-batch SGD with small gradient noise, whereas the experiments use online SGD with batch size 1. Bridging this gap, such as by adapting the analyses in Abbe et al. [2023b] on Gaussian data, is an interesting future direction.

Figure 4.6: An example of a PCFG tree $\mathrm{T}(\mathbf{x})$ that generates $\mathbf{x}$ ="The cat ran away". "The cat" is an example of level-2 span, and "cat" is as a boundary token for the spans of both the level-1 non-terminal `Noun` and the level-2 non-terminal `Noun Phrase`.



4.2.3 Implicit curriculum with PCFGs and Natural language

In this section, we empirically show that an implicit curriculum emerges generally, both when learning on probabilistic context-free grammars (PCFGs) and when performing natural language modeling tasks on the Wikipedia and Books datasets. We focus on BERT models [Devlin et al., 2018b][17], and discuss experiments on GPT-2 [Radford et al., 2019b] in Section 4.2.9.

The masked prediction task. Our experiments will be based on BERT models trained to perform masked prediction, which requires filling in masked-out tokens in an input sequence and excels at feature learning in natural languages [Hewitt and Manning, 2019a, Tenney et al., 2019, Li et al., 2022a].

Definition 24 (Masked prediction task with mask rate p)**.** *Let v denote the vocabulary that contains a special token $\mathcal{C}$, and let h denote an arbitrary sequence length. Given a sequence $\mathbf{x} \in v^h$, sample a set of masked positions $\mathcal{M} \in [h]$ following $P(i \in \mathcal{M}) = p, \forall i \in [h]$. Create a masked input $\mathbf{x}_{\setminus \mathcal{M}}$ from $\mathbf{x}$ by replacing tokens at positions in $\mathcal{M}$ with $\mathcal{C}$, a random token from $\mathcal{X}$, or kept unchanged with probabilities* 80%, 10%, 10% *respectively. Then, the masked prediction objective is the cross-entropy of the model's predictions at positions $i \in \mathcal{M}$ on input $\mathbf{x}_{\setminus \mathcal{M}}$.*

Since we are performing sequence-to-sequence modeling, we need to generalize the definition of the teacher $f_{\mathcal{T}}$ and student $f_{\mathcal{S}}$ from Section 4.2.1 accordingly, denoted as $f_{\mathcal{T}} : v^h \to \mathbb{R}^{h \times C}$ and $f_{\mathcal{S}} : v^h \to \mathbb{R}^{h \times C}$. We will use $p_{\mathcal{T}}^{(i)}(\mathbf{x}; \tau) := \mathrm{softmax}([f_{\mathcal{T}}(\mathbf{x})]_i / \tau)$ to denote the teacher's output distribution on the i_{th} position; similarly for $p_{\mathcal{S}}^{(i)}$. As before, we omit τ when $\tau = 1$. We use the following loss functions for the masked prediction task (Definition 24):

$$\ell(\mathbf{x}; f_{\mathcal{S}}) = \mathbb{E}_{\mathcal{M}} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathrm{KL}(\boldsymbol{e}_{x_i} \| p_{\mathcal{S}}^{(i)}(\mathbf{x}_{\setminus \mathcal{M}})), \tag{4.92}$$

$$\ell_{\mathrm{D}}(\mathbf{x}; f_{\mathcal{S}}, f_{\mathcal{T}}) = \mathbb{E}_{\mathcal{M}} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathrm{KL}(p_{\mathcal{T}}^{(i)}(\mathbf{x}_{\setminus \mathcal{M}}; \tau) \| p_{\mathcal{S}}^{(i)}(\mathbf{x}_{\setminus \mathcal{M}})), \tag{4.93}$$

where $\boldsymbol{e}_y$ is a one-hot vector whose y_{th} entry is 1.

We train BERT models with ℓ, ℓ_{D} and report the average top-1 accuracy on the masked tokens. As discussed in Section 4.2.2.1, the teacher and student have the same depth (4 layers) but differ in the number of attention heads, with 32 heads for the teacher and 8 heads for the student. Each attention

[17] See Section 4.2.8.4.1 for a primer on BERT.

head has dimension 8, so the teacher has width 256 and the student has width 64. All hyperparameter details are in Section 4.2.8.5.

4.2.3.1 n-gram curriculum in PCFGs

We first consider probabilistic context free grammars (PCFGs), which are commonly used to emulate the structure of natural language and thus provide mechanistic insights into language models [Zhao et al., 2023, Allen-Zhu and Li, 2023a]. A PCFG generates sentences following a tree structure; Figure 4.6 shows an example for the sentence "*The cat ran away*." More precisely, a PCFG $\mathcal{G} = (\mathcal{N}, \mathcal{R}, \mathcal{P}, v)$ is defined by a set of non-terminals $\mathcal{N}$, rules $\mathcal{R}$ over the non-terminals, a probability distribution $\mathcal{P}$ over $\mathcal{R}$, and a vocabulary (terminals) v. A sentence $\mathbf{x}$ is associated with a generation tree $\mathrm{T}(\mathbf{x})$, whose intermediate nodes are non-terminals in $\mathcal{N}$, leaf nodes are terminals in v, and edges are defined by rules sampled from $\mathcal{R}$ according to $\mathcal{P}$. A formal definition of PCFG is provided in Section 4.2.8.1. Our choices of PCFGs are taken from Allen-Zhu and Li [2023a], where all leaves in the same tree have the same distance to the root. Experiments in the main paper are based on the PCFG `cfg3b` generated by depth-7 trees, and results on other PCFGs are deferred to Section 4.2.8.3.

4.2.3.1.1 Progress measures of implicit curriculum

Unlike our experiments on parity, what constitutes as feature is less straightforward for PCFG. We will use three progress measures to quantify the implicit curriculum for masked language modeling on PCFGs, based on n-gram statistics and non-terminal prediction.

Measures that use n-gram statistics will measure the dependence of the model's predictions on tokens in the neighboring contexts, defined as follows:

Definition 25 (n-gram neighboring context)**.** *For a h-length sentence $\boldsymbol{x} \in v^h$ and for $i \in [h]$, we define the n-gram neighboring context around the i^{th} token as the set of tokens at positions within $(n-1)/2$ distance from i, denote as n-gram$(i) := \{j : \max(i - \lceil (n-1)/2 \rceil, 0) \le j \le \min(i + \lfloor (n-1)/2 \rfloor, h)\}$.*

In the example of Figure 4.6, for the word "cat", its 3-gram neighboring context consists of words "The" and "ran", and its 5-gram neighboring context additionally includes the word "away." The choice of n-grams is inspired by results in Zhao et al. [2023], which show that a BERT model can solve masked prediction by implementing a dynamic programming algorithm that builds hierarchically on increasingly larger n-gram neighboring context spans (Definition 25). A model that primarily uses short n-gram neighboring context will be largely affected if the tokens within the context are perturbed during evaluation. This motivates us to consider two n-gram based measures.

Measure 1: Robustness to removing n-gram context. Our first progress measure of feature learning checks how the model's prediction changes when the n-gram context is present or absent. For each masked position i, we measure the total variation (TV) distance between the probability distributions when masking out only the current token, and when masking out all the tokens in n-gram(i), i.e. the neighboring n-gram context centered at i. Recall that $\mathbf{x}_{\setminus \mathcal{M}}$ denotes a masked version of $\mathbf{x}$ with masked set $\mathcal{M}$ (Definition 24), and that $p^{(i)}$ denotes a model's output probability distribution at the

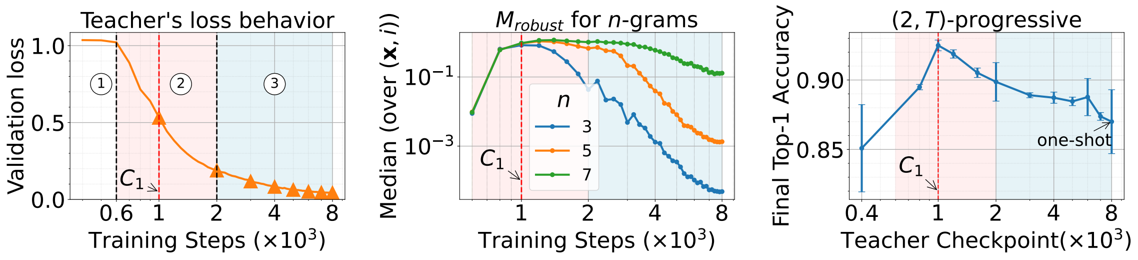


Figure 4.7: BERT on the PCFG `cfg3b`. *Left*: A 32-head teacher's loss exhibits three distinct phases: ① an initial phase with little change, ② a middle phase with a rapid drop, and ③ a final plateauing phase until the end of training. The triangles mark the selected checkpoints for progressive distillation, with the first teacher checkpoint (denoted by C_1) located at the middle of phase ②. *Middle*: L_{robust} across training, which peaks at C_1. The model gets more robust to shorter n-gram perturbation as training progresses. The median is taken over the input sequences. *Right*: A 8-head student's final accuracy with $(2, T)$-progressive distillation after 4000 total training steps. The x-axis marks the choice of the first teacher checkpoint. T is grid-searched over $\{500, 1000, 2000\}$. The best performance is obtained by choosing C_1. Although results in the plots are for a single training run of the teacher, similar behaviors occur robustly across random seeds.

i_{th} position. Then, our first measure is defined as

$$L_{\text{robust}}(f, \mathbf{x}, i, n) = \text{TV}(p^{(i)}(\mathbf{x}_{\setminus\{i\}}), p^{(i)}(\mathbf{x}_{\setminus n\text{-gram}(i)})). \tag{4.94}$$

We report median of $L_{\text{robust}}(f, \mathbf{x}, i, n)$ over randomly sampled $\mathbf{x}$ and i [18]. A larger $L_{\text{robust}}(f, \mathbf{x}, i, n)$ indicates that the model heavily depends on neighboring n-gram context tokens for the masked prediction.

Measure 2: Closeness between full and n-gram predictions. Our second progress measure examines the change in predictions when the model is given the full sequence versus only a local n-gram window:

$$L_{\text{close}}(f, \mathbf{x}, i, n) = \text{TV}(p^{(i)}(\mathbf{x}_{\setminus\{i\}}), p^{(i)}(\mathbf{x}_{n\text{-gram}(i)\setminus\{i\}})), \tag{4.95}$$

where $\mathbf{x}_{n\text{-gram}(i)\setminus\{i\}}$ denotes the n-gram context centered at position i, minus the position i itself. We report median of $L_{\text{close}}(f, \mathbf{x}, i, n)$ over randomly sampled $\mathbf{x}$ and i. A large $L_{\text{close}}(f, \mathbf{x}, i, n)$ indicates that the model utilizes contexts outside a n-gram window in its predictions.

Measure 3: Non-terminal prediction. Finally, we also measure how well the model outputs encode the features of the underlying PCFG by checking the accuracy at predicting non-terminals [Allen-Zhu and Li, 2023b]. The predictions are given by a linear classifier on top of the output embeddings.

Definition 26 (PCFG non-terminal prediction task)**.** *Define the span of a non-terminal n as the set of terminals within the subtree rooted at n, denoted by span(n). The (right) boundary of span(n) refers to the rightmost position within span(n). We say a non-terminal is of level i if it is at distance i from the root. Then, the level-i non-terminal prediction task aims to predict $n^{(i)}$ at the boundary of $n^{(i)}$.*

As an example, in Figure 4.6, the *level*-2 *non-terminal prediction task* aims to predict the non-terminals

[18]Our observations stay the same for other percentiles.

`Noun Phrase` and `Verb Phrase` at words “cat” and “away” respectively. More details are provided in Section 4.2.8.2.

4.2.3.1.2 Empirical verification of the n-gram curriculum

Similar to Section 4.2.2.1, we will start with examining the training dynamics of the teacher model. We observe a phase transition period akin to that of sparse parity, during which we identify an inflection point concerning L_{robust} and L_{close}. This inflection point proves to be a crucial intermediate checkpoint. We then demonstrate that progressive distillation improves feature learning in the student model, substantiated by the three measures defined in Section 4.2.3.1.1.

For training dynamics, we observe 3 distinct phases of training in the teacher's loss (Figure 4.7 left): 1) an initial phase where the loss doesn't change much for the first 5% of training; 2) a rapid loss drop phase in the next $\approx$ 20% of training; and 3) a final phase of slow loss drop till end of training. In particular, the rapid loss drop phase is reminiscent of the phase transition in sparse parity (Section 4.2.2). Moreover, we identify an inflection point (marked by C_1) during the second phase: before the inflection point, the robust loss L_{robust} increases (Figure 4.7 middle), and the loss L_{close} stays high (Figure 4.25 left); after the inflection point, both L_{robust} and L_{close} start to drop rapidly, suggesting that the model learns to utilize longer contexts as opposed to short neighboring n-grams.

(R4) The inflection point is best for $(2, T)$-progressive distillation. We study the importance of each teacher checkpoint by comparing the performance of $(2, T)$-progressive distillation, where the student learns from a single intermediate checkpoint in addition to the final checkpoint. The value of T is grid-searched (more details in Section 4.2.8.5). For the choice of the intermediate checkpoint, Figure 4.7 shows that the best intermediate checkpoint is the one at the inflection point (at 1000 training steps), which we denote as C_1. Note that at the inflection point, the teacher has the highest reliance on shorter n-grams (e.g. for $n = 3$), which are analogous to the low-degree monomials in Section 4.2.2 and serves as intermediate tasks that are likely easier to learn. Hence, C_1 being the optimal checkpoint choice further strengthens our hypothesis that an implicit curriculum is the key to the acceleration enabled by progressive distillation.

Following (**R4**), we will choose the checkpoints for progressive distillation at training steps that are multiples of that of C_1, i.e. at steps $\{i \times 10^3\}_{i=1}^{8}$. As shown in Figure 4.4 (right), progressive distillation helps the student learn faster than both one-shot distillation and cross entropy training. Furthermore, progressive distillation leads to improved feature learning.

(R5) Progressive distillation improves feature learning on PCFG. Progressive distillation improves over one-shot or no distillation over all 3 measures mentioned in Section 4.2.3.1.1. As shown in Figure 4.8, progressive distillation makes the student better utilize long contexts rather than local n-gram windows, evidenced by a lower L_{robust} and L_{close}. The student can also better predict the non-terminals, suggesting a better structural learning of the underlying PCFG.

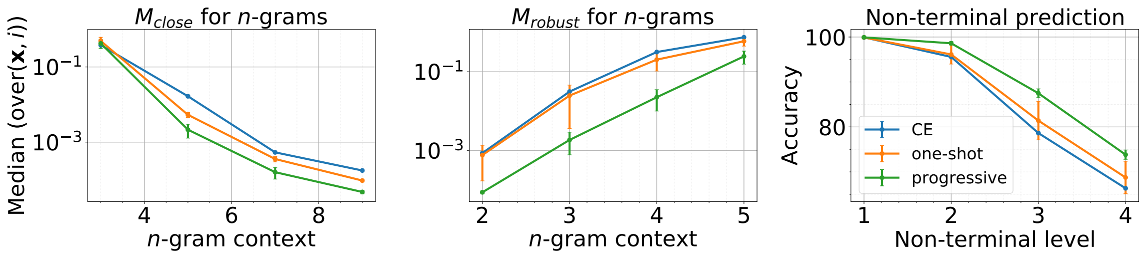


Figure 4.8: Comparisons on a 8-attention head BERT model. (Left) L_{close} for different n-grams. Progressive distillation has a lower L_{close} with longer n-gram context. (Middle) L_{robust} for different n-grams. Progressive distillation has a lower L_{robust} for all n-gram contexts. (Right) Probe performance to predict the non-terminals (NTs) (Definition 26). Progressive distilled student performs better when probed for higher level non-terminals in its contextual embeddings.

4.2.3.2 Beyond synthetic setups: implicit curriculum in natural languages

We conduct experiments on BERT training [Devlin et al., 2018b] on Wikipedia and Books (details in Section 4.2.10). The teacher and student both have 12 layers, with 12 and 4 attention heads per-layer respectively. Each attention head is of dimension 64, corresponding to a width-768 teacher and a width-256 student. Similar to PCFG, the teacher's loss exhibits 3 distinct phases (Figure 4.9 left), with an inflection point marking the change in L_{robust} (Figure 4.9 middle). The inflection point can hence provide an implicit curriculum towards easier-to-learn local n-grams. Finally, progressive distillation helps the student achieve better accuracy at masked language prediction (Figure 4.9 right).

Connections to related works. Our results align with those of Chen et al. [2023], who observed a phase transition in loss when training BERT on real-world language data corresponding to the model learning syntax rules of language. Comparable findings were also reported in a concurrent work on matrix completion [Gopalani et al., 2024]. For auto-regressive models, prior work has discussed the emergence of n-gram induction heads which indicate phases in which the model learns to perform in-context learning [Akyürek et al., 2024, Quirke et al., 2023, Olsson et al., 2022]. We observe similar behavior for PCFGs and Wikipedia datasets and quantify the phase change using n-gram context dependence. We take a step further and leverage the phase transitions to accelerate the training of a smaller student model.

4.2.4 Discussions

We have shown that progressive distillation can improve the student's feature learning via an implicit curriculum provided by the intermediate checkpoints. We discuss limitations and potential future directions below, and provide preliminary results for some of them in the appendix (see Section 4.2.5).

Impact of temperature. The teacher temperature τ is an important hyperparameter in knowledge distillation, where varying τ can sometimes lead to a greater performance gain than changing the distillation method [Touvron et al., 2021, Harutyunyan et al., 2022]. Our results are consistent with these prior findings. However, our experiments use limited temperature choices, i.e. the default

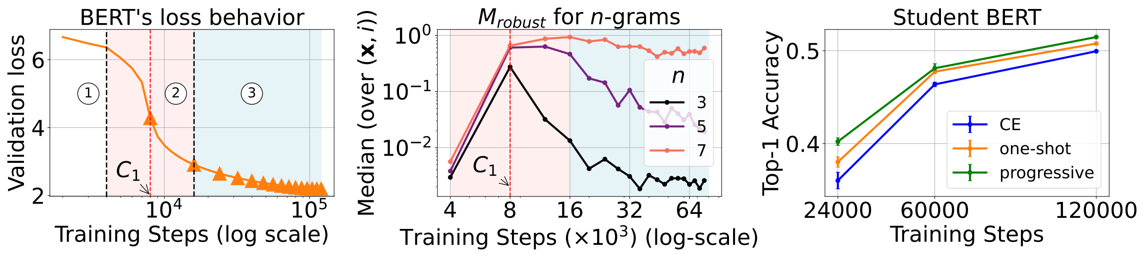


Figure 4.9: BERT on Wikipedia and Books. Left to right: (a) Similar to our experiments on PCFG (Figure 4.7), we observe three distinct phases in the loss behavior of 12-head teacher. The rapid loss drop phase signifies a transition phase for the model. The triangles mark the selected checkpoints for progressive distillation, with the first teacher checkpoint roughly picked in the middle of the second phase (C_1). (b) We observe L_{robust} peaks at C_1, and the model gets more robust to shorter n-gram context masking, as training progresses. (c) A 4-head student achieves better top-1 accuracy on masked prediction objective with progressive distillation.

($\tau = 1.0$) and low temperature ($\tau = 10^{-4}$ or 10^{-20}). A more precise understanding of temperature, especially its impact on optimization, is an interesting direction for future work.

Distillation via generations. Another related distillation setting is training smaller (language) models using the generations of larger models, which has been shown to greatly improve various abilities [Liu et al., 2024, Yue et al., 2023, Yu et al., 2023, Luo et al., 2023, Chaudhary, 2023, Taori et al., 2023, Zheng et al., 2023b, Liu and Yao, 2024, Agarwal et al., 2024, Mitra et al., 2024]. There are two differences between our experiments and these generation-based approaches. First, the supervision in our experiments are distributions (over classes or the vocabulary), while generations are samples from distributions. Our experiments with a low or zero temperature provide positive evidence towards bridging this gap, but the precise effect remains to be explored. More importantly, given an input, there is a unique supervision in our settings, whereas there could be multiple generations given by multiple steps of unrolling of the teacher. Extending our framework to these generative setting will be an important direction for future work.

4.2.5 Overview of the appendix

The appendix provides omitted proofs and additional empirical explorations, which we outline below.

Omitted proofs We will start with the proof of Theorem 22 in Section 4.2.6. The main idea is to show that the teacher can develop stronger correlation to in-support variables than to off-support variables, which can then be utilized by the students to reduce sample complexity.

Additional empirical results on sparse parity We present more experiments with MLP (Section 4.2.7.1) and Transformers (Section 4.2.7.2), as well as results on learning a hierarchical extension of sparse parity (Section 4.2.7.3). For Transformer experiments, we study how scaling along different dimensions of the architecture, such as MLP width and number of attention heads, affects the search of support for sparse parity. We discuss the effect of temperature in Figure 4.17. For the hierarchical extension of sparse parity, we show that the implicit curriculum occurs in different phases, which suggests a natural choice for number of intermediate checkpoints used in progressive distillation.

Masked prediction on PCFGs In Section 4.2.8.4, we provide a formal definition of probablistic context-free grammar (PCFG) and introduce the PCFGs that we use from Allen-Zhu and Li [2023b]. We then provide details of our experimental setup and conduct extensive ablation studies on training a BERT model using the masked prediction task with PCFG data. We experiment with variants of progressive distillation and confirm that they lead to improved performance on PCFGs, as measured by accuracy and the three progress measures introduced in Section 4.2.3.1.1. Furthermore, we investigate the effect of temperature, masking rate, and PCFG variation in Section 4.2.8.7.

Next-token prediction on PCFGs In Section 4.2.9, we conduct next-token prediction experiments using GPT-2 models on PCFG "cfg3f", i.e. the most complex PCFG in Allen-Zhu and Li [2023a]. We characterize conditions under which progressive distillation provides significant gains.

4.2.5.1 Additional related works

Understanding knowledge distillation There have been many works dedicated to understanding the effectiveness of knowledge distillation [Hinton et al., 2015, Mobahi et al., 2020, Menon et al., 2021, Dao et al., 2021, Nagarajan et al., 2024, Pareek et al., 2024]. For classification tasks, which are the focus of most knowledge distillation works, one intuitive explanation is that the teacher output provides a distribution over the class labels, which is more informative than the one-hot data labels. Menon et al. [2021] formalizes this intuition and shows that a teacher that provides the Bayes class probabilities leads to a tighter generalization gap. Motivated by their result and the observation that a high-accuracy teacher can be poorly calibrated, Ren et al. [2022] proposes to supervise the student using a moving average of the teacher across the training trajectory. While Ren et al. [2022] uses information of trajectory, their student learns from a fixed target throughout training, which is a major difference from progressive distillation. The teacher supervision also provides regularization benefits, such as controlling the bias-variance tradeoff [Zhou et al., 2020], encouraging sparsity [Mobahi et al., 2020], or as a form of label smoothing [Yuan et al., 2020]. Finally, prior work by Mobahi et al. [2020] and concurrent work by Pareek et al. [2024] have studied multi-step distillation, which also uses multiple teachers throughout training. However, unlike the progressive distillation setting considered in this work, they study self-distillation where the student in the previous distillation round becomes the teacher in the next round, and the results are for regression rather than classification.

Learning sparse parity There are well established hardness results for learning sparse parity. When given access to labels only, learning (d, k)-sparse parity with gradients from finite samples is an example of learning with statistical queries (SQ) [Kearns, 1998], for which a $\Omega(d^k)$ SQ computational lower bound applies [Edelman et al., 2023]. When learning with a fully-connected network (MLP), these parallel queries correspond to a combination of model width (i.e. neurons) and training steps,

[19] and hence the SQ lower bound implies a fundamental trade-off between the width, the number of training steps, and the number of samples [Edelman et al., 2023]. In particular, given the same number of training steps, narrower models require more samples to learn parity.

Feature learning In this work, we use feature learning to refer to a learning process that recovers a low-dimensional "feature" which helps reduce sample complexity. Sparse parity is a task that can benefit from feature learning, where the feature is the support. For the special case of $k = 2$, Glasgow [2024] shows that feature learning using a jointly-optimized 2-layer neural network can reduce the sample complexity from $\Theta(d^2)$ (corresponding to learning with NTK [Wei et al., 2019, Ghorbani et al., 2019]) to $O(d \text{poly} \log d)$. Sparse parity is an example of a single-/multi-index function, where the label is determined by a 1-dimensional/low-dimensional projection of the data. These functions have also been studied on Gaussian inputs [Nichani et al., 2022, Abbe et al., 2022, 2023b, Damian et al., 2024a,b] and have known separation between neural networks [Abbe et al., 2022, 2023b] and non-feature-learning kernel methods [Hsu, 2021].

Benefit of width in optimization Prior work has shown that width plays an important role in the optimization difficulty, where wider networks are more optimized easily. Du and Hu [2019] shows that sufficient width is necessary for the optimization on deep linear networks. Multiple works show that overparameterization leads to favorable optimization landscape, such as fewer sub-optimal local minima [Soudry and Hoffer, 2017, Soltanolkotabi et al., 2018] or guaranteed convergence at the limit [Chizat and Bach, 2018, 2020]. Wider models also exhibit faster decaying loss empirically [Yang et al., 2022, Bordelon et al., 2024a]. Most related to our focus on learning sparse parity, Edelman et al. [2023] relates the width to the number of parallel statistical queries (SQs). Combined with sparse parity's SQ lower bound, their result implies a trade-off where a larger width requires fewer optimization steps. Our work also acknowledges the benefit of width in optimization, but takes a different perspective by demonstrating that a smaller student can inherit the optimization benefit when learning from a higher-width teacher. Moreover, we consider the number of attention heads as another scaling dimension for Transformers, where the intuition is similar to having more "paths" [Dong et al., 2021]. There have been results on studying the limiting output distribution as the number of attention heads goes to infinity [Hron et al., 2020, Bordelon et al., 2024b], though to our knowledge, there are no quantitative descriptions for finite number of heads.

4.2.6 Proofs of results in Section 4.2.2.2

We provide the formal version of Theorem 22 in this section.

Recall that the teacher model is defined as

$$f_{\mathcal{T}}(\mathbf{x}) = \sum_{i=1}^{m} a_i \sigma\left(\langle \boldsymbol{w}_i, \mathbf{x} \rangle + b_i\right).$$

[19]More precisely, it is a combination of width and steps, as well as the batch size which affects the precision of the stochastic gradient. We omit the impact of batch size here since we keep the batch size unchanged in the experiments.

Algorithm 2: 2-stage training

```
Require: Stage lengths: $T_1$, $T_2$, learning rates $\eta_1, \eta_2$, batch size $B_1, B_2$, weight decay $\lambda_1, \lambda_2$.
  for $t \in [0, T_1]$ and all $i \in [m]$ do
    Sample $B_1$-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_1}$.
    Update the weights $\boldsymbol{w}_i$ as $\boldsymbol{w}_i^{(t)} \leftarrow \boldsymbol{w}_i^{(t-1)} - \eta_1 \mathbb{E}_{(\mathbf{x},y)\in\{(\mathbf{x}^{(j)},y^{(j)})\}_{j=1}^{B_1}} \nabla_{\boldsymbol{w}_i} \left( L_{\theta^{(t)}}(\mathbf{x}, y) + \lambda_1 \|\boldsymbol{w}_i\|^2 \right)$.
  end for
  for $t \in [0, T_2]$ and all $i \in [m]$ do
    Sample $B_2$-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_2}$.
    Update the outer layer weights $a_i$ as
    $a_i^{(t+T_1)} \leftarrow a_i^{(t+T_1-1)} - \eta_2 \mathbb{E}_{(\mathbf{x},y)\in\{(\mathbf{x}^{(i)},y^{(i)})\}_{j=1}^{B_2}} \nabla_{a_i} \left( L_{\theta^{(t+T_1-1)}}(\mathbf{x}, y) + \lambda_2 a_i^2 \right)$.
  end for
```

The student model is similarly defined as

$$f_{\mathcal{S}}(\mathbf{x}) = \sum_{i=1}^{\tilde{m}} \tilde{a}_i \sigma\left(\langle \tilde{\boldsymbol{w}}_i, \mathbf{x}\rangle + \tilde{b}_i\right).$$

Setup We assume the data points are sampled at random from $\mathcal{U}(\{\pm 1\}^d)$. Without loss of generality, let the target k-sparse parity function be $y = x_1 x_2 \cdots x_k$. *Symmetric initialization:* Following [Barak et al., 2022b], we use the following symmetric initialization: for each $1 \le i \le m/2$,

$$\begin{aligned} &\boldsymbol{w}_i \sim \mathcal{U}(\{\pm 1\}^d), \quad b_i \sim \mathcal{U}(\{-1 + k^{-1}, \cdots, 1 - k^{-1}\}), \quad a_i \sim \mathcal{U}(\{\pm 1/m\}), \\ &\boldsymbol{w}_{i+m/2} = \boldsymbol{w}_i, \quad b_{i+m/2} = b_i, \quad a_{i+m/2} = -a_i. \end{aligned}$$

Two-stage training: Following prior work [Barak et al., 2022b, Abbe et al., 2023b, 2024], we adopt a two-stage batch gradient descent training, where we first train the first-layer weights $\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m\}$, keeping the output weights $\{a_i\}_{i=1}^m$ fixed. In the second stage of training, we fit the output weights $\{a_i\}_{i=1}^m$ while keeping others fixed. We keep the biases $\{b_i\}_{i=1}^m$ fixed throughout training. Similar strategy for training the student model as well. The teacher is trained with hinge loss, given by $\ell(\mathbf{x}, y) = \max(0, 1 - f_{\mathcal{T}}(\mathbf{x})y)$. The student is trained with $\ell_{\mathrm{D}}(\mathbf{x}, y; f_{\mathcal{S}}, f_{\mathcal{T}}) = \max(0, 1 - f_{\mathcal{S}}(\mathbf{x}) f_{\mathcal{T}}(\mathbf{x}))$.

The training process is summarized in Algorithm 2.

Sample complexity benefits with progressive distillation for the student Our result is that progressive distillation provably reduces the sample complexity compared to (one-shot) distillation or no distillation. The key is to establish a separation between the correlations with in-support and off-support variables, which happens with high probability as formalized in Corollary 3. Under such event, we show:

Theorem 23 (Sample complexity benefits with progressive distillation)**.** *Suppose the teacher model has been trained with 2-stage training in Algorithm 2, which satisfies the conditions in Corollary 3 at the end of first stage and achieves loss $\mathcal{O}(d^{-c})$ for some constant $c \ge 1$ at the end of the second stage. Suppose we train a student model $f_{\mathcal{S}}$ of size $\tilde{m} = \tilde{\Theta}(2^k k)$ using the following two strategies:*

Corollary 3: conditions satisfied by the teacher after first phase

W.h.p. the output of the teacher after the first phase satisfies the following condition for all i.

$$\left|\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) \cdot \text{Maj}(\mathbf{x}) x_i\right| \geq \Omega(k^{-1}), \quad \text{if } i \in [k],$$
$$\left|\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) \cdot \text{Maj}(\mathbf{x}) x_i\right| \leq o(k^{-1}), \quad \text{if } i \notin [k].$$

1. ***Progressive distillation:*** *Train for the first $T_1 = 1$ steps w.r.t. the teacher's logits at T_1 checkpoint. Then, train with the final teacher checkpoint in the second stage.*
2. ***Distillation:*** *Train with the final teacher checkpoint throughout training.*

Then,

1. *Under progressive distillation, the total sample complexity to reach a loss of ϵ with probability $1-\delta$ is*
$$\Theta(k^2 \log(d\tilde{m}/\delta) + 2^k d^2 k^4 \epsilon^{-2} \log(k/\delta)).$$
2. *The necessary sample complexity under distillation is at least $\Omega(d^{\min(2c,k-1)})$.*

The proof consists of two parts: 1) showing that the teacher develops strong correlation with the in-support variables after the first stage of training (Lemma 63, Corollary 3), and 2) showing that given the support, the second phase of training converges quickly (Corollary 4). These two helper lemmas are proven in Section 4.2.6.1.1 (first stage) and Section 4.2.6.1.2 (second stage). The proof of Theorem 23 is given in Section 4.2.6.2.

Notations Before stating the proofs, we provide a list of necessary notations.

- At any training step t, $f_{\mathcal{T}}^{(t)}$ will refer to the teacher's output at that step. Its parameters are referred to as $\theta^{(t)} = \{a_i^{(t)}, \boldsymbol{w}_i^{(t)}, b_i^{(t)}\}_{i=1}^m$. The loss for $f_{\mathcal{T}}^{(t)}$ is denoted by $L(f_{\mathcal{T}}^{(t)})$ or $L_{\theta^{(t)}}$. Notations for the student $f_{\mathcal{S}}$ are defined similarly.
- Given a set $\tilde{S}$, $\chi_{\tilde{S}}$ denotes the Fourier function on $\tilde{S}$, where $\chi_{\tilde{S}}(\mathbf{x}) = \prod_{i\in\tilde{S}} x_i$. We are particularly interested in $\tilde{S} = S$, i.e. the support of the sparse parity.
- Maj : $\{\pm1\}^d \to \pm1$ represents the majority function. On any $\mathbf{x}$, Maj returns the sign of $\sum_{i=1}^d \mathbf{x}_i$. ζ_i for $i \geq 1$ represents its ith fourier coefficient, i.e. $\zeta_i = \mathbb{E}_{\mathbf{x},y}\text{Maj}(\mathbf{x})\chi_S(\mathbf{x})$ for any $S \in \{0,1\}^d$ with $|S| = i$. $\zeta_i = 0$ when i is even, and $\zeta_i = \Theta(i^{-1/3}/\binom{d}{i})$ when i is odd [O'Donnell, 2014].
- τ_g denotes the error tolerance in the gradient estimate due to mini-batch gradient estimation: let g be the population gradient and $\hat{g}$ be the estimated gradient with a few examples, τ_g is defined such that $\|\hat{g} - g\|_\infty \leq \tau_g$. A τ_g-error gradient estimate can be obtained using a batch size of $\tilde{\Omega}(1/\tau_g^2)$.

4.2.6.1 Analysis for the teacher

4.2.6.1.1 First stage analysis for the teacher

First, we show that with an appropriate learning rate, the magnitude of the weights w_{ij} on coordinates $i \in S$ increases to $\frac{1}{2k}$, while the coordinates $i \notin S$ stay $\mathcal{O}\left(\frac{1}{kd}\right)$ small.

Lemma 63 (Single step gradient descent, adapted from Claims 1, 2 in Barak et al. [2022b])**.** *Fix $\tau_g, \delta > 0$. Set T_1 as 1. Suppose the batch size $B_1 \geq \Omega(\tau_g^{-2} \log(md/\delta))$. For learning rate $\eta_1 = \frac{m}{k|\zeta_{k-1}|}$ and $\lambda_1 = 1$, the following conditions hold true for all neurons $i \in [m]$ at the end of first stage of training w.p. at least $1 - \delta$.*

1. $\left| w_{ij}^{(1)} - \frac{\text{sign}(a_i^{(0)} \zeta_{k-1}) \, \text{sign}(\chi_{[k] \setminus \{j\}}(\boldsymbol{w}_i^{(0)}))}{2k} \right| \leq \frac{\tau_g}{|\zeta_{k-1}|}$*, for all $j \in [k]$.*

2. $\left| w_{ij}^{(1)} - \frac{\zeta_{k+1}}{|\zeta_{k-1}|} \frac{\text{sign}(a_i^{(0)}) \, \text{sign}(\chi_{[k] \cup \{j\}}(\boldsymbol{w}_i^{(0)}))}{2k} \right| \leq \frac{\tau_g}{|k\zeta_{k-1}|}$*, for all $j > k$.*

Proof. The proof follows that of [Barak et al., 2022b], which we outline here for completeness. The proof has two major components: First, the magnitude of the population gradient at initialization reveals the support of the sparse parity. Second, the batch gradient and the population gradient can be made sufficiently close given a sufficiently large batch size. We will explain each step below.

Claim 7. *At initialization, the population gradient of the weight vector in neuron i is given by $\mathbb{E}_{\mathbf{x},y} \nabla_{w_{ij}} \ell(\mathbf{x}, y; f_{\mathcal{T}}^{(0)}) = -\mathbb{E}_{\mathbf{x},y} \nabla_{w_{ij}} f_{\mathcal{T}}^{(0)}(\mathbf{x}) y$, which can be split across the coordinates as*

$$\mathbb{E}_{\mathbf{x},y} \nabla_{w_{ij}} f_{\mathcal{T}}^{(0)}(\mathbf{x}) y = -\frac{1}{2} a_i^{(0)} \zeta_{k-1} \chi_{[k] \setminus \{j\}}(\boldsymbol{w}^{(0)}), \quad \text{for all } j \in S$$
$$\mathbb{E}_{\mathbf{x},y} \nabla_{w_{ij}} f_{\mathcal{T}}^{(0)}(\mathbf{x}) y = -\frac{1}{2} a_i^{(0)} \zeta_{k+1} \chi_{[k] \cup \{j\}}(\boldsymbol{w}^{(0)}), \quad \text{for all } j \notin S$$

Thus, the gradient of the weight coordinates w_{ij} for any neuron i and $j \in S$ has magnitude $|\zeta_{k-1}|$, while the gradients of the weight coordinates w_{ij} for any neuron i and $j \notin S$ has magnitude $|\zeta_{k+1}|$. The gap between the gradient in support and out of support is given by $|\zeta_{k-1}| - |\zeta_{k+1}| \geq 0.03((d-1)^{-(k-1)/2})$ (Lemma 2 in Barak et al. [2022b]).

The second component involves applying a hoeffding's inequality to show the gap between sample and population gradient.

Claim 8. *Fix $\delta, \tau_g > 0$. For all i, j, for a randomly sampled batch of size B_1, $\{(\mathbf{x}_k, y_k)\}_{k=1}^{B_1}$, with probability at least $1 - \delta$,*

$$\left| \mathbb{E}_{\mathbf{x},y \sim \mathcal{U}(\{\pm\}^d)} \nabla_{w_{ij}} f_{\mathcal{T}}^{(0)}(\mathbf{x}) - \mathbb{E}_{\{(\mathbf{x}_k, y_k)\}_{k=1}^{B_1}} \nabla_{w_{ij}} f_{\mathcal{T}}^{(0)}(\mathbf{x}) \right| \leq \tau_g,$$

provided $B_1 \geq \Omega(\tau_g^{-2} \log(md/\delta))$.

Because we want the noise τ_g to be smaller than the magnitude of the true gradients for the coordinates in the support S, we want τ_g to be smaller than $|\zeta_{k-1}|$. We set this to get favorable condition for second phase of training (see Lemma 65). □

On the other hand, we show that after the first phase, the output of the network has positive correlations to the individual variables in the support of the label function, and thus the checkpoint after the first phase can be used to speed up training of future models.

Lemma 64 (Correlation with in-support variables)**.** *Under the event that the conditions in Lemma 63 are satisfied by each neuron, which occurs with probability at least $1 - \delta$ w.r.t. the randomness of initialization and sampling, the output of the model after the first phase satisfies the following conditions:*

1. *$\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) x_i \geq \frac{1}{8k} + \mathcal{O}(\tau_g d \left|\zeta_{k-1}\right|^{-1}) + \mathcal{O}(m^{-1/2})$ for all $i \in S$.*
2. *$\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) x_i \leq \mathcal{O}((kd)^{-1})$ for all $i \notin S$.*
3. *$\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) \chi_S(\mathbf{x}) \leq \mathcal{O}(\tau_g d \left|\zeta_{k-1}\right|^{-1})$ for all S with even $|S|$.*
4. *$\left\|f_{\mathcal{T}}^{(1)}\right\|_2^2 = \mathbb{E}_{\mathbf{x},y}[f_{\mathcal{T}}^{(1)}(\mathbf{x})]^2 \leq \mathcal{O}(d/k)$.*

Proof. Consider a neuron $i \in [m/2]$ and its symmetric counterpart $i + m/2$. W.L.O.G., we assume $\operatorname{sign}(w_{ij}^{(0)}) = \operatorname{sign}(a_i^{(0)} \zeta_{k-1})$ for all $j \in [k]$, and $\operatorname{sign}(a_i^{(0)}) = 1$. Recall that k is assumed to be even, hence $\operatorname{sign}(\chi_{[k]}(\boldsymbol{w}_i^{(0)})) = 1$. Then, the condition in Lemma 63 can be simplified as

$$w_{ij}^{(1)} = \frac{1}{2k} + v_{ij}, \quad w_{i+m/2,j}^{(1)} = -\frac{1}{2k} - v_{ij}, \text{ for all } j \in [k],$$
$$w_{ij}^{(1)} = \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\operatorname{sign}(w_{ij}^{(0)}) + v_{ij}, \quad w_{i+m/2,j}^{(1)} = -\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\operatorname{sign}(w_{ij}^{(0)}) + v_{ij}, \text{ for all } j \geq k,$$

where v_{ij} satisfies the following conditions.

$$\left|v_{ij}\right| \leq \frac{\tau_g}{|\zeta_{k-1}|}, \text{ for all } j \in [k],$$
$$\left|v_{ij}\right| \leq \frac{\tau_g}{|k\zeta_{k-1}|}, \text{ for all } j \geq k.$$

Then, the sum of the output of the neurons i and $i + m/2$ on an input $\mathbf{x}$ (ignoring the magnitude of a_i) is given by

$$(f_{\mathcal{T}}^{(1)})_i(\mathbf{x}) = \sigma\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right)$$
$$- \sigma\left(-\frac{1}{2k}\sum_{j=1}^{k} x_j - \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right),$$

and

$$f_{\mathcal{T}}^{(1)}(\mathbf{x}) = \sum_{i=1}^{m/2} a_i (f_{\mathcal{T}}^{(1)})_i(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m/2}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x}).$$

1. In-support correlations: We are interested in the correlation of this function to a variable x_u for $u \in S$. We argue for $u = 1$, as the similar argument applies for others. Thus, we are interested in

$$\mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_1 = \mathbb{E}_{\mathbf{x},y}\sigma\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right)x_1$$
$$- \sigma\left(-\frac{1}{2k}\sum_{j=1}^{k} x_j - \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right)x_1. \tag{4.96}$$

We focus on the first term; argument for the second term is similar. First of all, we can ignore $\langle \boldsymbol{v}_i, \mathbf{x}\rangle$ incurring an error of $\mathcal{O}(\tau_g d \left|\zeta_{k-1}\right|^{-1})$.

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{x},y}\sigma\left(\frac{1}{2k}\sum_{j=1}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\right)x_1\\
&\quad=\mathbb{E}_{\mathbf{x},y:x_1=+1}\sigma\left(\frac{1}{2k}+\frac{1}{2k}\sum_{j=2}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\right)\\
&\quad-\mathbb{E}_{\mathbf{x},y:x_1=-1}\sigma\left(-\frac{1}{2k}+\frac{1}{2k}\sum_{j=2}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\right)\\
&\quad\geq\frac{1}{2k}\mathbb{E}_{\mathbf{x},y}\mathbb{I}\left(\frac{1}{2k}\sum_{j=2}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\geq 0\right).
\end{aligned}
$$

The final step follows from the observation that the argument of σ in the first term is $\frac{1}{k}$ higher than the argument of σ in the second term. This implies that when the first term is non-zero, it's at least $\frac{1}{2k}$ higher than the second term. Hence, we lower bound by considering one scenario where the first term is non-zero.

Continuing, we can further split the indicator function into cases when each term in the argument of the indicator function is positive.

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{x},y}\sigma\left(\frac{1}{2k}\sum_{j=1}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\right)x_1\\
&\quad\geq\frac{1}{2k}\mathbb{E}_{\mathbf{x},y}\mathbb{I}\left(\frac{1}{2k}\sum_{j=2}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+b_i\geq 0\right)\\
&\quad\geq\frac{1}{2k}\mathbb{E}_{\mathbf{x},y}\mathbb{I}\left(\sum_{j=2}^{k}x_j\geq 0\right)\mathbb{I}\left(\sum_{j=k+1}^{d}x_j\geq 0\right)\mathbb{I}\left(b_i\geq 0\right)\\
&\quad\geq\frac{1}{8k}\mathbb{I}\left(b_i\geq 0\right).
\end{aligned}
$$

From Equation (4.96), we then have

$$
\mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_1\geq\frac{1}{4k}\mathbb{I}\left(b_i\geq 0\right)+\mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1}).
$$

As b_i has been kept at random initialization and thus is a random variable selected from the set $\{-1+\frac{1}{k},\cdots,1-\frac{1}{k}\}$, with probability $\frac{1}{2}$, $\mathbb{I}\left(b_i\geq 0\right)$. This implies, w.p. atleast $1/2$ w.r.t. a neuron's bias initialization, $\mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_1\geq\frac{1}{4k}+\mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1})$. The final bound comes from the fact that $\mathbb{E}_{\mathbf{x},y}f_{\mathcal{T}}(\mathbf{x})x_1=\mathbb{E}_{\mathbf{x},y}\frac{1}{m}\sum_{i=1}^{m}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_1\geq\frac{1}{8k}+\mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1})+\mathcal{O}(m^{-1/2})$, where the error term is bounded using Hoeffding's inequality.

2. Out-of-support correlations: Similar to the Equation (4.96), we have for $u\notin S$,

$$
\begin{aligned}
\mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_u=\mathbb{E}_{\mathbf{x},y}&\sigma\left(\frac{1}{2k}\sum_{j=1}^{k}x_j+\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+\langle\boldsymbol{v}_i,\mathbf{x}\rangle+b_i\right)x_u\\
&-\sigma\left(-\frac{1}{2k}\sum_{j=1}^{k}x_j-\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\text{sign}(w_{ij}^{(0)})x_j+\langle\boldsymbol{v}_i,\mathbf{x}\rangle+b_i\right)x_u.
\end{aligned}
\tag{4.97}
$$

However, we observe that the influence of x_u in each of the terms is bounded by $\frac{1}{k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}$. Consider the first term; the argument for the second term is similar. We can again ignore $\langle \boldsymbol{v}_i, \mathbf{x}\rangle$ incurring an error of $\mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1})$.

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x},y}\sigma\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i\right)x_u \\
&\quad= \mathbb{E}_{\mathbf{x},y:x_u=+1}\sigma\left(\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\operatorname{sign}(w_{iu}^{(0)}) + \frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1\to d;j\neq u}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i\right) \\
&\quad- \mathbb{E}_{\mathbf{x},y:x_u=-1}\sigma\left(-\frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\operatorname{sign}(w_{iu}^{(0)}) + \frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1\to d;j\neq u}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i\right) \\
&\quad= \mathbb{E}_{\mathbf{x},y}\frac{C(\mathbf{x})}{k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\operatorname{sign}(w_{iu}^{(0)})\mathbb{I}\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1\to d;j\neq u}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i \geq 0\right),
\end{aligned}$$

where $C(\mathbf{x}) \in \{1,2\}$ denotes a function that depends on $\mathbf{x}$. The final step follows from a first order taylor expansion of σ. The magnitude can hence be bounded by $\frac{1}{k}\frac{|\zeta_{k+1}|}{|\zeta_{k-1}|}$. This can be bounded by $\frac{1}{kd}$ (section 5.3, O'Donnell [2014]). The final bound comes from the fact that $\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}(\mathbf{x})x_u = \mathbb{E}_{\mathbf{x},y}\frac{1}{m}\sum_{i=1}^{m}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})x_u \leq \mathcal{O}((kd)^{-1})$.

3. Correlations to support of an even size: The function $(f_{\mathcal{T}}^{(1)})_i$ is given by

$$\begin{aligned}
(f_{\mathcal{T}}^{(1)})_i(\mathbf{x}) =& \sigma\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right) \\
&- \sigma\left(-\frac{1}{2k}\sum_{j=1}^{k} x_j - \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + \langle \boldsymbol{v}_i, \mathbf{x}\rangle + b_i\right) \\
=& \sigma\left(\frac{1}{2k}\sum_{j=1}^{k} x_j + \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i\right) \\
&- \sigma\left(-\frac{1}{2k}\sum_{j=1}^{k} x_j - \frac{1}{2k}\frac{\zeta_{k+1}}{|\zeta_{k-1}|}\sum_{j=k+1}^{d}\operatorname{sign}(w_{ij}^{(0)})x_j + b_i\right) + \mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1}) \\
:=& g(\mathbf{x}) + \mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1}).
\end{aligned}$$

One can observe that $g(\mathbf{x})$ is a symmetric function and so an odd function. Thus, $\mathbb{E}_{\mathbf{x},y} g(\mathbf{x})\chi_S(\mathbf{x}) = 0$ (exercise 1.8, O'Donnell [2014]) and so, $\mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})\chi_S(\mathbf{x}) = \mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1})$.

4. Output norm: Focusing on function $(f_{f_{\mathcal{T}}}^{(1)})_i$:

$$\begin{aligned}
\left\|(f_{\mathcal{T}}^{(1)})_i\right\|_2^2 &= \mathbb{E}_{\mathbf{x},y}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})^2 \\
&= \mathbb{E}_{\mathbf{x},y}\left(\sigma(\langle w_{ij}^{(1)}, \mathbf{x}\rangle + b_i) - \sigma(\langle w_{i+m/2,j}^{(1)}, \mathbf{x}\rangle + b_i)\right)^2 \\
&\leq \mathbb{E}_{\mathbf{x},y}\min\left(\left\|\boldsymbol{w}_i^{(1)}\right\|_2^2 + b_i^2, \left\|\boldsymbol{w}_{i+m/2}^{(1)}\right\|_2^2 + b_i^2\right)\|\mathbf{x}\|_2^2 = \mathcal{O}\left(\frac{1}{k}\right)\cdot d.
\end{aligned}$$

The intermediate step uses Cauchy-Schwartz inequality, and the final step uses the values of $w_{ij}^{(1)}, w_{i+m/2,j}^{(1)}$. As $f_{\mathcal{T}}^{(1)}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m/2}(f_{\mathcal{T}}^{(1)})_i(\mathbf{x})$, we have $\left\|(f_{\mathcal{T}}^{(1)})\right\|_2^2 \leq \frac{2}{m}\sum_{i=1}^{m/2}\left\|(f_{\mathcal{T}}^{(1)})_i\right\|_2^2 = \mathcal{O}\left(\frac{d}{k}\right)$. □

Corollary 3. *Under the event that the conditions in Lemma 63 are satisfied by each neuron, which occurs with probability at least $1-\delta$ w.r.t. the randomness of initialization and sampling, the output of the model after the first phase can be given as:*

$$f_{\mathcal{T}}^{(1)}(\mathbf{x}) = \sum_{j=1}^{k} c_j x_j + \sum_{j=k+1}^{d} c_j x_j + \sum_{S\subseteq[d]:|S|\%2=1,|S|\geq 3} c_S \chi_S(\mathbf{x}) + \sum_{S\subseteq[d]:|S|\%2=0} c_S\chi_S(\mathbf{x}),$$

where

$$\begin{aligned}
|c_j| &\geq \Omega(k^{-1}), && \text{for all } 1\leq j\leq k,\\
|c_j| &\leq \mathcal{O}((kd)^{-1}), && \text{for all } j>k,\\
|c_S| &\leq \mathcal{O}(\tau_g d\,|\zeta_{k-1}|^{-1}), && \text{for all } S\subseteq[d] \text{ with } |S|\%2=0,\\
|c_S| &\leq \mathcal{O}(d/k), && \text{for all } S\subseteq[d] \text{ with } |S|\%2=1.
\end{aligned}$$

As such, the following correlations hold true for all i.

$$\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x})\cdot Maj(\mathbf{x})x_i = \frac{1}{2}c_i + \mathcal{O}(\tau_g d^{5/3}\,|\zeta_{k-1}|^{-1}).$$

If batch size B_1 is set $\geq \Omega(k^2 d^{10/3}\zeta_{k-1}^{-2})$, such that $\tau_g \leq \mathcal{O}(k^{-1}d^{-5/3}\,|\zeta_{k-1}|)$, then the following holds for all i.

$$\begin{aligned}
\left|\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x})\cdot Maj(\mathbf{x})x_i\right| &\geq \Omega(k^{-1}), && \text{if } i\in[k],\\
\left|\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x})\cdot Maj(\mathbf{x})x_i\right| &\leq o(k^{-1}), && \text{if } i\notin[k],
\end{aligned}$$

Proof. The form of $f_{\mathcal{T}}^{(1)}$ follows from the fourier coefficient analysis in Lemma 64.

Now, we can use the formulation to derive

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x})\cdot \mathrm{Maj}(\mathbf{x})x_i\\
=&\mathbb{E}_{\mathbf{x},y}\sum_{j=1}^{d} c_j x_j\cdot\mathrm{Maj}(\mathbf{x})\cdot x_i + \mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=1,|S|\geq3} c_S \mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})\cdot x_i\\
&+\mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\chi_S(\mathbf{x})\cdot\mathrm{Maj}(\mathbf{x})x_i\\
=&\mathbb{E}_{\mathbf{x},y}\sum_{j=1}^{d} c_j x_j\cdot\mathrm{Maj}(\mathbf{x})\cdot x_i + \mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})\cdot x_j\\
=&c_i\mathbb{E}_{\mathbf{x},y}\mathrm{Maj}(\mathbf{x}) + \mathbb{E}_{\mathbf{x},y}\sum_{j,j\neq k} c_j\mathrm{Maj}(\mathbf{x})x_j x_i + \mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})\cdot x_i\\
=&\frac{1}{2}c_i + \mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})\cdot x_i.
\end{aligned}$$

The second step removes $\mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\chi_S(\mathbf{x})\cdot\mathrm{Maj}(\mathbf{x})x_i$ because $\mathrm{Maj}(\mathbf{x})$ is an odd function, and so $\mathbb{E}_{\mathbf{x},y}\mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})x_i$ will be 0 for odd sized S. Similar argument holds for removing $\mathbb{E}_{\mathbf{x},y}\sum_{j,j\neq i} c_j\mathrm{Maj}(\mathbf{x})x_j x_i$ in the final step. We finish the proof by bounding $\mathbb{E}_{\mathbf{x},y}\sum_{S\subseteq[d]:|S|\%2=0} c_S\mathrm{Maj}(\mathbf{x})\chi_S(\mathbf{x})\cdot x_i$.

As $|c_S| \le \mathcal{O}(\tau_g d \, |\zeta_{k-1}|^{-1})$ for all S with $|S|\%2 = 0$, we can bound it as

$$\begin{aligned}
&\left|\mathbb{E}_{\mathbf{x},y} \sum_{S\subseteq[d]:|S|\%2=0} c_S \text{Maj}(\mathbf{x})\chi_S(\mathbf{x}) \cdot x_i\right| \\
\le& \mathcal{O}(\tau_g d \, |\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subseteq[d]:|S|\%2=0} \left|\mathbb{E}_{\mathbf{x},y} \text{Maj}(\mathbf{x})\chi_S(\mathbf{x}) x_i\right|\right) \\
\le& \mathcal{O}(\tau_g d \, |\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subseteq[d]} \left|\mathbb{E}_{\mathbf{x},y} \text{Maj}(\mathbf{x})\chi_S(\mathbf{x})\right|\right) \\
\le& \mathcal{O}(\tau_g d \, |\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subseteq[d]} \left|\mathbb{E}_{\mathbf{x},y} \text{Maj}(\mathbf{x})\chi_S(\mathbf{x})\right|\right) \\
=& \mathcal{O}(\tau_g d \, |\zeta_{k-1}|^{-1}) \cdot \sum_{S\subseteq[d]} \Theta\left(\frac{|S|^{-1/3}}{\binom{d}{|S|}}\right) \\
=& \mathcal{O}(\tau_g d^{5/3} \, |\zeta_{k-1}|^{-1}).
\end{aligned}$$

Here the pre-final step follows from the bounds on the Fourier coefficients of Maj outlined in Section 4.2.6. Finally, we set $B_1 \ge \Omega(\tau_g^{-2})$ is set such that $\tau_g \le \mathcal{O}(k^{-1} d^{-5/3} \zeta_{k-1})$. This makes $\mathcal{O}(\tau_g d^{5/3} \, |\zeta_{k-1}|^{-1}) = o(1/k)$. Hence, with appropriate batch size B_1,

$$\mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(1)}(\mathbf{x}) \cdot \text{Maj}(\mathbf{x}) x_i = \frac{1}{2} c_i + o(1/k).$$

The proof follows from the magnitude of c_i derived above. □

4.2.6.1.2 Second stage analysis for the teacher

Lemma 65 (Second stage Training, cf. Theorem 4 in [Barak et al., 2022b]). *Fix $\epsilon, \delta > 0$. Suppose $m \ge \Omega(2^k k \log(k/\delta))$, $d \ge \Omega\left(k^4 \log(kd/\epsilon)\right)$. Furthermore, suppose $B_1 \ge \Omega(|\zeta_{k-1}|^2 k^2 \log(kd/\epsilon))$ s.t. the weights satisfy the conditions in Lemma 63 with $\tau_g = \mathcal{O}(|\zeta_{k-1}| \, k^{-1})$ after the first phase. Then after $T_2 = \Omega(md^2k^3/\epsilon^2)$ steps of training with batch size $B_2 = 1$, learning rate $\eta_2 = 4k^{1.5}/(d\sqrt{m(T_2-1)})$ and decay $\lambda_2 = 0$, we have with expectation over the randomness of the initialization and the sampling of the batches:*

$$\min_{t\in[T_2]} \mathbb{E}\left[L_{\theta^{(t)}}(\mathbf{x}, y)\right] \le \epsilon.$$

Thus, the minimal sample complexity to reach a loss of ϵ is given by

$$\begin{aligned}
T_1 \times B_1 + T_2 \times B_2 &= \Theta(|\zeta_{k-1}|^2 k^2 \log(kd/\epsilon)) + \Theta(md^2k^3/\epsilon^2) \\
&= \Theta(d^{k-1} k^2 \log(dk/\epsilon) + 2^k d^2 k^4 \epsilon^{-2} \log(k/\delta)).
\end{aligned}$$

Corollary 4. *Under the conditions outlined in Lemma 65, after T_2 steps of training in the second phase, if $t^\dagger$ denote the time step at which the model achieves the minimum loss, i.e. $t^\dagger := \arg\min_{t\in[T_2]} \mathbb{E}\left[L_{\theta^{(t)}}(\mathbf{x}, y)\right]$, then*

$$\mathbb{E}\left[f_{\mathcal{T}}^{(t^\dagger)}(\mathbf{x}) x_i\right] \le \epsilon, \text{ for all } i \in [d].$$

The proof follows from the fact that if the correlation along $y = \prod_{i \in S} x_i$ is large ($\geq 1 - \epsilon$ as hinge loss is below ϵ), the correlations along other Fourier basis functions will be small. Hence, depending on how saturated the model is, the signal along the support elements are small.

We will use a slightly modified version of Lemma 65 with higher sample complexity in the first phase, to ensure the stronger conditions of Corollary 3 hold true as well. This will be necessary to get improved signal to teach a smaller student.[20]

Corollary 5 (Modified Version of Lemma 65)**.** *Fix $\epsilon, \delta > 0$. Suppose $m \geq \Omega(2^k k \log(k/\delta))$, $d \geq \Omega\left(k^4 \log(kd/\epsilon)\right)$. Furthermore, suppose $B_1 \geq \Omega(|\zeta_{k-1}|^2 k^2 d^{10/3} \log(kd/\epsilon))$ s.t. the weights satisfy the conditions in Corollary 3 with $\tau_g = \mathcal{O}(|\zeta_{k-1}| k^{-1} d^{-5/3})$ after the first phase. Then after $T_2 = \Omega(md^2k^3/\epsilon^2)$ steps of training with batch size $B_2 = 1$, learning rate $\eta_2 = 4k^{1.5}/(d\sqrt{m(T_2-1)})$ and decay $\lambda_2 = 0$, we have with expectation over the randomness of the initialization and the sampling of the batches:*

$$\min_{t \in [T_2]} \mathbb{E}\left[L_{\theta^{(t)}}(\mathbf{x}, y)\right] \leq \epsilon.$$

Thus, the minimal sample complexity to reach a loss of ϵ is given by

$$\begin{aligned} T_1 \times B_1 + T_2 \times B_2 &= \Theta(|\zeta_{k-1}|^2 d^{10/3} k^2 \log(kd/\epsilon)) + \Theta(md^2k^3/\epsilon^2) \\ &= \Theta(d^{k+7/3} k^2 \log(dk/\epsilon) + 2^k d^2 k^4 \epsilon^{-2} \log(k/\delta)). \end{aligned}$$

4.2.6.2 Analysis for the student

Proof of Theorem 23. We will first prove the sample complexity upper bound for progressive distillation, followed by a sample complexity lower bound for distillation.

Sample complexity for Progressive distillation: Under progressive distillation, the label is given by $f_{\mathcal{T}}^{(T_1)}$ for the first T_1 steps. We will follow similar steps as Lemma 63, where the label is replaced by $f_{\mathcal{T}}^{(T_1)}$. Claim 7 changes, while Claim 8 stays the same. We will showcase the change in Claim 7 here.

At initialization, the population gradient of the weight vector in neuron i at coordinate j is given by

$$\begin{aligned} &\mathbb{E}_{\mathbf{x},y} \nabla_{\tilde{w}_{ij}^{(0)}} \ell_{\mathrm{D}}(\mathbf{x}, y; f_{\mathcal{S}}^{(0)}, f_{\mathcal{T}}) \\ &= -\mathbb{E}_{\mathbf{x},y} \nabla_{\tilde{w}_{ij}^{(0)}} f_{\mathcal{S}}^{(0)}(\mathbf{x}) f_{\mathcal{T}}^{(T_1)}(\mathbf{x}) \\ &= -a_i \mathbb{E}_{\mathbf{x},y} \mathbb{I}\left[\langle \tilde{\boldsymbol{w}}_i^{(0)}, \mathbf{x}\rangle + \tilde{b}_i \geq 0\right] f_{\mathcal{T}}^{(T_1)}(\mathbf{x}) x_j \\ &= -a_i \mathbb{E}_{\mathbf{x},y}\left(\frac{1}{2} + \frac{1}{2}\mathrm{Maj}(\tilde{\boldsymbol{w}}_i^{(0)}, \mathbf{x})\right) f_{\mathcal{T}}^{(T_1)}(\mathbf{x}) x_j \\ &= -a_i \frac{1}{2} \mathbb{E}_{\mathbf{x},y} f_{\mathcal{T}}^{(T_1)}(\mathbf{x}) x_j - a_i \frac{1}{2} \mathbb{E}_{\mathbf{x},y} \mathrm{Maj}(\tilde{\boldsymbol{w}}_i^{(0)}, \mathbf{x}) f_{\mathcal{T}}^{(T_1)}(\mathbf{x}) x_j, \end{aligned}$$

where the relation between $\mathbb{I}\left[\langle \tilde{\boldsymbol{w}}_i^{(0)}, \mathbf{x}\rangle + \tilde{b}_i \geq 0\right]$ and $\mathrm{Maj}(\tilde{\boldsymbol{w}}_i^{(0)}, \mathbf{x})$ follows because of $\left|\tilde{b}_i\right| < 1$ at initialization. From Corollary 3,

$$\begin{aligned} \left|\mathbb{E}_{\mathbf{x},y} \nabla_{\tilde{w}_{ij}^{(0)}} \ell_{\mathrm{D}}(\mathbf{x}, y; f_{\mathcal{S}}^{(0)}, f_{\mathcal{T}})\right| &\geq \Omega(k^{-1}), \quad \text{if } j \in [k], \\ \left|\mathbb{E}_{\mathbf{x},y} \nabla_{\tilde{w}_{ij}^{(0)}} \ell_{\mathrm{D}}(\mathbf{x}, y; f_{\mathcal{S}}^{(0)}, f_{\mathcal{T}})\right| &\leq o(k^{-1}), \quad \text{if } j \notin [k]. \end{aligned}$$

[20] We haven't optimized the error bounds in Corollary 3. Our sample complexity bounds are likely loose in Corollary 5

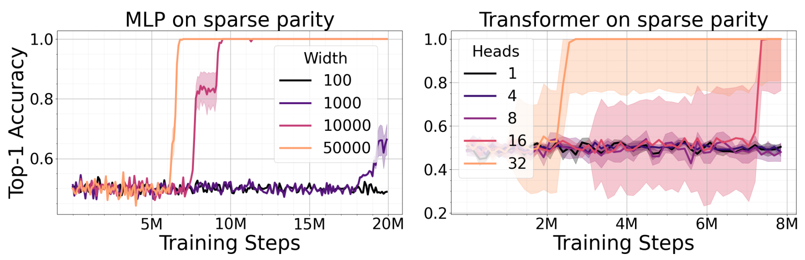


Figure 4.10: **Larger models learn sparse parity faster.** A larger model has more width (MLP, *left*) or more attention heads (Transformers, *right*). The results are for $(100, 6)$-parity, aggregated over 5 runs for each setup.

Thus, a fourier gap exists between the population gradients on in-support and out-of-support coordinates in the gradients. We can then apply Claim 8 to show that a finite batch size of $B_1 \geq \Omega(k^2 \log(d\tilde{m}/\delta))$ is sufficient to maintain this gap between the coordinates in support and out of support. Thus, the change in the necessary sample complexity comes from the reduced sample complexity in the first phase. The proof for the second phase training is exactly equal to the proof for the teacher in Theorem 23.

Sample complexity for Distillation: On the other hand, for the teacher checkpoint with loss $\mathcal{O}(d^{-c})$, the correlation to the monomial terms in the support is bounded by $\mathcal{O}(d^{(-c)})$ (by Corollary 4). If we want to learn from the correlations to the support, we need the number of samples to be at least $\Omega(d^{2c})$ as the gradient noise needs to be lower than $\mathcal{O}(d^{-c})$ (by Claim 8). To learn the support from the true label, we need the number of samples to be at least $\Omega(d^{k-1})$, by the following result:

Lemma 66 (Width-optimization trade-off, cf. Proposition 3 in [Edelman et al., 2023]). *For $\delta > 0$, gradient noise $\tau_g > 0$, and model width $m > 0$, if $T \leq \frac{1}{2}\binom{d}{k}\frac{\delta\tau_g^2}{m}$, then there exists a (d, k)-sparse parity such that w.p. at least $1 - \delta$ over the randomness of initialization and samples, the loss is lower bounded as $L(f_T^{(t)}) \geq 1 - \tau_g$ for all $t \in \{1 \cdots T\}$.*

This result implies that for a fixed batch size (and hence a fixed τ_g), we either require a bigger width, or more number of gradient steps (which translates to sample complexity since we are using fresh samples each batch). Hence, for the model to learn the support from a combination of the two components, it needs a sample complexity at least $\Omega(d^{\min(2c,k-1)}/\tilde{m})$. $\square$

4.2.7 Results on sparse parity and its generalization

4.2.7.1 Additional results on sparse parity with MLP

We take both the teacher and student models to be 1-hidden-layer MLPs with ReLU activations. The teacher has a hidden width of 5×10^4, and the students are of widths 10^2 or 10^3. All models are trained using SGD with batch size 1 for $20M$ steps on sparse parity data with $n = 100$ and $k = 6$ (Definition 23). The support is set to be the first 6 coordinates of the input vector without loss of generality. The learning rate is searched over $\{10^{-2}, 5 \times 10^{-3}, 10^{-3}\}$. Evaluation is based on a held-out set consisting of 4096 examples, and we report the average across 3 different training seeds. For one-shot distillation, we use the teacher checkpoint at the end of training (20M checkpoint), at which point the teacher has fully saturated. For progressive distillation, we use $N = 200$ equally spaced teacher checkpoints that are 0.1M steps apart.

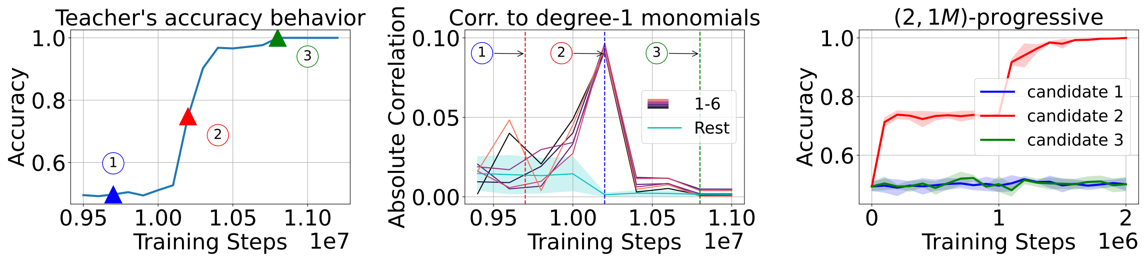


Figure 4.11: Repeated experiments from Figure 4.5 for a student of width 1000.

4.2.7.2 Learning with Transformers: parallel search with attention heads

The benefit of progressive distillation and the implicit curriculum is not specific to MLP. This section presents similar results with Transformers [Vaswani et al., 2017]. The d-dimensional input vector is now treated as a length-d sequence, and the label is predicted using the last token's output. We fix the support S to be the first 6 coordinates of the sequence. Note that unlike MLP, Transformer's learning is not permutation-invariant to the location of S due to the causal mask. Nevertheless, given the same S, the comparison on learning speed is still meaningful.

For Transformers, the parallel queries come from both the MLP width and also the *number of attention heads*. To illustrate this, consider the following two solutions (which we formalize in Section 4.2.7.2.1) to sparse parity: The first solution uses attention to locate the support and then uses MLP to compute the product of the in-support variables. The second solution copies over all variables to the final position, whose MLP is then responsible for both identifying the support and computing the product. The second solution is less interesting as it reduces to an MLP, so we focus on the first solution in the following, which utilizes the attention mechanism unique to Transformers.

(R6) More attention heads helps with the search for support Our experiments are based on 2-layer Transformers [21] with 8 dimensions per attention head. As shown in Figure 4.10 (right), increasing the number of heads makes learning faster. There are clear phase transitions similar to the MLP case.

Ablation with other ways to vary the model size Most Transformer experiments in this work keep the per-head dimension to be fixed and vary the number of attention heads between the teacher and the student. The MLP input dimension is the sum of the attention head dimensions, so a student with fewer heads will have a smaller MLP than the teacher, which is preferable in terms of efficiency. Fixing the per-head dimension is a widely adopted setup in practice, such as in the Llama series [Touvron et al., 2023]. We now additionally consider two other ways to vary the model size. In particular, we vary the number of heads, while 1) fixing the hidden dimension (i.e. the total dimension of all

[21] We use 2 layers since 1-layer Transformers are hard to train empirically, despite being representationally sufficient to solve sparse parity.

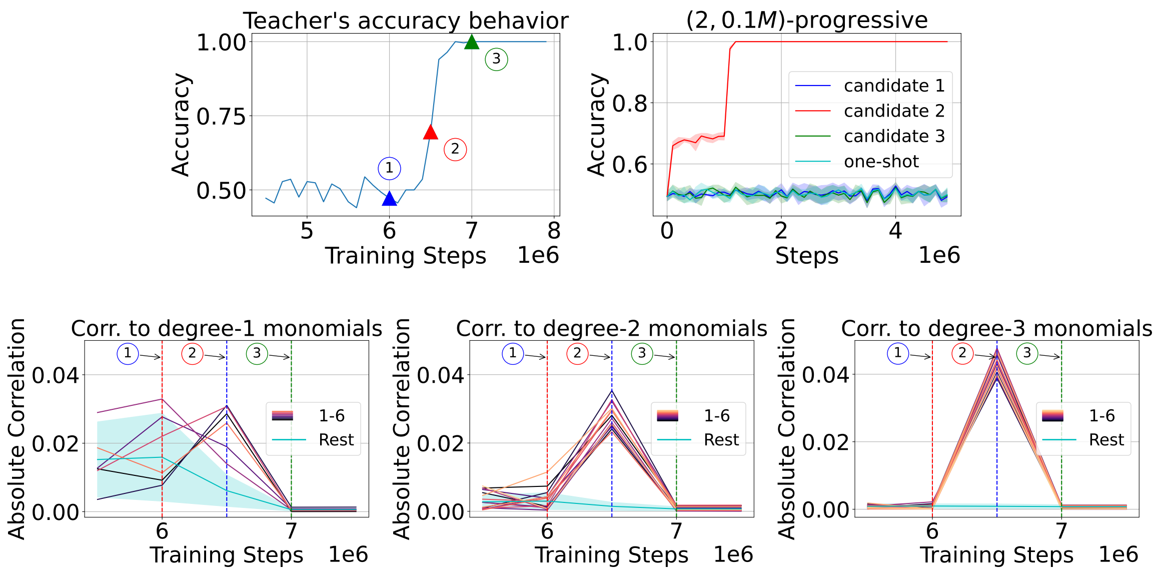
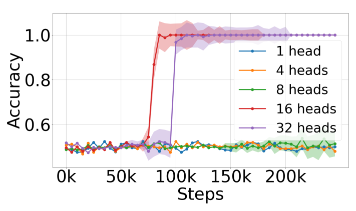
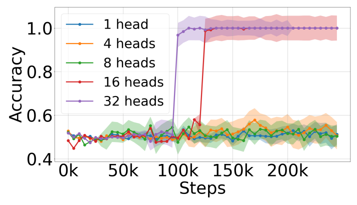


Figure 4.12: Repeated experiments from Figure 4.5 but for a different teacher. For $(2, 0.1M)$ progressive distillation, the checkpoint that lies in the middle of the second phase accelerates training the most. In Figure 4.5, we used a teacher that was trained with learning rate 5×10^{-3}. The correlation plot for the teacher to degree-1 monomials had a clear gap for degree-1 monomials in-support and out-of-support at the middle of the second phase (indicated by candidate 2). However, for a teacher that is trained with a higher learning rate 10^{-2}, we didn't find such a clean gap in correlations for degree-1 monomials. On the other hand, correlations to degree-2 and degree-3 monomials showed a clean gap between in-support and off-support variables at the middle of the phase transition. Hence, the student needn't learn only from degree-1 monomials to get training acceleration, any low degree monomials suffice to teach the student about the support. Rest for degree-2 monomials refers to all monomials of the form $x_i x_j$ where atleast one of $i, j \notin S$. Similar definition for degree-3 monomials.

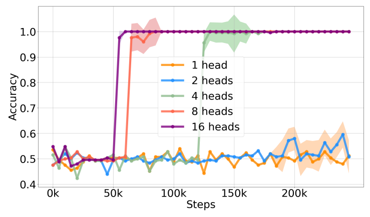
heads concatenated) to be 256, or 2) fixing the dimension of each head to be 256 and averaging the output from each head, in which case the hidden dimension is also 256. These two setups are less common in practice but nevertheless serves as complementary evidence: the performance difference comes solely from the number of attention heads, as the MLP dimension is kept the same. As shown in Figure 4.13 (b,c), increasing the number of attention also increases the training speed in these two setups.



(a) Per-head dimension = 8

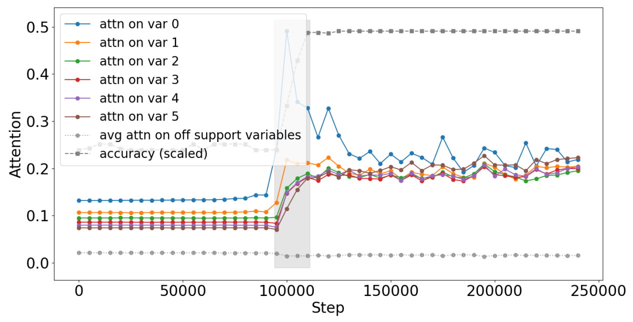


(b) Hidden dimension = 256



(c) Dimension 256, averaging heads

Figure 4.13: Increasing the number of attention heads speeds up training. Each plot compares the accuracy throughout training for 2-layer models with various heads, while fixing: (a) the per-head dimension to 8; (b) the MLP hidden dimension to 256; (c) both the per-head and MLP hidden dimension to 256, by averaging (rather than concatenating) the heads. We report runs with the learning rate that has the highest mean accuracy and break tie with training speeds. The shadows show the variances of the runs.



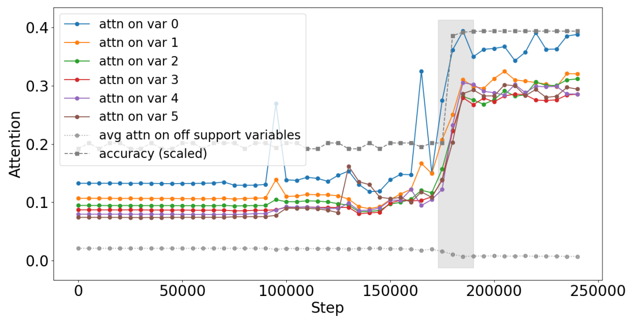


Figure 4.14: **In-support attention growth co-occurs with accuracy increase** Attention on individual coordinates on or off the support of the sparse parity, taking the median of 1024 random binary input sequences. The shade highlights the teacher's phase transition period. The model accuracy is marked by the gray dashed line, with scale adjusted for better display. The two subfigures show the same type of results but with different randomness seeds.

Ablation with 2-shot distillation We repeat the 2-shot distillation ablation for MLP. We first confirm that the low-degree curriculum described in Section 4.2.2.1 is also observed in Transformers. As shown in Figure 4.15, the 2-layer 32-head teacher model exhibits significantly higher correlation with the in-support monomials (i.e. $\{x_i\}_{i \in S}$ than with off-support monomials during the phase transition. [22] Then, we show in Figure 4.16 that using as few as 1 intermediate checkpoint suffices to significantly

[22]Note that the upper right subplot in Figure 4.15 has a second correlation spike with the in-support variables. However, supervising with this second checkpoint does not provide acceleration. This suggests that there might be mechanisms other

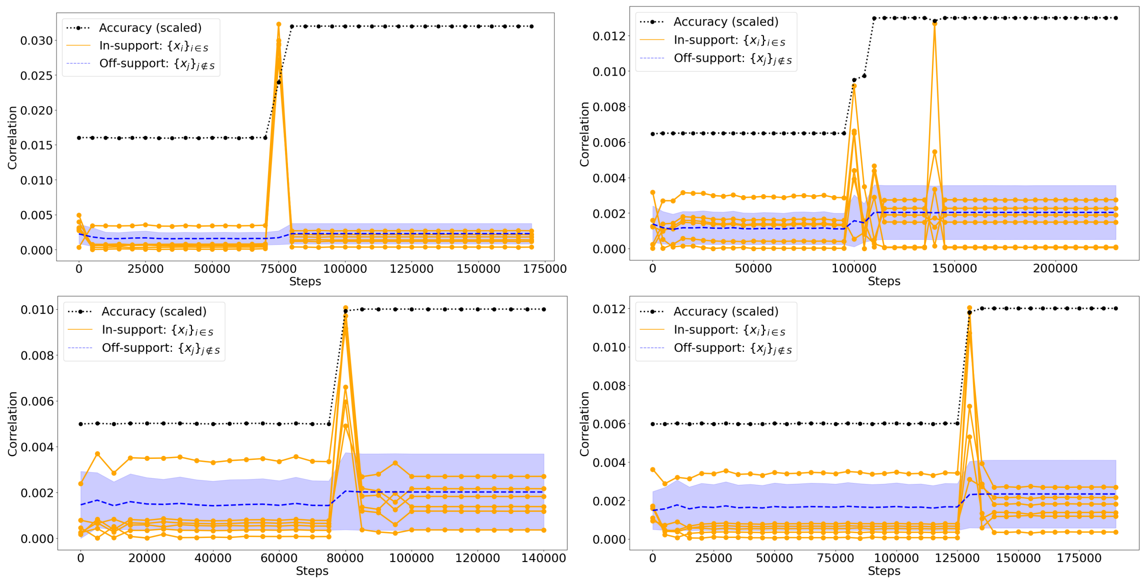


Figure 4.15: **Low-degree curriculum in Transformers on (100, 6)-sparse parity.** The x-axis shows the training steps, and y-axis shows the 2-layer 32-head teacher's correlation with in-support (orange lines) vs off-support (blue lines, aggregated into mean and standard deviation) degree-1 monomials. The black dotted lines mark the accuracy, scaled for better display. The correlation values are calculated using 100k randomly drawn sequences. The 4 subplots correspond to models trained using 4 random seeds.

speeds up the training of the student.

Ablation with various temperatures As mentioned in Section 4.2.1, our progressive distillation results use a low temperature in order to remove potential favorable regularization effects from soft labels. We chose a temperature of $\tau = 10^{-4}$ for sparse parity, where the output dimension is 2. In Figure 4.17, we empirically confirm that setting the temperature to be below 0.01 is sufficient to get results that are qualitatively similar to using $\tau = 0$ (i.e. taking the argmax). Note that using a higher temperature such as $\tau = 1$ can make learning slower despite potentially having more regularization effects from softer labels. We leave understanding the exact effect of temperature to future work.

4.2.7.2.1 Two Transformer solutions for sparse parity (Proposition 9 and Proposition 10)

We consider a simplified version of a Transformer block, without the residual connection or the layernorm:

$$f_{\text{block}} = f_{\text{mlp}}^{(L)} \circ f_{\text{attn}},$$

where

$$f_{\text{attn}}(X; W_Q, W_K, W_V) := \mathsf{CausalAttn}(X W_Q W_K^\top X^\top) X W_V,$$

with W_Q, W_K, W_V being the query, key, value matrices, and $f_{\text{mlp}}^{(L)}(x; \{W_l, b_l\}_{l\in[L]})$ is a L-layer MLP that recursively apply $f_{\text{mlp}}^{(l+1)}(x) = \sigma(W_{l+1} f_{\text{mlp}}^{(l)}(x) + b_{l+1})$ position-wise. σ is the relu function for

than the low-degree curriculum at play.

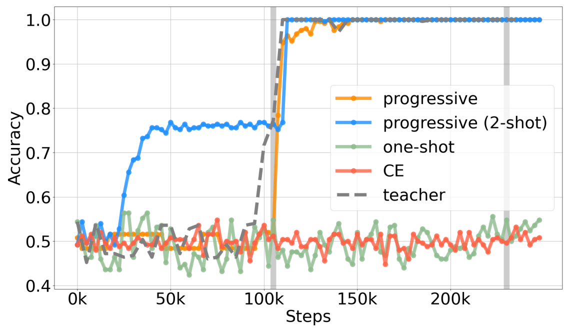


Figure 4.16: **2-shot progressive distillation with Transformers**: Compared to cross-entropy training or one-shot distillation, transformers learn faster with progressive distillation, where the intermediate checkpoints are taken either at regular 10k intervals ("progressive"), or during the phase transition ("progressive (2-shot)"). The two vertical lines show the teacher training steps at which the two checkpoints for 2-shot distillation are chosen. We set the teacher temperature to be $\tau = 10^{-4}$ for progressive distillation, and $\tau = 1$ for one-shot distillation.

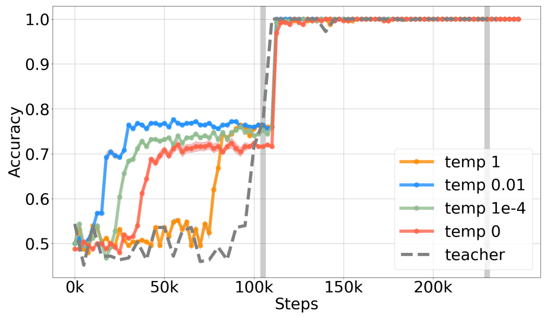


Figure 4.17: **The benefit of progressive distillation holds with hard labels**, as shown by comparing 2-shot progressive distillation with different temperatures. The two gray vertical lines mark the training steps at which the teacher checkpoints are taken.

$l \in [L-1]$, and is the identity function for $l = L$.

Proposition 9 (Attention support selection)**.** *(d,k)-sparse parity can be solved by a 1-layer Transformer with a 2-layer MLP, whose attention weights satisfy $\alpha_{i,d} \propto \exp(c\mathbb{1}[i \in S])$ for some large constant $c > 0$. The MLP has hidden dimension $4(k+1)$, L_∞ norm bounded by $4k(k+1)$.*

Proof. The idea is that the attention selects the k in-support variables, and the MLP computes the product of these variables.

To select the in-support variables, we want the attention weight $\alpha_{i,d} \propto \exp(c\mathbb{1}[i \in S])$, for some large constant c. This can be achieved by having the projection matrices W_Q, W_K focus only on the position and ignore the tokens. In particular, let z denote the input sequence, and let the embedding of a token be $\boldsymbol{x}_i = \boldsymbol{v}_{z_i} + p_i$, where $\{\boldsymbol{v}_0, \boldsymbol{v}_1\}$ are embeddings for the binary token 0 or 1, and p_i is the position encoding for position i. Take $\{\boldsymbol{v}_0, \boldsymbol{v}_1\}$, $\{p_i\}_i$ such that $\boldsymbol{v}_{z_i} \perp p_i$. Let $c > 0$ be a large enough constant. Choose W_Q, W_K such that for any $i \in [d]$, $\boldsymbol{x}_i^\top W_Q^\top W_K \boldsymbol{x}_d = p_i^\top W_Q^\top W_K p_d = c \cdot \mathbb{1}[i \in S]$. This ensures that $\alpha_{i,d} \propto \exp(c\mathbb{1}[i \in S])$.

Then, the role of attention is to average over the in-support tokens. For simplicity of exposition, let's take $c \to \infty$ for now (i.e. using saturated attention [Merrill et al., 2022]), so that $\alpha_{i,n} \to \frac{\mathbb{1}[i \in S]}{k}$; that is, the attention weights at the last position average over the in-support variables. Take W_V to be a vector, such that W_V ignores the positional information and the input token 0, and only preserves the input token 1, i.e. $W_V p_i = 0, \forall i \in [d]$, $W_V \boldsymbol{v}_0 = 0$, and $W_V \boldsymbol{v}_1 = 1$.

Next, the MLP needs to compute the parity function over the k in-support variables. The input to the MLP is hence proportional to $(\sum_{i \in S} \mathbb{I}[z_i = 1])\boldsymbol{v}_1$, and the size of the set of inputs is $k+1$. To determine the size of the MLP, we use Lemma 10 from Chapter 3 on 1D discrete function interpolation. Setting $B_x = 1$, $B_y = 1$, and $\Delta = \frac{1}{|S|}$, the parity function over these $k+1$ input values can be approximated by a 2-layer MLP with inner dimension $4(k+1)$, with norm bounded by $4k(k+1)$.

As a concrete example, one way to satisfy the requirements above is to set the attention weights to $\boldsymbol{v}_1 = W_V = \boldsymbol{e}_1 := [1,0,0,0]$, $\boldsymbol{v}_0 = \boldsymbol{e}_2 := [0,1,0,0]$. Set $p_i = p_n = \boldsymbol{e}_3$ for $i \in S$, and $p_i = \boldsymbol{e}_4$ for $i \notin S$.

Set $W_Q = W_K = c \begin{bmatrix} 0 & 0 & 1 & 0 \\ \\ 0 & 0 & 0 & 0 \end{bmatrix}$ for some sufficiently large $c > 0$.

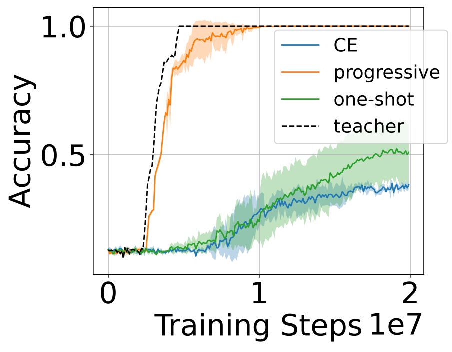
□

Proposition 10 (No attention selection)**.** *There exists a 1-layer Transformer with 3-layer MLP that computes k-sparse parity, whose attention weights satisfy $\alpha_{i,d} = \frac{1}{d}$. Consequently, the MLP computes the sparse parity function given the full set of variables.*

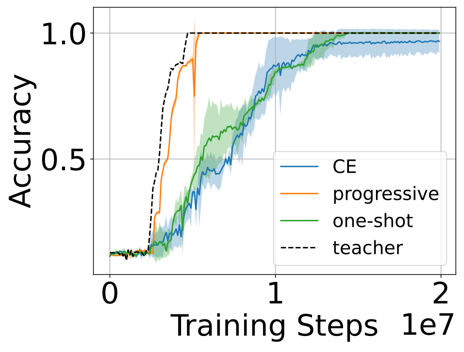
Proof. The idea is for the uniform attention to copy all tokens to the last position. However, unlike in Proposition 9, the attention needs to copy the tokens into a length-d embedding vector, as we need to preserve the position information in this embedding vector. We need to generalize Lemma 10 accordingly to handle multi-dimensional inputs, i.e. using Lemma 11. Then, the MLP at the last position computes the sparse parity over the k coordinates while ignoring the others. Hence the effective input set is $|\mathcal{X}| = 2^k$. Setting $B_x = 1$, $B_y = 1$, and $\Delta = 1$, there exists a 3-layer MLP with width 2^k and norm bound 2^{k+2} by Lemma 11. □

Preliminary interpretability analysis: Transformer does utilize attention in practice We observe that the model focuses attention on relevant tokens and that the amount of attention weights put on the support is tightly correlated with the accuracy, which suggests that the model indeed utilizes the attention mechanism in learning sparse parity.

Specifically, Figure 4.14 shows the results on 2-layer 16-head GPT-2 models. The attention weights are for the final position, whose logits are used for computing the binary parity label for the entire sequence. We track the attention weights along *length-2 paths* from the first and the second layer. For example, for a single-head model, let $\mathbf{a}_i^{(l)} \in \Delta^{d-1}$ denote the l_{th}-layer attention vector at the i_{th} position; then, the on-support attention for a given sample is computed as $\langle \mathbf{a}_d^{(2)}), \mathbf{v}_{\mathcal{T}}^{(1)} \rangle$, where $[\mathbf{v}_{\mathcal{T}}^{(1)}]_i := \sum_{j \in \mathcal{T}} \mathbf{a}_i^{(1)}[j]$ is the total amount of first-layer attention weights that the i_{th} position puts on the support $\mathcal{T}$. For multi-head models, $\mathbf{a}_i^{(l)} \in \Delta^{d-1}$ is defined as the sum of attention vectors from all heads, and the rest is computed similarly.



(a) Width-100 student.



(b) Width-1000 student.

Figure 4.18: 8-way classification using a hierarchical decision tree of depth 3, with each node represented by 5-sparse parity. Progressive distillation helps student learn faster from a width-50k teacher, compared to one-shot distillation from the final checkpoint.

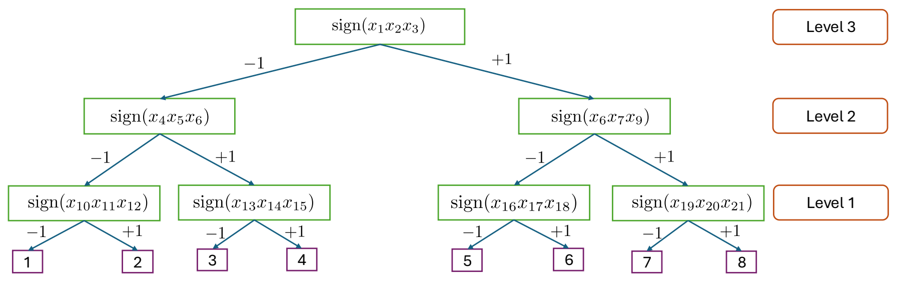

sign($x_1x_2x_3$)
-1
$+1$
sign($x_4x_5x_6$)
sign($x_6x_7x_9$)
sign($x_{10}x_{11}x_{12}$)
sign($x_{13}x_{14}x_{15}$)
sign($x_{16}x_{17}x_{18}$)
sign($x_{19}x_{20}x_{21}$)
1
2
3
4
5
6
7
8
Level 3
Level 2
Level 1

Figure 4.19: An illustration of hierarchical data generation, for a 3-level tree with 3 variables per feature. A feature corresponds to a tree node, each marked by a rectangle. The product of the binary variables in a feature determines which child to take: the left child is chosen if the product evaluates to -1, and the right child is chosen if the product is $+1$. The final label for an example is decided based on the tree leaf reached.

4.2.7.3 A hierarchical generalization of sparse parity

This section considers an extension of sparse parity, where the labels are given by a decision tree. Sparse parity can be considered as a special case with tree depth 1.

Definition: The input $\mathbf{x}$ is a boolean vector picked uniformly at random from the d-dimensional hypercube $\{\pm 1\}^d$, and the label $y \in [K]$ where $K := 2^D$ for some fixed $D \in \mathbb{N}$. The underlying labeling function for y follows a binary decision tree of depth D, whose leaves correspond to class labels. The branching at a node depends on a sparse parity problem. An example visualization is provided in Figure 4.19.

More formally, the nodes in the decision tree are represented by a set of sparse parity problems $\mathbb{S} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{K-1}\}$, where $\mathcal{T}_j$ is determined by product of a subset of size k variables selected from the dimensions of the input $\mathbf{x}$ (e.g. $x_1 x_2 \cdots x_5$ for $k = 5$). An input $\mathbf{x}$ belongs to the class $i \in [K]$ iff

$$[\prod_{j=1}^{D} \mathbb{I}\left[c(i,j)\mathcal{T}_{v_j^{(i)}}(\mathbf{x}) > 0\right] > 0, \quad \text{where}$$

$$c(i,j) = \begin{cases} 1, & \text{if } i \geq 2^{D-j} \\ -1, & \text{otherwise} \end{cases}$$

Here, $v_1^{(i)}, \cdots v_D^{(i)}$ denote the features in $\mathbb{S}$ that lie on the path joining the root of the decision tree to the leaf representing the label i. An example is given in Figure 4.19.

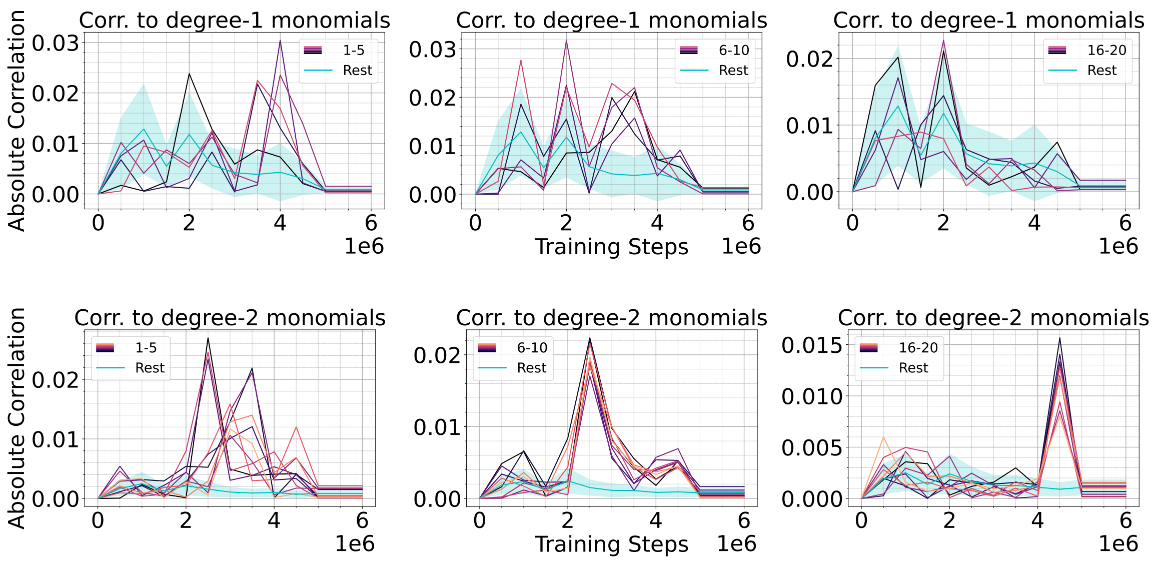


Figure 4.20: Setting: 8-way classification using a hierarchical decision tree of depth 3, with each node represented by 5-sparse parity. The relevant features for class $y = 1$ are $x_1 \cdots x_5, x_6 \cdots x_{10}, x_{16} \cdots x_{20}$ at tree levels 3, 2, and 1 respectively (Figure 4.19). The irrelevant features are $x_{36}, \cdots, x_{100}$. Here we plot the magnitude of correlation to degree-1 monomials $\mathbb{E}_{\mathbf{x},y}[p_{\mathcal{T}}(\mathbf{x})]_1 x_i$ for each i in the relevant feature groups for class 0. Because the degree-1 monomials show noisy correlations, we also report the magnitude of correlation to degree-2 monomials $\mathbb{E}_{\mathbf{x},y}[p_{\mathcal{T}}(\mathbf{x})]_1 x_i x_j$ for each i, j in the relevant feature groups for class 1. For degree-2 monomials, rest refers to correlation to monomials of the form $x_i x_j$ where atleast one variable is outside support variables ($x_36, \cdots, x_100$). The correlations to degree-1 (or 2) monomials on the relevant features spike at different training steps.

Experiment Setup: In this section, we focus on 8-way classification, where the data is generated by a tree of depth 3. Each feature in $\mathbb{S}$ is given by a product of 5 variables. We keep the variables distinct in each feature, i.e., $\mathcal{T}_1 = x_1 x_2 \cdots x_5$, $\mathcal{T}_2 = x_6 x_7 \cdots x_{10}$ and so on.

Experiments and Observations: We conduct similar experiments as our sparse parity experiments. In Figure 4.18, we show that progressive distillation helps train a smaller student as fast as the teacher, and even reach 100% accuracy.

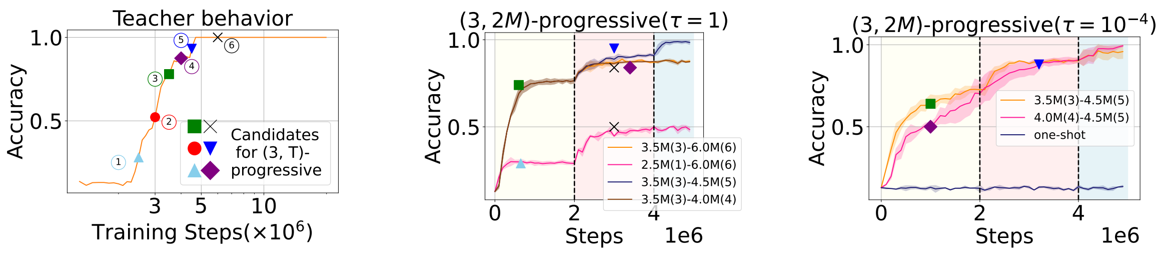


Figure 4.21: Setting: 8-way classification using a hierarchical decision tree of depth 3, with each node represented by 5-sparse parity. $(3, 2M)$-progressive distillation from 3 checkpoints on a 1000 width student; 2 intermediate teacher checkpoints are used each for $2M$ steps, and then the final checkpoint is used till end of training. **Observations:** (a) Teacher shows a phase transition in accuracy during training. 6 candidate checkpoints for $(3, 2M)$-progressive distillation have been marked, out of which 2 are selected in each setting. The checkpoint at $6M$ lies outside the phase transition of the teacher. (b): We show the behavior of a few representative settings. Two main observations: (1) Selecting only a single checkpoint during the phase transition of the teacher is sub-optimal, as shown by plots that contain $6M$ checkpoint as an intermediate checkpoint, (2) 2 checkpoints during the stage transition suffice to train the student to 100% accuracy, however the performance can heavily depend on their selection. Figure 4.20 shows that the teacher learns the low-level features at $4.5M$ checkpoint, making it crucial for distillation. (c): Even with extremely low temperature, the benefit of the phase transition checkpoint persists, suggesting that the monomial curriculum, not regularization, is the key to the success of progressive distillation.

Low-degree curriculum: We show the correlations of the teacher's logits for a particular label and its relevant features in Figure 4.20. We observe similar spikes in the degree-1 monomials involving the support of the features. However, because there are multiple features defining a label class, with features at level 1 being shared among multiple labels, we see a difference in the time-frames at which the spikes appear in the degree-1 monomials of the features. As such, a single teacher checkpoint won't give information of entire support to a student to learn from.

Effectiveness of $(3, T)$-progressive distillation: We consider progressive distillation with 3 checkpoints, where the student only uses 2 intermediate teacher checkpoint in addition to the final one. We show in Figure 4.21 that there exists a $(3, 2M)$-progressive distillation that can help train a student successfully. Furthermore, we demonstrate that these two intermediate checkpoints must be positioned within the phase transition to achieve 100% accuracy in training the student. This supports the hypothesis that a low-degree curriculum is crucial for progressive distillation since the correlations with degree-1 monomials are high only during the phase transition period. Additionally, we find that a distillation strategy with only a single intermediate checkpoint and the final checkpoint is insufficient for the student to achieve 100% accuracy, which aligns with our observation that degree-1 monomials for different features emerge at different steps. However, we also note that even within the phase transition, the optimal selection of the two checkpoints can significantly impact the student model's performance.

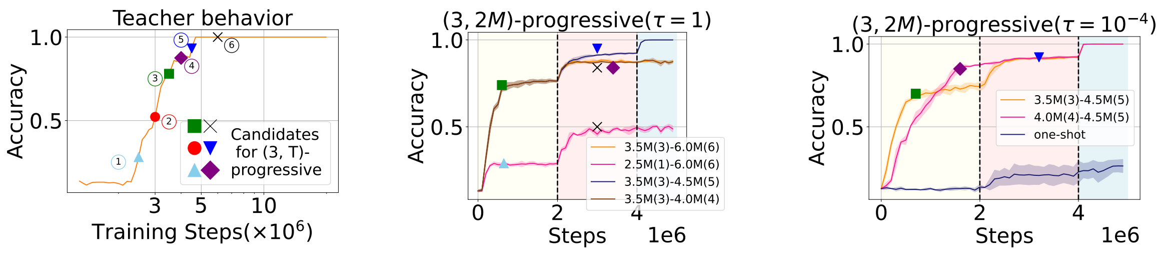


Figure 4.22: Same experiments as Figure 4.21 for a width-100 student.

4.2.8 Extensive study on PCFGs

4.2.8.1 A formal description of PCFGs

We study progressive distillation using probabilistic context free grammar (PCFG). Compared to sparse parity and hierarchical data, PCFG is a more realistic proxy for natural languages and has been commonly used as a sandbox for mechanistically understanding the training of language models [Zhao et al., 2023, Allen-Zhu and Li, 2023b]. A PCFG consists of a set of non-terminals (NTs) and grammar rules involving the non-terminals that specify the generation process of a sentence. For example, for the sentence *The cat ran away*, the grammatical structure dictates words *the*, *cat*, *ran*, *away* as `determinant`, `noun`, `verb`, and `adverb`. *ran* and *away* together represent a `verb phrase`, and *the*, *cat* together represent a `noun phrase` (see Figure 4.6). For a language model to generate grammatically correct sentences, it needs to learn the underlying grammatical rules.

A probabilistic context-free grammar (PCFG) is defined as a 4-tuple $\mathcal{G} = (\mathcal{N}, v, \mathcal{R}, \mathcal{P})$, where

- $\mathcal{N}$ is the set of non-terminals, which can be considered as internal nodes of a parse tree. There is a special non-terminal S, known as the start symbol.
- $[v]$ is the set of all possible words, corresponding to parse tree leaves.
- $\mathcal{R}$ denotes a set of rules. For all $A, B, C \in \mathcal{N}$, there is a rule $A \to BC$ in $\mathcal{R}$. Furthermore, there are rules $A \to w$ for all $A \in \mathcal{N}, w \in [v]$.
- $\mathcal{P}$ specifies the probability of each rule to be used in the generation process. For a rule $r \in \mathcal{R}$, if $\mathcal{P}[r] = 0$, then the rule is an invalid rule under the generation process. Furthermore, for each non-terminal $A \in \mathcal{N}$, on all rules $r \in \mathcal{R}$ of the form $A \to \cdot$, $\sum_{r \in \mathcal{R}: r = A \to \cdot} \mathcal{P}(r) = 1$. We denote $\mathcal{R}(A)$ as the set of all non-zero rules from A.

A concrete example of PCFGs is to model grammars of natural languages [Jurafsky, 2000]. In this case, language tokens form the vocabulary of PCFG, while parts of speech such as nouns, verbs or noun phrases, verb phrases form the non-terminals. Rules like noun phrases being composed of a determinant and a noun form the core of such PCFG, while the probability of each rule is determined by their occurrences across sentences in the language.

Data generation from PCFG Given a PCFG $\mathcal{G} = (\mathcal{N}, v, \mathcal{R}, \mathcal{P})$, a string is generated in a recursive fashion as follows: we start with $s_1 = \text{ROOT}$ at step 1, and maintain a string $s_t \in ([v] \cup \mathcal{N})^*$ at step t. At step t, if all characters in s_t belong to $[v]$, the generation process terminates, and s_t is the resulting string. Otherwise, for each character $A \in s_t$, if $A \in \mathcal{N}$, we sample a rule $r \in \mathcal{R}$ of the form $A \to \cdot$ with probability $\mathcal{P}(r)$ and replace A by characters given by $r(A)$.

Tracking n-grams As outlined in Section 4.2.3, we track the behavior of trained models by measuring the behavior of their output on the neighboring n-gram context. In the context of PCFGs and masked language modeling for BERT, Zhao et al. [2023] theoretically demonstrate that one of the optimal algorithms for predicting masked tokens is a dynamic programming algorithm based on the inside-outside algorithm (textbook reference: Jurafsky [2000]). This algorithm computes "inside probabilities" for spans of tokens of various lengths, representing pairwise token dependencies within those spans. For example, in the setting of Figure 4.6, the inside probability for the span "The cat" indicates the likelihood that these two tokens co-occur. The dynamic programming approach calculates these inside probabilities hierarchically, with smaller spans forming the basis for larger spans. The model's performance ultimately depends on how accurately it represents span probabilities across different lengths. For instance, if the token "cat" is masked in the sentence "The cat ran away", the success of the model depends on the representation of the likelihood of the spans "The cat", "cat ran", "The cat ran", and "The cat ran away". We denote the neighboring tokens in the n-gram window span of a token as its n-gram context.

4.2.8.1.1 Variants of progressive distillation

Comparisons at different lengths We follow common practices for training self-attention models for both one-shot distillation and progressive distillation. We use Adam optimizer [Kingma and Ba, 2014], 512 batch size training (to imitate large batch training), and a cosine learning rate schedule [Loshchilov and Hutter, 2016] which is generally used to train large language models. As cosine learning rate depends on the total training horizon, in order to show that progressive distillation converges faster than one-shot distillation, we compare the two algorithms by varying the number of training samples for the student. That is, we train the teacher model with 4×10^6 training samples (equal to 8000 steps), and compare the two algorithms for a student model at $\{1, 2, 4, 8\} \times 10^6$ training samples (equal to $\{2000, 4000, 8000, 16000\}$ steps).

Progressive Distillation choices Because we are considering comparisons at different training lengths for the student, we have to consider a more general version of progressive distillation introduced in Definition 22. In Definition 22, progressive distillation is defined by two parameters, (a) number of teacher checkpoints (N) for supervision, and (b) training steps per checkpoint. We define our selection criteria for the N checkpoints later. However, after selecting the N checkpoints, we have the following two variants of progressive distillation.

1. N-shot Equal-split distillation: Here, we simply split the entire student's training length into N equal intervals, where the student is supervised by the ith teacher checkpoint in interval $i \in [N]$.
2. N-shot κT_0-Equal-split distillation: Here $\kappa \in (0, 1]$, and T_0 refers to the total training length of the teacher. The idea is to decide the allocation on the basis of the training length of the teacher, instead of the training length for the student. We train the student under the supervision of each checkpoint for $\kappa T_0 / N$ training steps. Teacher checkpoints that fail to fit into the student's supervision schedule are ignored (corresponding to a large κ), and the final checkpoint is kept till the end of training if the student is trained for longer than κT_0. We can view $\frac{1}{\kappa}$ as the amount of "speed up"; for instance, we recover one-shot distillation with $\kappa \to 0$. Our experiments (Section 4.2.8.1.1) suggest that $\kappa = 1/2$ is a reasonable rule of thumb that can help the student learn faster than the teacher at any given training length.

In the main paper, in Figures 4.4, 4.8 and 4.9, we have reported performance on PCFG and Wikipedia for N-shot T_0-Equal-split distillation as progressive distillation. We conduct more ablation studies on κ in Section 4.2.8.1.1. We keep the exploration of optimal strategies of progressive distillation to future work.

Selection criteria for N teacher checkpoints: While there are multiple ways in which one can pick the reference checkpoints to train the student model, we use a simple strategy which is sufficient to demonstrate the benefit of progressive distillation. Similar to our observation of transition phase for parity in Section 4.2.2, we search for transition phases in the loss behavior of the teacher and select the first teacher checkpoint roughly in the middle of the transition phase. The rest are picked at multiples of this initial checkpoint.

4.2.8.2 Details on Non-terminal prediction with Multi-head linear probing

Following Allen-Zhu and Li [2023b], we train a position-based linear attention on the model's embeddings to predict the non-terminals at each level of underlying PCFG. We consider a set of linear functions $f_r : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{N}|}$, where $r \in [H]$ and H is the number of "heads" in the linear attention model. If $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_L$ denote the model's output embeddings for a sequence $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L$, then the prediction of the model at each index $i \in [L]$ is given by

$$G_i(x) = \sum_{r \in [H], k \in [L]} w_{r,i \to k} f_r(\boldsymbol{e}_k),$$

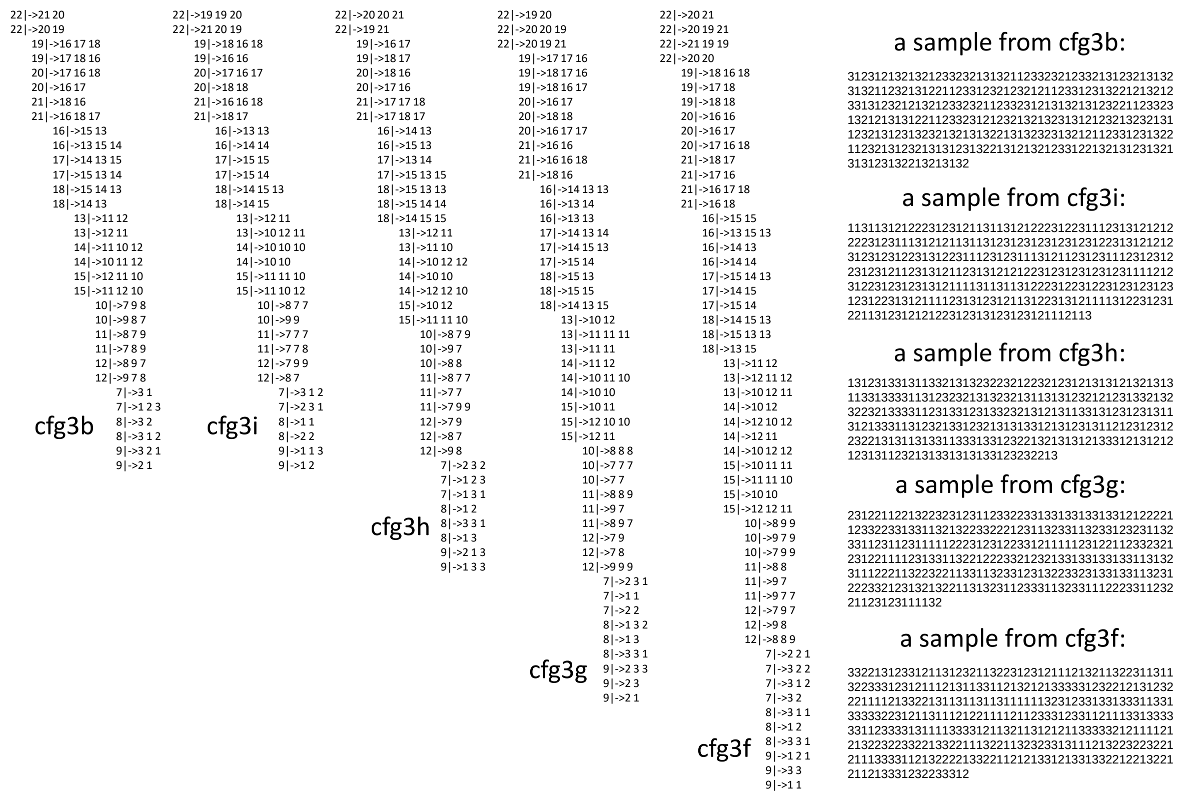
$$w_{r,i \to k} = \frac{exp(\langle P_{i,r}, P_{k,r} \rangle)}{\sum_{k' \in [L]} exp(\langle P_{i,r}, P_{k',r} \rangle)},$$

for trainable parameters $P_{i,r} \in \mathbb{R}^d$. We train the parameters with logistic regression on 51200 examples and test on a validation set of 1024 examples.

4.2.8.3 Details on the synthetic PCFGs

We use 5 synthetic PCFGs considered by Allen-Zhu and Li [2023a] (please see Figure 4.23 for the rules involved in the PCFGs). These 5 PCFGs differ in difficulty, based on the number of rules per non-terminal and the ambiguities in the rules per non-terminal. Under a PCFG, each string is generated by generation trees of depth 7. We give differences in the PCFGs, as outlined by Allen-Zhu and Li [2023a] below.

- In cfg3b, the PCFG is constructed such that the degree $|\mathcal{R}(A)| = 2$ for every non-terminal A. In any generation rule, consecutive pairs of symbols on the generated symbols are distinct. The $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $251, 278, 308, 342$ respectively.
- In cfg3i, $|\mathcal{R}(A)| = 2$ for every non-terminal A. However, the consecutive pairs of symbols needn't be distinct in generation rules. he $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $276, 307, 340, 386$ respectively.
- In cfg3h, $|\mathcal{R}(A)| \in \{2, 3\}$ for every non-terminal A. he $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $202, 238, 270, 300$ respectively.
- In cfg3g, $|\mathcal{R}(A)| = 3$ for every non-terminal A. he $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $212, 258, 294, 341$ respectively.
- In cfg3f, $|\mathcal{R}(A)| \in \{3, 4\}$ for every non-terminal A. he $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $191, 247, 302, 364$ respectively.



cfg3b

```
22|->21 20
22|->20 19
    19|->16 17 18
    19|->17 18 16
    20|->17 16 18
    20|->16 17
    21|->18 16
    21|->16 18 17
        16|->15 13
        16|->13 15 14
        17|->14 13 15
        17|->15 13 14
        18|->15 14 13
        18|->14 13
            13|->11 12
            13|->12 11
            14|->11 10 12
            14|->10 11 12
            15|->12 11 10
            15|->11 12 10
                10|->7 9 8
                10|->9 8 7
                11|->8 7 9
                11|->7 8 9
                12|->8 9 7
                12|->9 7 8
                    7|->3 1
                    7|->1 2 3
                    8|->3 2
                    8|->3 1 2
                    9|->3 2 1
                    9|->2 1
```

cfg3i

```
22|->19 19 20
22|->21 20 19
    19|->18 16 18
    19|->16 16
    20|->17 16 17
    20|->18 18
    21|->16 16 18
    21|->18 17
        16|->13 13
        16|->14 14
        17|->15 15
        17|->15 14
        18|->14 15 13
        18|->14 15
            13|->12 11
            13|->10 12 11
            14|->10 10 10
            14|->10 10
            15|->11 11 10
            15|->11 10 12
                10|->8 7 7
                10|->9 9
                11|->7 7 7
                11|->7 7 8
                12|->7 9 9
                12|->8 7
                    7|->3 1 2
                    7|->2 3 1
                    8|->1 1
                    8|->2 2
                    9|->1 1 3
                    9|->1 2
```

cfg3h

```
22|->20 20 21
22|->19 21
    19|->16 17
    19|->18 17
    20|->18 16
    20|->17 16
    21|->17 17 18
    21|->17 18 17
        16|->14 13
        16|->15 13
        17|->13 14
        17|->15 13 15
        18|->15 13 13
        18|->15 14 14
        18|->14 15 15
            13|->12 11
            13|->11 10
            14|->10 12 12
            14|->10 10
            14|->12 12 10
            15|->10 12
            15|->11 11 10
                10|->8 7 9
                10|->9 7
                10|->8 8
                11|->8 7 7
                11|->7 7
                11|->7 9 9
                12|->7 9
                12|->8 7
                12|->9 8
                    7|->2 3 2
                    7|->1 2 3
                    7|->1 3 1
                    8|->1 2
                    8|->3 3 1
                    8|->1 3
                    9|->2 1 3
                    9|->1 3 3
```

cfg3g

```
22|->19 20
22|->20 20 19
22|->20 19 21
    19|->17 17 16
    19|->18 17 16
    19|->18 16 17
    20|->16 17
    20|->18 18
    20|->16 17 17
    21|->16 16
    21|->16 16 18
    21|->18 16
        16|->14 13 13
        16|->13 14
        16|->13 13
        17|->14 13 14
        17|->14 15 13
        17|->15 14
        18|->15 13
        18|->15 15
        18|->14 13 15
            13|->10 12
            13|->11 11 11
            13|->11 11
            14|->11 12
            14|->10 11 10
            14|->10 10
            15|->10 11
            15|->12 10 10
            15|->12 11
                10|->8 8 8
                10|->7 7 7
                10|->7 7
                11|->8 8 9
                11|->9 7
                11|->8 9 7
                12|->7 9
                12|->7 8
                12|->9 9 9
                    7|->2 3 1
                    7|->1 1
                    7|->2 2
                    8|->1 3 2
                    8|->1 3
                    8|->3 3 1
                    9|->2 3 3
                    9|->2 3
                    9|->2 1
```

cfg3f

```
22|->20 21
22|->20 19 21
22|->21 19 19
22|->20 20
    19|->18 16 18
    19|->17 18
    19|->18 18
    20|->16 16
    20|->16 17
    20|->17 16 18
    21|->18 17
    21|->17 16
    21|->16 17 18
    21|->16 18
        16|->15 15
        16|->13 15 13
        16|->14 13
        16|->14 14
        17|->15 14 13
        17|->14 15
        17|->15 14
        18|->14 15 13
        18|->15 13 13
        18|->13 15
            13|->11 12
            13|->12 11 12
            13|->10 12 11
            14|->10 12
            14|->12 10 12
            14|->12 11
            14|->10 12 12
            15|->10 11 11
            15|->11 11 10
            15|->10 10
            15|->12 12 11
                10|->8 9 9
                10|->9 7 9
                10|->7 9 9
                11|->8 8
                11|->9 7
                11|->9 7 7
                12|->7 9 7
                12|->9 8
                12|->8 8 9
                    7|->2 2 1
                    7|->3 2 2
                    7|->3 1 2
                    7|->3 2
                    8|->3 1 1
                    8|->1 2
                    8|->3 3 1
                    9|->1 2 1
                    9|->3 3
                    9|->1 1
```

a sample from cfg3b:

```
312312132132123323213132112332321233213123213132
313211232131221123312321232121123312313221213212
331312321213212332321123323121313213123221123323
132121313122112332312123213231312123213232131
123213123132321321313221313232313212112331231322
112321312321313123132213121321233122132131231321
313123132213213132
```

a sample from cfg3i:

```
113113121222312312113113121222312231112313121212
222312311131212113113123123123123123122313121212
312312312231312231112312311131211231231112312312
231231211231312112313121212231231231231231111212
312231231231312111131131131222312231223123123123
123122313121111231312312113122313121111312231231
221131231212122312313123123121112113
```

a sample from cfg3h:

```
131231331311332131323223212232123121313121321313
113313333113123232131323213113131232121231332132
322321333311231331231332321312131133131231231311
312133311312321331232131313312131231311212312312
232213131131331133313312322132131312133312131212
1231311232131331313133123232213
```

a sample from cfg3g:

```
231221122132232312311233223313313313312122221
123322331331132132233222123113233113233123231132
331123112311111222312312233121111123122112332321
231221111231331132212223321232133133133133113132
311122211322322113311323312313223323133133113231
222332123132132211313231123331132331112223311232
21123123111132
```

a sample from cfg3f:

```
332213123312113123211322312312111213211322311311
322333123121112131133112132121333331232212131232
221111213322131131131131111113231233133133311331
333332231211311121221111211233312331121113313333
331123333131111333312113211312121133333212111121
213223223322133221113221132323313111213223223221
211133331121322221332211212133121331332212213221
211213331232233312
```

Figure 4.23: The synthetic PCFGs considered from Allen-Zhu and Li [2023b]. Vocabulary is $\{1, 2, 3\}$ in each setting. More details on the differences between the PCFGs are in Section 4.2.8.3.

4.2.8.4 Extensive experiments on BERT

We first give some details on the architecture of BERT and its pre-training loss function.

4.2.8.4.1 A primer on BERT

BERT [Devlin et al., 2018b] is an encoder-only transformer that is trained with masked language modeling (MLM) (Figure 4.24). In encoder-only architecture, the contextual information are shared across the tokens using bidirectional self-attention layers. During pre-training, the model is trained with MLM loss, that perturbs certain fraction of the tokens in the input at random and the model is trained to predict the original tokens at positions of the perturbed tokens. The pre-training recipe follows a 80-10-10 principle, where tokens at 80% of the perturbed positions are replaced by a special $\langle mask \rangle$ token, while tokens at 10% of the perturbed positions are replaced by random tokens from the vocabulary, while remaining positions are filled with the original tokens themselves. We stick to this principle, while creating data for training from different PCFGs.

Model architecture considered: We train depth-4 BERT models with $\{8, 16, 32\}$ attention heads, each of which operates on 8 dimensions, using a 30% masking rate. The head dimension is fixed to 8, with the corresponding width of the 4 models being $\{64, 128, 256\}$ respectively.

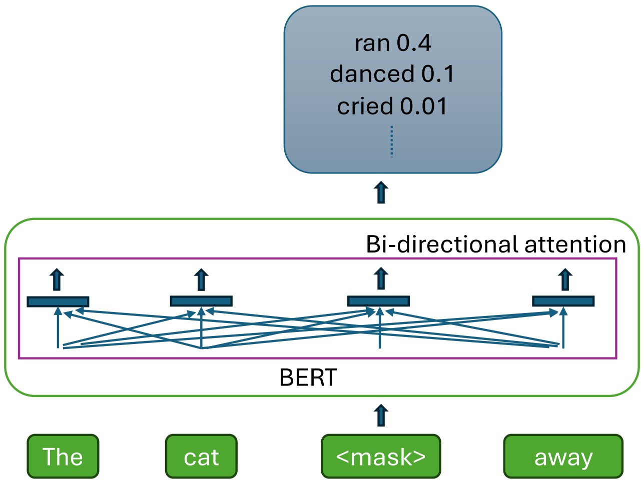


Figure 4.24: An informal representation of BERT [Devlin et al., 2018b]. The model uses bidirectional attention layers to share contextual information across the tokens. During pre-training, few of input tokens are replaced by special $< mask >$ tokens, and the model is trained to predict the masked tokens.

4.2.8.4.2 Data Generations

Data for masked language modeling: We generate 8×10^6 random sequences for each PCFG. We follow Devlin et al. [2018b] to create masked input sequences and output labels, i.e. for each sampled sequence we mask $p\%$ of tokens for input and the labels are given by the tokens in the masked positions of the original sequence. We also follow the 80-10-10 principle, where for input, the tokens in 80% of the masked positions are represented by a special mask token $\mathcal{C}$, while 10% of the masked positions are represented by a randomly sampled token from the vocabulary and the remaining 10% are represented by tokens from the original sequence.

4.2.8.5 Hyperparameter details

We use a batch size of 512 in each setting. We use Adam [Kingma and Ba, 2014] optimizer with 0 weight decay, $\beta_1, \beta_2 = (0.9, 0.95)$. We use cosine decay learning rate. We extensively tune the learning rate in the grid $\{10^{-2}, 7.5 \times 10^{-3}, 5 \times 10^{-3}, 2.5 \times 10^{-3}, 10^{-3}\}$ in each setting. We train the teacher on 4×10^6 training samples (equal to 8×10^3 steps).

Distillation experiments at different training horizons: To thoroughly compare the sample complexity requirements of one-shot and progressive distillation, we evaluate both algorithms using a smaller student model across various training sample sizes. The smaller student is trained with $\{1, 2, 4, 8\} \times 10^6$ training samples (equal to $\{2 \times 10^3, 4 \times 10^3, 8 \times 10^3, 16 \times 10^3\}$ training steps) and the performance is compared in each horizon. For example, Figure 4.4 (right) plot contains 4 distinct points for each method which represents the performance of the smaller model under the 4 different training steps (sample sizes).

Training split for $(2, T)$-progressive distillation for PCFGs: We report the performance in Figure 4.7 for 4000 training steps. We find the best training time split T between the intermediate checkpoint and the final checkpoint in the grid $\{500, 1000, 15000, 2000\}$, i.e. the student is trained with the logits of the first intermediate teacher checkpoint till step T and then the teacher is switched to the final teacher checkpoint.

Low-temperature distillation: We focus on distillation with a small temperature of $\tau = 10^{-4}$ (in Equation (4.90)), for the following reasons. First, as discussed in Section 4.2.2, it removes any potential regularization effects induced by soft labels. Moreover, using such a small temperature corresponds to training with the top-1 predictions of the teacher model, which is more memory-efficient compared to training with the full teacher logits, especially when the vocabulary size is large.

4.2.8.6 Additional Curriculum probing on the teacher's checkpoints

In this section, we study the performance of different progressive distillation variants and compare them to one-shot distillation. As per our experiments in Figure 4.7, we use the 8 teacher checkpoints selected for supervision. In Figure 4.27, we compare one-shot distillation to the two variants of progressive distillation, i.e. 8-shot Equal-split and 8-shot $\frac{T_0}{2}$-Equal-split distillation. We observe that both variants of progressive distillation help the student learn faster than one-shot distillation, and the gap diminishes as the students are trained for longer. The optimal strategy for progressive distillation depends on the training budget for the student. For training steps lower than the teacher's T_0 budget, $\frac{T_0}{2}$-Equal-split distillation slightly performs better than T_0-Equal-split distillation, which changes as we train longer. **To keep things simple, we focus on $\frac{T_0}{2}$-Equal-split progressive distillation in all of our subsequent experiments.**

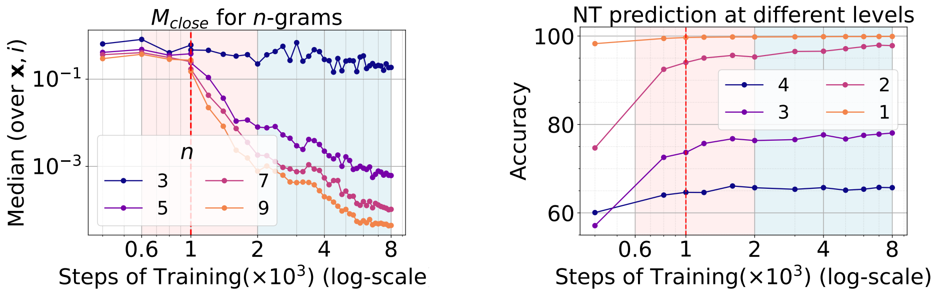


Figure 4.25: We conduct additional probing experiments on the teacher's (4 layer, 32 attention head BERT) logits during training to indicate curriculum learning. (left) TV distance between model's predictions with full context and context with only n-gram tokens (L_{close}). We observe that the teacher's logits get closer to higher n-gram context predictions, and the inflection appears at the middle of the second phase (our first selected checkpoint for progressive distillation) (right) Performance of linear classifier probe on teacher's intermediate checkpoints to predict the non-terminals at different levels of the PCFG generation tree. We observe that the probe's performance is $> 95\%$ of the final probe performance by the middle of the second phase, indicating the model has almost learned the underlying PCFG features by this time.

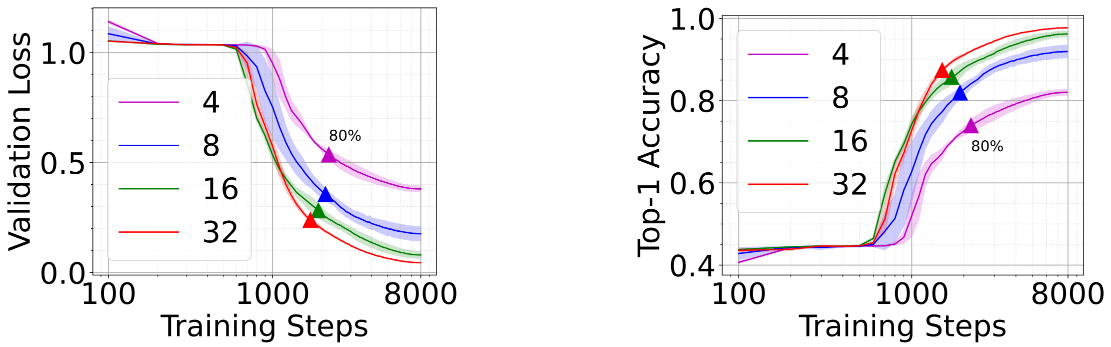


Figure 4.26: Comparison of BERT's training behavior on `cfg3b` with varying numbers of attention heads (where the embedding dimension scales linearly with the number of attention heads) over 8×10^3 training steps. The x-axis represents the number of training steps and is in log scale. Larger BERT models show an earlier and more pronounced drop in loss/increase in accuracy compared to smaller models. For reference, each training curve is annotated at the point where the model reaches 80% of its performance at the final step.

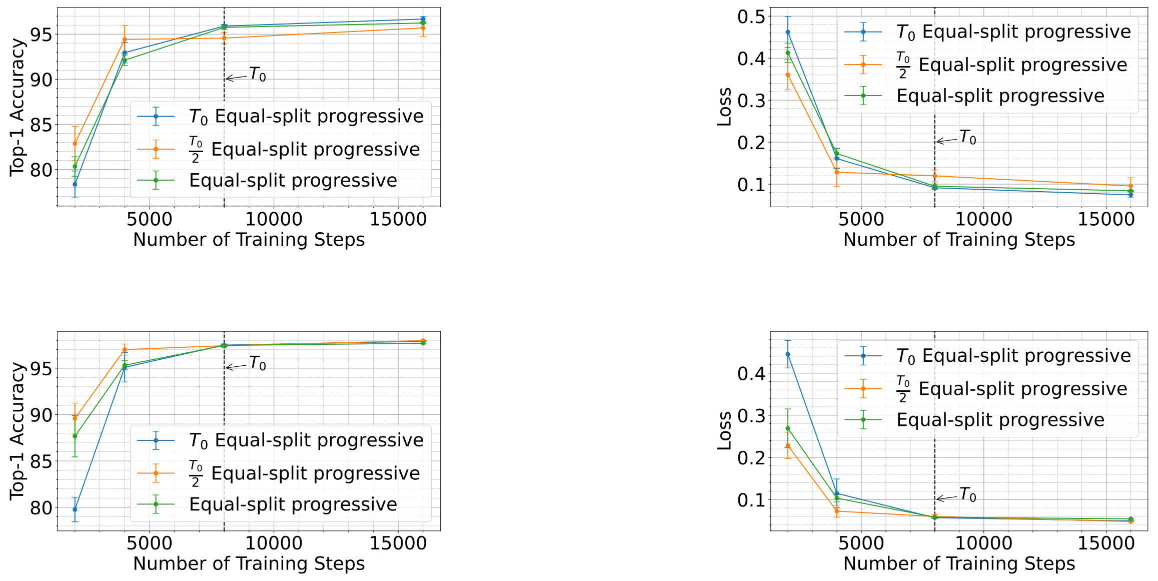
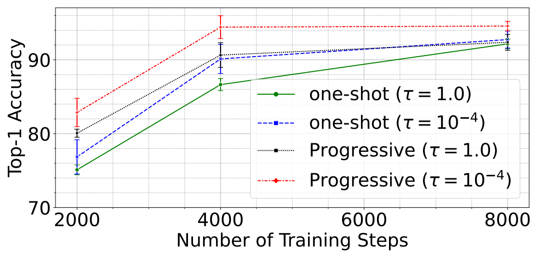


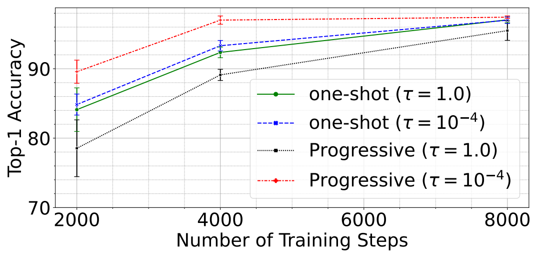
Figure 4.27: Experiments on BERT (Left to right/top to bottom): (a), (b) show the comparisons for an 8-attention head student, (c), (d) show the comparisons for a 16-attention head student. We observe differences between the different variants of progressive distillation at different training steps. For training steps lower than the teacher's (marked by T_0), $T_0/2$-Equal-split progressive distillation is better, implying that for shorter training, we shouldn't try to fit all the teacher's checkpoints. The trend reverses as the training sample budget approaches T_0 and beyond.

4.2.8.7 Ablations with hyperparameters

Ablation with temperature Here, we compare progressive distillation and one-shot distillation at temperature 1 and temperature 10^{-4} (representing hard label supervision) (Figure 4.28). We observe that progressive distillation at temperature 10^{-4} performs better than one-shot distillation at both temperatures. However, progressive distillation at temperature 1 can perform worse than one-shot distillation for a stronger student. We keep explorations on the effect of temperature on the algorithms as future work.



(a) Number of heads=8



(b) Number of heads=16

Figure 4.28: The experiments above compare progressive distillation and one-shot distillation at temperature 1 and 10^{-4} (representing hard label supervision) for PCFGs `cfg3b` at masking rate 30% using a BERT model with 8 attention heads (left)/ 16 attention heads (right), per head dimension 8, and 4 layers. We observe that progressive distillation with hard labels performs better than one-shot distillation at temperatures 1 and 10^{-4}. However, progressive distillation at temperature 1 can perform worse than one-shot distillation for stronger student. We keep explorations on the effect of temperature on the algorithms as future work. Here, we use $\frac{T_0}{2}$-Equal-split progressive distillation as progressive distillation, where $T_0 = 8000$ is the total number steps used for teacher training.

Ablation with mask rate In Figure 4.29, we compare progressive distillation with one-shot distillation at different masking rates. We observe that at all masking rates, progressive distillation performs better than one-shot distillation.

Ablation with difficulty of PCFG In Figure 4.30, we compare progressive distillation with one-shot distillation with increasing difficulty of the underlying PCFG. The benefit of progressive distillation over one-shot distillation is influenced by the model's capacity and the specific PCFG being trained.

4.2.9 Autoregressive training with GPT2

Setting: Similar to experiments on BERT, we train GPT2 models of depth 4 with $\{8, 16, 32\}$ attention heads, while keeping the dimension per attention head fixed at 8.

A brief introduction into GPT models: GPT models are trained with the auto-regressive loss. The teacher and student models operate on sequences of input domain $f_{\mathcal{T}} : \mathcal{X}^h \to \mathbb{R}^C$ and $f_{\mathcal{S}} : \mathcal{X}^h \to \mathbb{R}^C$, where the input sequence length h can be arbitrary. Denote the length-h input sequence as $\mathbf{x} :=$

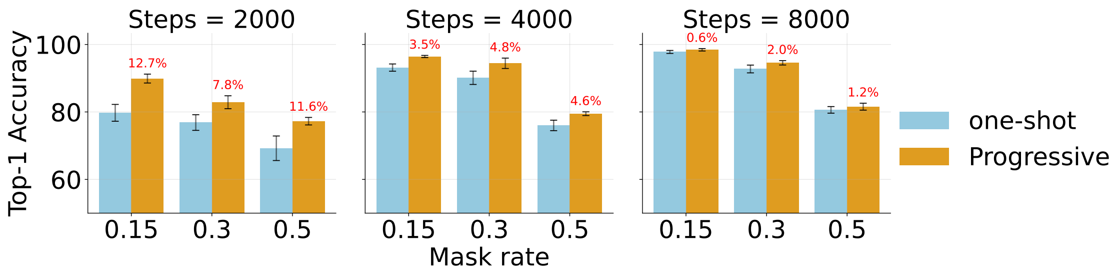


Figure 4.29: The experiments above compare progressive distillation and one-shot distillation for PCFGs `cfg3b` at different masking rates using a BERT model with 8 attention heads, per head dimension 8, and 4 layers. The relative gap between the performance of progressive distillation and one-shot distillation have been reported on the bar plots. We observe that progressive distillation performs better than one-shot distillation at all masking rates, with the gap diminishing with the number of training steps. Here, we use $\frac{T_0}{2}$-Equal-split progressive distillation as progressive distillation, where $T_0 = 8000$ is the total number steps used for teacher training.

$[x_1, \cdots, x_h]$, and denote $\mathbf{x}_{i:j}$ as the subsequence $[x_i, \cdots, x_j]$ (i.e. the indexing is inclusive on both ends). The cross entropy loss for next-token prediction training on $\mathbf{x}$ is given by

$$\frac{1}{h}\sum_{i=1}^{h} \mathrm{KL}(\boldsymbol{e}_{x_i} \| p_{\mathcal{S}}(\mathbf{x}_{1:i-1}))),$$

where $\boldsymbol{e}_{x_i}$ denotes a one-hot vector with 1 in x_ith coordinate. We take a different approach, where we compare the algorithms at different difficult levels, by training on a subset of tokens in each sequence. The subsets that we consider are the boundary tokens at different levels of PCFG generation (recall Figure 4.6).

Formally, if $\mathcal{C}^{(\ell)}(\boldsymbol{x})$ represents the set of level-ℓ boundary tokens, then we define the cross entropy loss and the distillation loss corresponding to boundary tokens at any level ℓ of the PCFG as

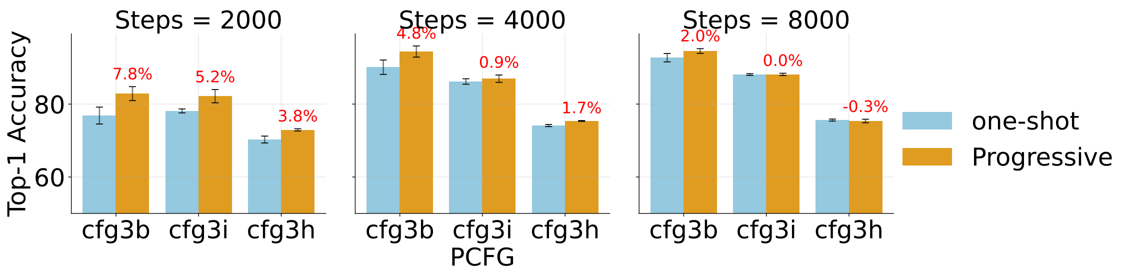
$$\ell^{(\ell)}(\mathbf{x}; f_{\mathcal{S}}) = \frac{1}{|\mathcal{C}^{(\ell)}(\boldsymbol{x})|} \sum_{i: x_i \in \mathcal{C}^{(\ell)}(\boldsymbol{x})} \mathrm{KL}(\boldsymbol{e}_{x_i} \| p_{\mathcal{S}}(\mathbf{x}_{1:i-1})); \tag{4.98}$$

$$\ell_{\mathrm{D}}^{(\ell)}(\mathbf{x}; f_{\mathcal{S}}, f_{\mathcal{T}}) = \frac{1}{|\mathcal{C}^{(\ell)}(\boldsymbol{x})|} \sum_{i: x_i \in \mathcal{C}^{(\ell)}(\boldsymbol{x})} \mathrm{KL}(p_{\mathcal{T}}(\mathbf{x}_{1:i-1}; \tau) \| p_{\mathcal{S}}(\mathbf{x}_{1:i-1})). \tag{4.99}$$

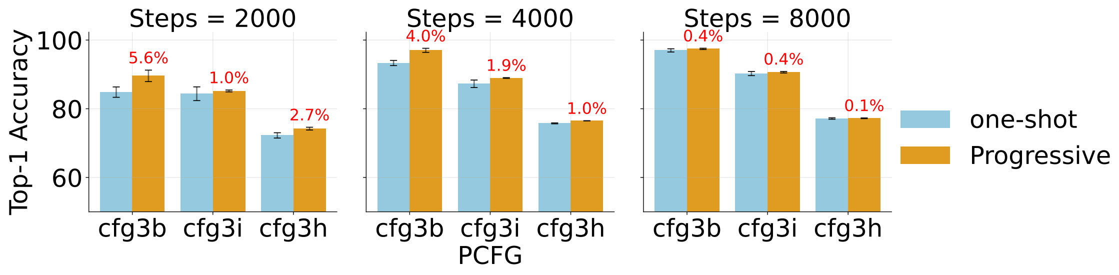
There are a few remarks that need to be made about the above loss function. First, note that the subsets satisfy the condition $\mathcal{C}^{(\ell_1)}(\boldsymbol{x}) \subseteq \mathcal{C}^{(\ell_2)}(\boldsymbol{x})$ for all $\ell_1 \geq \ell_2$. Hence, the loss $L^{(\ell_2)}$ includes loss $L^{(\ell_1)}$ for all $\ell_1 \geq \ell_2$ and losses $L \in \{\ell, \ell_{\mathrm{D}}\}$. Second, $L^{(1)}$ will average the losses at all tokens, which is the standard auto-regressive loss used in practice to train large language models.

We focus on `cfg3f` that has 6 levels in the generation process, and we report the behavior of the models when trained with losses $L^{(2)}, L^{(3)}, L^{(4)}$, with $L \in \{\ell, \ell_{\mathrm{D}}\}$. We focus on $\frac{T_0}{2}$-Equal-split progressive distillation.

Definitions for L_{robust} and L_{close}. Similar to our experiments on BERT, we track the change in the model's predictions with and without the n-gram context tokens. However, as the model is



(a) Number of attention heads=8



(b) Number of attention heads=16

Figure 4.30: The experiments above compare progressive distillation and one-shot distillation for PCFGs `cfg3b`, `cfg3h`, and `cfg3i` at masking rate 30% using BERT models with 8/16 attention heads, per head dimension 8, and 4 layers. The relative gap between the performance of progressive distillation and one-shot distillation have been reported on the bar plots. The benefit of progressive distillation over one-shot distillation is influenced by the model's capacity and the specific PCFG being trained. For instance, on `cfg3i`, the student model can only achieve a top-1 accuracy of 75%. Progressive distillation reaches this within 2000 steps but fails to improve further, resulting in minimal gains over one-shot distillation when compared with `cfg3b`. The comparisons are at temperature $\tau = 10^{-4}$. Here, we use $\frac{T_0}{2}$-Equal-Split Progressive Distillation as Progressive Distillation, where $T_0 = 8000$ is the total number steps used for teacher training. Our teacher is a BERT model with 32 attention heads, per head dimension 8, and 4 layers, which doesn't train on `cfg3g` and `cfg3f`, hence we don't report the performance of the student on these PCFGs.

trained autoregressively, we need to change our definitions of L_{robust} and L_{close} from Equations (4.94) and (4.95), as well as the definition of n-grams.

For a h length sentence $\boldsymbol{x} \in v^h$ and for $i \in [h]$, we define the *n-gram neighboring context* around the i_{th} token as the set of tokens at positions within $n-1$ distance to the left from i, i.e. the set $\{x_j\}$ for $i-n < j < i$.

For L_{close} on a teacher $f_{\mathcal{T}}$ and ngram length n, we measure the TV distance between the model's probability distributions of the model at any position i when all the tokens at positions $1, 2, \cdots, i-1$ are available, and when only the tokens in the neighboring n-gram context window are available (i.e. at positions $i-n+2, \cdots, i-1$)[23]

$$L_{\text{close}}(f_{\mathcal{T}}, \mathbf{x}, i, n) = \text{TV}(p_{\mathcal{T}}(\mathbf{x}_{1:i-1}), p_{\mathcal{T}}(\mathbf{x}_{i-n+1:i-1})). \tag{4.100}$$

For L_{robust} on a teacher $f_{\mathcal{T}}$ and an n-gram length n, we measure the total variation (TV) distance between the model's probability distributions at any position i, considering two scenarios: one where all tokens at positions $1, 2, \ldots, i-1$ are available, and another where the tokens within the n-gram context window are masked. However, since the attention mechanism in GPT requires a token at position $i-1$ before it can predict x_i and we don't have a special token to replace the masked tokens, we cannot remove that specific token from the context. Therefore, we keep the token at position $i-1$ intact while masking the other tokens within the n-gram context window. We refer to this modified approach as "skip n-gram."

$$L_{\text{robust}}(f_{\mathcal{T}}, \mathbf{x}, i, n) = \text{TV}(p_{\mathcal{T}}(\mathbf{x}_{1:i})), p_{\mathcal{T}}(\mathbf{x}_{\{1,\cdots,i-n+1,i\}}))). \tag{4.101}$$

4.2.9.1 Observations

Teacher's behavior during training Figure 4.32 shows the loss behavior of a teacher run. We observe 2 distinct phases of training: a rapid loss drop phase in the first 10% of training, and a final phase of slow loss drop till end of training. In Figure 4.31, we compare the training accuracy behavior across models of different sizes. At log scale, we observe a very small dormant phase in the training behavior at the start of training. Larger models transition to the rapid loss drop phase faster than smaller models and also show a more prominent change in this phase.

Teacher's checkpoint selection for progressive distillation As outlined in the previous section, we select the first supervision checkpoint at roughly the middle of the first phase ($1/20_{\text{th}}$ fraction of training), and the other checkpoints are selected at $\{i/20\}_{i=2}^{20}$ fractions of training.

Similar inflection points in loss as BERT and an implicit curriculum:

We observe inflection points in the model's behaviors at the first selected checkpoint. Similar to our observations on BERT, we observe a curriculum on the reliance of the model's predictions on 3-gram predictions (Figure 4.32). Hence, we check whether progressive distillation can help train a smaller model faser.

(R7) Progressive distillation helps train smaller model faster In Figures 4.33 and 4.34, we compare one-shot distillation to $\frac{T_0}{2}$-Equal-split progressive distillation. We observe that progressive distillation

[23]others are simply masked out during attention score computation to avoid shifts in position embeddings.

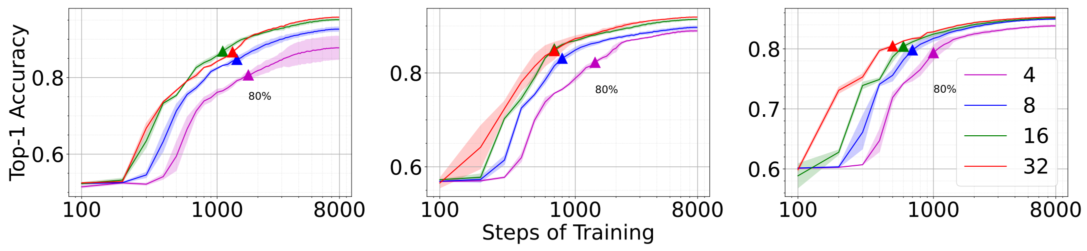
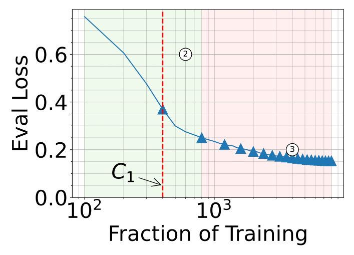


Figure 4.31: (left to right) Models are trained with the cross entropy loss $\ell^{(4)}, \ell^{(3)}, \ell^{(2)}$ respectively. Here, we compare GPT's training behavior with cross entropy loss on `cfg3f` with varying numbers of attention heads (where the embedding dimension scales linearly with the number of attention heads) over 8×10^3 training steps. Larger models show an earlier and more pronounced increase in performance compared to smaller models. For reference, each training curve is annotated at the point where the model reaches 80% of its performance at the final step.

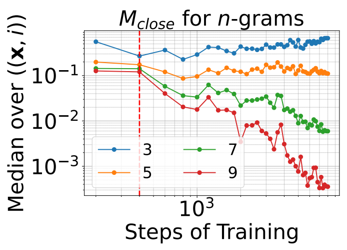
help the student learn faster than one-shot distillation, and the gap diminishes as the students are trained for longer. However, the gap between progressive distillation and distillation decreases as more tokens are involved in the loss function i.e. the gap is smaller for loss $L_0^{(2)}$ compared to loss $L_0^{(4)}$. We conjecture that auto-regressive training with all tokens involved provides a strong curriculum for the model to learn the structure of the language. We keep a thorough study of this analysis to future work.

4.2.10 Details on Wikipedia + Books experiments

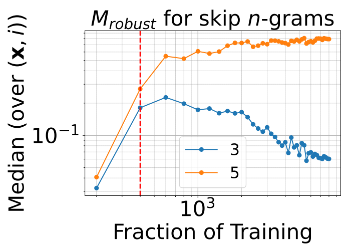
We use the same hyperparameters for Adam training as our experiments on BERT and PCFG in Section 4.2.8.5. However, we fix the peak learning rate to 10^{-4} [Devlin et al., 2018b] in each case to minimize computation costs.



(a) Loss behavior of teacher model

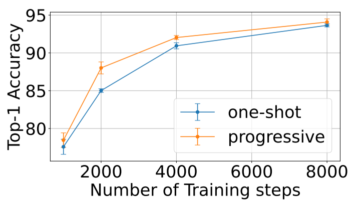


(b) Median of $L_{\text{close}}(f_{\mathcal{T}}, \mathbf{x}, i, n)$ over $\mathbf{x}, i$ for different n

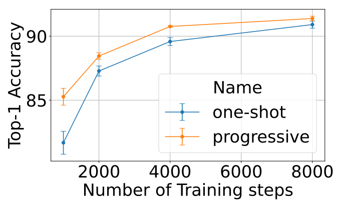


(c) Median of $L_{\text{robust}}(f_{\mathcal{T}}, \mathbf{x}, i, n)$ over $\mathbf{x}, i$ for different n

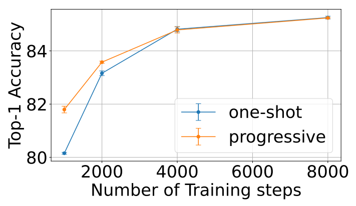
Figure 4.32: Experiments on GPT: Behavior of teacher model when trained on `cfg3f` with cross entropy loss: $\ell^{(3)}$. We observe two distinct phases; (2) a rapid drop in loss phase, and (3) slow drop in loss till end of training. The rapid loss drop phase signifies a transition phase for the model, similar to one we observed for hierarchical boolean data (Section 4.2.2). All selected checkpoints for progressive distillation are marked by triangles. The first teacher checkpoint is roughly picked at the center of the second phase. The rest of the checkpoints are picked at training steps that are multiples of the first one. (b) and (c) show inflection points in the teacher's predictions with full context and with/without n-gram contexts at the selected checkpoint.



(a) Loss: $\ell_{\mathrm{D}}^{(4)}$

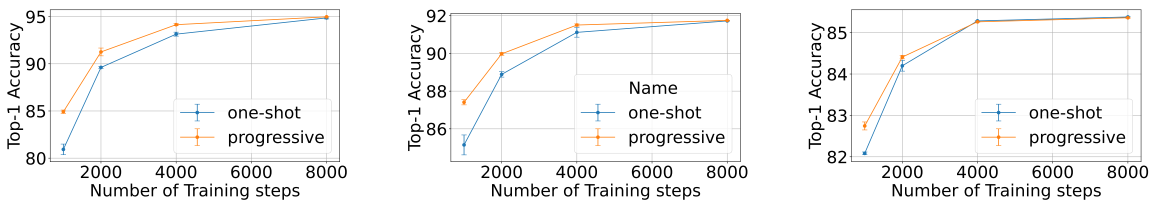


(b) Loss: $\ell_{\mathrm{D}}^{(3)}$



(c) Loss: $\ell_{\mathrm{D}}^{(2)}$

Figure 4.33: Experiments on GPT (Left to right) for an 8- attention head model at different losses. Here, progressive distillation refers to $\frac{T_0}{2}$-Equal-split progressive distillation. We observe that progressive distillation outperforms one-shot distillation at all training sample budgets, with the gap diminishing with increasing training sample budget. The gap between progressive distillation and distillation decreases as the number of tokens involved in the loss function increases i.e. the gap is smaller for loss $L_0^{(2)}$ compared to loss $L_0^{(4)}$.



(a) Loss: $\ell_{\mathrm{D}}^{(4)}$ (b) Loss: $\ell_{\mathrm{D}}^{(3)}$ (c) Loss: $\ell_{\mathrm{D}}^{(2)}$

Figure 4.34: Experiments on GPT (Left to right) for a 16-attention head model at different losses. Here, progressive distillation refers to $\frac{T_0}{2}$-Equal-split progressive distillation. We observe that progressive distillation outperforms one-shot distillation at all training sample budgets, with the gap diminishing with increasing training sample budget. The gap between progressive distillation and distillation decreases as the number of tokens involved in the loss function increases i.e. the gap is smaller for loss $L_0^{(2)}$ compared to loss $L_0^{(4)}$.

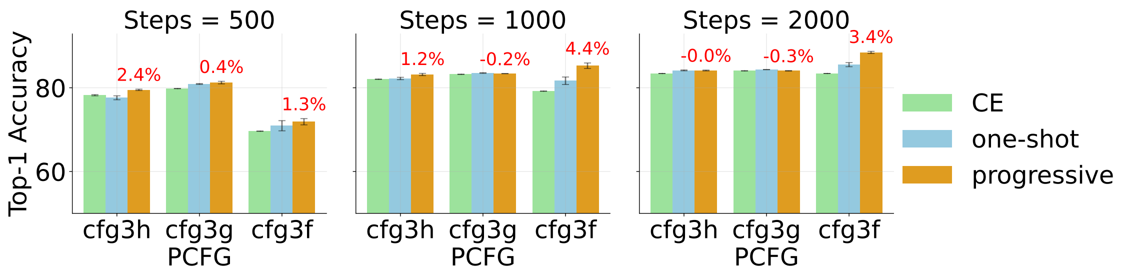


Figure 4.35: The experiments above compare progressive distillation and one-shot distillation for PCFGs `cfg3h`, `cfg3g`, and `cfg3f` on GPT models with 8 attention heads (each head having a dimension of 8) and 4 layers. The models were trained using the distillation loss $L_0^{(3)}$. The relative performance gap between progressive and one-shot distillation is presented in the bar plots. Notably, the advantage of progressive distillation over one-shot distillation depends on the specific PCFG being trained. For example, with `cfg3f`, the student model can achieve beyond 90% top-1 accuracy, and progressive distillation allows it to reach this more quickly. In contrast, for `cfg3g`, the student model's top-1 accuracy plateaus at 84%, and after 500 steps, progressive distillation shows only marginal gains over one-shot distillation. All comparisons were made at a temperature $\tau = 10^{-4}$. Here, progressive distillation refers to $\frac{T_0}{2}$-Equal-split progressive distillation, where $T_0 = 8000$ denotes the total number of steps for teacher training. The teacher model is a GPT with 32 attention heads, each with a dimension of 8, and 4 layers.

Bibliography

E. Abbe, E. Boix-Adsera, and T. Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. *arXiv preprint arXiv: 2202.08658*, 2022.

E. Abbe, S. Bengio, A. Lotfi, and K. Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv: Arxiv-2301.13105*, 2023a.

E. Abbe, E. Boix-Adserà, and T. Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. *Annual Conference Computational Learning Theory*, 2023b. doi: 10.48550/arXiv.2302.11055.

E. Abbe, E. Cornacchia, and A. Lotfi. Provable advantage of curriculum learning on parity targets with mixed inputs. *Advances in Neural Information Processing Systems*, 36, 2024.

S. Abnar, M. Dehghani, and W. Zuidema. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*, 2020.

R. Agarwal, N. Vieillard, Y. Zhou, P. Stanczyk, S. R. Garea, M. Geist, and O. Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Aharon, M. Elad, and A. M. Bruckstein. On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them. *Linear algebra and its applications*, 416(1):48–67, 2006.

Y. Ait-Sahalia et al. Closed-form likelihood expansions for multivariate diffusions. *The Annals of Statistics*, 36(2):906–937, 2008a.

Y. Ait-Sahalia et al. Closed-form likelihood expansions for multivariate diffusions. *The Annals of Statistics*, 36(2):906–937, 2008b.

E. Akyürek, B. Wang, Y. Kim, and J. Andreas. In-context language learning: Architectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024.

Z. Allen-Zhu and Y. Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=Uuf2q9TfXGA.

Z. Allen-Zhu and Y. Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023b.

E. S. Allman, C. Matias, and J. A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A):3099–3132, 2009.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1. JMLR Workshop and Conference Proceedings, 2012.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of machine learning research*, 15:2773–2832, 2014.

C. Anil, A. Pokle, K. Liang, J. Treutlein, Y. Wu, S. Bai, J. Z. Kolter, and R. B. Grosse. Path independent equilibrium models can better exploit test-time computation. *Advances in Neural Information Processing Systems*, 35:7796–7809, 2022a.

C. Anil, Y. Wu, A. Andreassen, A. Lewkowycz, V. Misra, V. Ramasesh, A. Slone, G. Gur-Ari, E. Dyer, and B. Neyshabur. Exploring length generalization in large language models. *arXiv:2207.04901*, 2022b.

R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton. Large scale distributed neural network training through online distillation. *International Conference on Learning Representations*, 2018.

S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

S. Arora, R. Ge, and A. Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Conference on Learning Theory*, pages 779–806. PMLR, 2014.

S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

A. Bansal, A. Schwarzschild, E. Borgnia, Z. Emam, F. Huang, M. Goldblum, and T. Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking. *arXiv:-2202.05826*, 2022.

B. Barak, B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. *arXiv:2207.08799*, 2022a.

B. Barak, B. L. Edelman, S. Goel, S. M. Kakade, E. Malach, and C. Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *NeurIPS*, 2022b. URL http://papers.nips.cc/paper_files/paper/2022/hash/884baf65392170763b27c914087bde01-Abstract-Conference.html.

D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Symposium on the Theory of Computing*, 1986.

D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 1988.

Y. Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48 (1):207–219, Mar. 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7.

A. Bertsch, U. Alon, G. Neubig, and M. R. Gormley. Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*, 2023.

R. Bhattacharya et al. Criteria for recurrence and existence of invariant measures for multidimensional diffusions. *The Annals of Probability*, 6(4):541–553, 1978.

S. Bhattamishra, K. Ahuja, and N. Goyal. On the ability and limitations of transformers to recognize formal languages. In *Conference on Empirical Methods in Natural Language Processing*, 2020a.

S. Bhattamishra, K. Ahuja, and N. Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online, Nov. 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.576. URL https://aclanthology.org/2020.emnlp-main.576.

S. Bhattamishra, A. Patel, V. Kanade, and P. Blunsom. Simplicity bias in transformers and their ability to learn sparse boolean functions. *Annual Meeting of the Association for Computational Linguistics*, 2022. doi: 10.48550/arXiv.2211.12316.

S. Bhattamishra, M. Hahn, P. Blunsom, and V. Kanade. Separations in the representational capabilities of transformers and recurrent architectures. *arXiv preprint arXiv: 2406.09347*, 2024.

B. Bilodeau, N. Jaques, P. W. Koh, and B. Kim. Impossibility theorems for feature attribution. *arXiv preprint arXiv:2212.11870*, 2022.

T. Bolukbasi, A. Pearce, A. Yuan, A. Coenen, E. Reif, F. Viégas, and M. Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv: Arxiv-2104.07143*, 2021.

B. Bordelon and C. Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *Advances in Neural Information Processing Systems*, 35:32240–32256, 2022.

B. Bordelon and C. Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

B. Bordelon, A. B. Atanasov, and C. Pehlevan. A dynamical model of neural scaling laws. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=nbOY1OmtRc.

B. Bordelon, H. T. Chaudhry, and C. Pehlevan. Infinite limits of multi-head transformer dynamics. *arXiv preprint arXiv: 2405.15712*, 2024b.

G. Bresler. Efficiently learning ising models on arbitrary graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 771–782, 2015.

M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer. On identifiability in transformers. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJg1f6EFDB.

S. Bubeck, R. Eldan, and J. Lehec. Sampling from a log-concave distribution with projected langevin monte carlo. *Discrete & Computational Geometry*, 59(4):757–783, 2018.

C. Burns, P. Izmailov, J. H. Kirchner, B. Baker, L. Gao, L. Aschenbrenner, Y. Chen, A. Ecoffet, M. Joglekar, J. Leike, I. Sutskever, and J. Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv: 2312.09390*, 2023.

N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, L. Schubert, C. Voss, B. Egan, and S. K. Lim. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

A. K. Chandra, S. Fortune, and R. Lipton. Unbounded fan-in circuits and associative functions. In *Symposium on Theory of Computing*, 1983.

S. Chaudhary. Code alpaca: An instruction-following llama model for code generation. *GitHub repository*, 2023.

A. Chen, R. Schwartz-Ziv, K. Cho, M. L. Leavitt, and N. Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. *arXiv preprint arXiv:2309.07311*, 2023.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, 2021a.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leiki, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *arXiv:2107.03374*, 2021b.

S. Chen, F. Zhang, K. Sone, and D. Roth. Improving faithfulness in abstractive summarization with contrast candidate generation and selection. *North American Chapter Of The Association For Computational Linguistics*, 2021c. doi: 10.18653/V1/2021.NAACL-MAIN.475.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

V. Chen, J. Li, J. S. Kim, G. Plumb, and A. Talwalkar. Interpretable machine learning: Moving from mythos to diagnostics. *Commun. ACM*, 65(8):43–50, jul 2022. ISSN 0001-0782. doi: 10.1145/3546036. URL https://doi.org/10.1145/3546036.

D. Chiang and P. Cholak. Overcoming a theoretical limitation of self-attention. *ACL*, 2022.

L. Chizat and F. Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In J. Abernethy and S. Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 1305–1338. PMLR, 09-12 Jul 2020. URL https://proceedings.mlr.press/v125/chizat20a.html.

L. Chizat and F. R. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3040–3050, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html.

J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. *arXiv preprint arXiv: 1910.01348*, 2019.

N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In *Studies in Logic and the Foundations of Mathematics*. 1959.

X. Chu, Z. Tian, B. Zhang, X. Wang, X. Wei, H. Xia, and C. Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

B. Chughtai, L. Chan, and N. Nanda. A toy model of universality: Reverse engineering how networks learn group operations. *International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2302.03025.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? An analysis of BERT's attention. In *ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019a.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, Aug. 2019b. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

J. E. Cohen and N. Gillis. Identifiability of complete dictionary learning. *SIAM Journal on Mathematics of Data Science*, 1(3):518–536, 2019.

J. M. Cohen, S. Kaur, Y. Li, J. Z. Kolter, and A. Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *International Conference on Learning Representations*, 2021.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 1989.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ACL*, 2019.

A. Damian, E. Nichani, R. Ge, and J. D. Lee. Smoothing the landscape boosts the signal for sgd: Optimal sample complexity for learning single index models. *Advances in Neural Information Processing Systems*, 36, 2024a.

A. Damian, L. Pillaud-Vivien, J. D. Lee, and J. Bruna. Computational-statistical gaps in gaussian single-index models. *Annual Conference Computational Learning Theory*, 2024b. doi: 10.48550/arXiv.2403.05529.

A. Daniely. Depth separation for neural networks. In *Conference on Learning Theory*, pages 690–696. PMLR, 2017.

T. Dao, G. M. Kamath, V. Syrgkanis, and L. Mackey. Knowledge distillation as semiparametric inference, 2021.

T. Dao, D. Y. Fu, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

G. Dar, M. Geva, A. Gupta, and J. Berant. Analyzing transformers in embedding space, 2022.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

G. Delétang, A. Ruoss, J. Grau-Moya, T. Genewein, L. K. Wenliang, E. Catt, M. Hutter, S. Legg, and P. A. Ortega. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Y. Deng, Z. Li, and Z. Song. Attention scheme inspired softmax regression, 2023.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018a.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018b.

B. Dhingra, M. Faruqui, A. Parikh, M.-W. Chang, D. Das, and W. Cohen. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy, jul 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1483. URL https://aclanthology.org/P19-1483.

Y. Dong, J. Cordonnier, and A. Loukas. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2793–2803. PMLR, 2021. URL http://proceedings.mlr.press/v139/dong21a.html.

D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

I. Drori, S. Zhang, R. Shuttleworth, L. Tang, A. Lu, E. Ke, K. Liu, L. Chen, S. Tran, N. Cheng, R. Wang, N. Singh, T. L. Patti, J. Lynch, A. Shporer, N. Verma, E. Wu, and G. Strang. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 2022.

S. Du and W. Hu. Width provably matters in optimization for deep linear neural networks. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2019.

C. Dyer. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*, 2014.

N. Dziri, A. Madotto, O. Zaiane, and A. Bose. Neural path hunter: Reducing hallucination in dialogue systems via path grounding. *Conference On Empirical Methods In Natural Language Processing*, 2021. doi: 10.18653/v1/2021.emnlp-main.168.

N. Dziri, S. Milton, M. Yu, O. R. Zaiane, and S. Reddy. On the origin of hallucinations in conversational models: Is it the datasets or the models? *North American Chapter Of The Association For Computational Linguistics*, 2022. doi: 10.48550/arXiv.2204.07931.

J. Ebrahimi, D. Gelda, and W. Zhang. How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.384. URL https://aclanthology.org/2020.findings-emnlp.384.

W. Eccles and F. Jordan. A trigger relay utilizing three-electrode thermionic vacuum tubes. *The Electrician*, 83:298, 1919.

W. H. Eccles and F. W. Jordan. Improvements in ionic relays. *British patent number: GB*, 148582:704, 1918.

B. Edelman. Combinatorial tasks as model systems of deep learning. *Doctoral dissertation, Harvard University Graduate School of Arts and Sciences*, 2024.

B. L. Edelman, S. Goel, S. Kakade, and C. Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, 2022.

B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv:2309.03800*, 2023.

A. Egri-Nagy and C. L. Nehaniv. Computational holonomy decomposition of transformation semigroups. *arXiv:1508.06345*, 2015.

S. Eilenberg. *Automata, languages, and machines*. Academic Press, 1974.

R. Eldan and Y. Li. Tinystories: How small can language models be and still speak coherent english?, 2023.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR, 2016.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://Transformer-circuits.pub/2021/framework/index.html.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8: 399–404, 1956. doi: 10.4153/CJM-1956-045-5.

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference On Learning Representations*, 2018.

M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 1984.

R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7518–7528, 2020.

Y. Gao, S. Mahadevan, and Z. Song. An over-parameterized exponential regression, 2023.

S. Garg, D. Tsipras, P. S. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

P. Georgiev, F. Theis, and A. Cichocki. Sparse component analysis and blind source separation of underdetermined mixtures. *IEEE transactions on neural networks*, 16(4):992–996, 2005.

F. Gers and J. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE transactions on neural networks*, 12 6:1333–40, 2001.

B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv: 1904.12191*, 2019.

A. Giannou, S. Rajput, J.-y. Sohn, K. Lee, J. D. Lee, and D. Papailiopoulos. Looped transformers as programmable computers. *arXiv preprint arXiv:2301.13196*, 2023.

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1v4N2l0-`.

M. Glasgow. Sgd finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the xor problem. *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2309.15111.

M. Glasgow. SGD finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the XOR problem. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=HgOJlxzB16.

E. Gobet. Lan property for ergodic diffusions with discrete observations. In *Annales de l'IHP Probabilités et statistiques*, volume 38, pages 711–737, 2002.

S. Goel, V. Kanade, A. Klivans, and J. Thaler. Reliably learning the ReLU in polynomial time. In *Conference on Learning Theory*, 2017.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

P. Gopalani, E. S. Lubana, and W. Hu. How do transformers fill in the blanks? a case study on matrix completion. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

A. Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

C. Grimsley, E. Mayfield, and J. R.S. Bursten. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1780–1790, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.220.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv: 2312.00752*, 2023.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv: 2111.00396*, 2021.

J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher. Non-autoregressive neural machine translation. *arXiv:1711.02281*, 2017.

S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. D. Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, H. S. Behl, X. Wang, S. Bubeck, R. Eldan, A. T. Kalai, Y. T. Lee, and Y. Li. Textbooks are all you need. *arXiv preprint arXiv: 2306.11644*, 2023.

M. Guo, J. Ainslie, D. C. Uthus, S. Ontañón, J. Ni, Y. Sung, and Y. Yang. Longt5: Efficient text-to-text transformer for long sequences. In M. Carpuat, M. de Marneffe, and I. V. M. Ruíz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 724–736. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-naacl.55. URL https://doi.org/10.18653/v1/2022.findings-naacl.55.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010a.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010b.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(2), 2012.

J. Haab, N. Deutschmann, and M. R. Martínez. Is attention interpretation? a quantitative assessment on sets. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 303–321, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-23618-1.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv:1912.01603*, 2019.

M. Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

H. Hälvä and A. Hyvarinen. Hidden markov nonlinear ica: Unsupervised learning from nonstationary time series. In *Conference on Uncertainty in Artificial Intelligence*, pages 939–948. PMLR, 2020.

Y. Hao, D. Angluin, and R. Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10, 2022.

J. Z. HaoChen, C. Wei, A. Gaidon, and T. Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34, 2021.

T. E. Harris and F. S. D. Ross. Fundamentals of a method for evaluating rail net capacities. 1955. URL https://api.semanticscholar.org/CorpusID:107689892.

R. Harshman. Foundations of the parafac procedure: Model and conditions for an explanatory factor analysis. *Technical Report UCLA Working Papers in Phonetics 16, University of California, Los Angeles, Los Angeles, CA*, 1970.

H. Harutyunyan, A. S. Rawat, A. K. Menon, S. Kim, and S. Kumar. Supervision complexity and its role in knowledge distillation. In *The Eleventh International Conference on Learning Representations*, 2022.

H. Harutyunyan, A. S. Rawat, A. K. Menon, S. Kim, and S. Kumar. Supervision complexity and its role in knowledge distillation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8jU7wy7N7mA.

A. Haviv, O. Ram, O. Press, P. Izsak, and O. Levy. Transformer language models without positional encodings still learn positional information. *arXiv:2203.16634*, 2022.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

T. He, J. Zhang, Z. Zhou, and J. R. Glass. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? *Conference On Empirical Methods In Natural Language Processing*, 2019. doi: 10.18653/v1/2021.emnlp-main.415.

O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020.

C. Hertrich, A. Basu, M. D. Summa, and M. Skutella. Towards lower bounds on the depth of ReLU neural networks. In *Advances in Neural Information Processing Systems*, 2021.

J. Hewitt and P. Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275.

J. Hewitt and C. D. Manning. A structural probe for finding syntax in word representations. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL https://aclanthology.org/N19-1419.

J. Hewitt and C. D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019b.

J. Hewitt, M. Hahn, S. Ganguli, P. Liang, and C. D. Manning. RNNs can generate bounded hierarchical languages with optimal memory. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

W. D. Hillis and G. L. Steele Jr. Data parallel algorithms. *Communications of the ACM*, 1986.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv: 1503.02531*, 2015.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997a.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997b. doi: 10.1162/neco.1997.9.8.1735.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models. *arXiv preprint arXiv: 2203.15556*, 2022.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 1989.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *arXiv:1801.06146*, 2018.

J. Hron, Y. Bahri, J. Sohl-Dickstein, and R. Novak. Infinite attention: Nngp and ntk for deep attention networks. *arXiv preprint arXiv: 2006.10540*, 2020.

D. Hsu. Dimension lower bounds for linear approaches to function approximation. 2021. URL https://www.cs.columbia.edu/~djhsu/papers/dimension-argument.pdf.

P. M. Htut, J. Phang, S. Bordia, and S. R. Bowman. Do attention heads in bert track syntactic dependencies?, 2019.

D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-recurrent transformers. *arXiv:2203.07852*, 2022.

A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *arXiv preprint arXiv:1605.06336*, 2016.

A. Hyvarinen and H. Morioka. Nonlinear ICA of Temporally Dependent Stationary Sources. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 460–469. PMLR, 20–22 Apr 2017a. URL https://proceedings.mlr.press/v54/hyvarinen17a.html.

A. Hyvarinen and H. Morioka. Nonlinear ica of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*, pages 460–469. PMLR, 2017b.

A. Jafari, M. Rezagholizadeh, P. Sharma, and A. Ghodsi. Annealing knowledge distillation. *Conference of the European Chapter of the Association for Computational Linguistics*, 2021. doi: 10.18653/v1/2021.eacl-main.212.

S. Jain and B. C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357.

M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

S. Jelassi, M. E. Sander, and Y. Li. Vision transformers provably learn spatial structure. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=eMW9AkXaREI.

S. Jelassi, S. d'Ascoli, C. Domingo-Enrich, Y. Wu, Y. Li, and F. Charton. Length generalization in arithmetic transformers. *arXiv preprint arXiv: 2306.15400*, 2023.

Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

D. Jia, K. Han, Y. Wang, Y. Tang, J. Guo, C. Zhang, and D. Tao. Learning efficient vision transformers via fine-grained manifold distillation. *Neural Information Processing Systems*, 2021.

L. Jin, F. Doshi-Velez, T. Miller, W. Schuler, and L. Schwartz. Unsupervised grammar induction with depth-bounded pcfg. *Transactions of the Association for Computational Linguistics*, 6:211–224, 2018.

X. Jin, B. Peng, Y. Wu, Y. Liu, J. Liu, D. Liang, J. Yan, and X. Hu. Knowledge distillation via route constrained optimization. *IEEE International Conference on Computer Vision*, 2019. doi: 10.1109/ICCV.2019.00143.

M. I. Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

D. Kalimeris, G. Kaplun, P. Nakkiran, B. L. Edelman, T. Yang, B. Barak, and H. Zhang. SGD on neural networks learns functions of increasing complexity. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3491–3501, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/b432f34c5a997c8e7c806a895ecc5e25-Abstract.html.

F. Karlsson. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392, 2007.

J. Kasai, H. Peng, Y. Zhang, D. Yogatama, G. Ilharco, N. Pappas, Y. Mao, W. Chen, and N. A. Smith. Finetuning pretrained transformers into rnns. *arXiv:2103.13076*, 2021.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

G. Ke, D. He, and T.-Y. Liu. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6): 983–1006, 1998.

U. Khandelwal, H. He, P. Qi, and D. Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. *International Conference On Learning Representations*, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. J. Kleitman, B. R. Rothschild, and J. H. Spencer. The number of semigroups of order n. *Proceedings of the American Mathematical Society*, 1976.

G. Kobayashi, T. Kuribayashi, S. Yokoi, and K. Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574.

L. Kong, C. de Masson d'Autume, L. Yu, W. Ling, Z. Dai, and D. Yogatama. A mutual information maximization perspective of language representation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx79eBKwr.

L. Kovács and C. Praeger. Finite permutation groups with large abelian quotients. *Pacific Journal of Mathematics*, 1989.

O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1445. URL https://aclanthology.org/D19-1445.

M. Krasner and L. Kaloujnine. Produit complet des groupes de permutations et probleme d'extension de groupes II. *Acta Scientiarum Mathematicarum*, 1951.

K. Krohn and J. Rhodes. Algebraic theory of machines, I: Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 1965.

J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

G. Lample and F. Charton. Deep learning for symbolic mathematics. *arXiv:1912.01412*, 2019.

J. Lanchantin, S. Toshniwal, J. Weston, A. Szlam, and S. Sukhbaatar. Learning to reason and memorize with self-notes. *arXiv preprint arXiv: Arxiv-2305.00833*, 2023.

H. Lee, R. Ge, T. Ma, A. Risteski, and S. Arora. On the ability of neural nets to express distributions. In *Conference on Learning Theory*, pages 1271–1296. PMLR, 2017.

J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo. Predicting what you already know helps: Provable self-supervised learning. *Advances in Neural Information Processing Systems*, 34, 2021.

N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv: 2307.03381*, 2023.

H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv: 2202.01110*, 2022a.

J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1082. URL https://aclanthology.org/N16-1082.

X. Li and H. Gong. Robust optimization for multilingual translation with imbalanced data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25086–25099. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/d324a0cc02881779dcda44a675fdcaaa-Paper.pdf.

Y. Li and A. Risteski. The limitations of limited context for constituency parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2675–2687, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.208. URL https://aclanthology.org/2021.acl-long.208.

Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. d. M. d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. Sutherland Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals. Competition-level code generation with alphacode. *arXiv:2203.07814*, 2022b.

Y. Li, Y. Li, and A. Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19689–19729. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/li23p.html.

O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, O. Abend, R. Alon, T. Asida, A. Bergman, R. Glozman, M. Gokhman, A. Manevich, N. Ratner, N. Rozen, E. Shwartz, M. Zusman, and Y. Shoham. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv: 2403.19887*, 2024.

H. Lin, G. Han, J. Ma, S. Huang, X. Lin, and S.-F. Chang. Supervised masked knowledge distillation for few-shot transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19649–19659, June 2023.

Y. Lin, Y. C. Tan, and R. Frank. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

B. G. Lindsay and P. Basak. Multivariate normal mixtures: a fast consistent method of moments. *Journal of the American Statistical Association*, 88(422):468–476, 1993.

Z. C. Lipton. The mythos of model interpretability, 2017.

B. Liu, P. Ravikumar, and A. Risteski. Contrastive learning of strong-mixing continuous-time stochastic processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3151–3159. PMLR, 2021.

B. Liu, J. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Transformers learn shortcuts to automata. *International Conference On Learning Representations*, 2022a. doi: 10.48550/arXiv.2210.10749.

B. Liu, D. J. Hsu, P. Ravikumar, and A. Risteski. Masked prediction: A parameter identifiability view. *Advances in Neural Information Processing Systems*, 35:21241–21254, 2022b.

B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Transformers learn shortcuts to automata. *International Conference on Learning Representations*, 2023a. doi: 10.48550/arXiv.2210.10749. URL https://openreview.net/forum?id=De4FYqjFueZ.

B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Exposing attention glitches with flip-flop language modeling. *Neural Information Processing Systems*, 2023b.

E. Liu and G. Neubig. Are representations built from the ground up? an empirical examination of local composition in language models. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9053–9073, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.617. URL https://aclanthology.org/2022.emnlp-main.617.

H. Liu and A. C.-C. Yao. Augmenting math word problems via iterative question composing. *arXiv preprint arXiv:2401.09003*, 2024.

H. Liu, S. M. Xie, Z. Li, and T. Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. *arXiv preprint arXiv:2210.14199*, 2022c.

L. Liu, X. Liu, J. Gao, W. Chen, and J. Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.463. URL https://aclanthology.org/2020.emnlp-main.463.

N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1112. URL https://aclanthology.org/N19-1112.

R. Liu, J. Wei, F. Liu, C. Si, Y. Zhang, J. Rao, S. Zheng, D. Peng, D. Yang, D. Zhou, et al. Best practices and lessons learned on synthetic data for language models. *arXiv preprint arXiv:2404.07503*, 2024.

S. Longpre, K. Perisetla, A. Chen, N. Ramesh, C. DuBois, and S. Singh. Entity-based knowledge conflicts in question answering. *Conference On Empirical Methods In Natural Language Processing*, 2021. doi: 10.18653/v1/2021.emnlp-main.565.

I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017a.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017b.

H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen, and D. Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

O. Maler. On the Krohn-Rhodes cascaded decomposition theorem. In *Time for Verification*. Springer-Verlag, 2010.

O. Maler and A. Pnueli. On the cascaded decomposition of automata, its complexity and its application to logic (Draft). 1994.

N. Malkin, Z. Wang, and N. Jojic. Coherence boosting: When your pretrained language model is not paying enough attention. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8214–8236. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.565. URL https://doi.org/10.18653/v1/2022.acl-long.565.

C. Meister, S. Lazov, I. Augenstein, and R. Cotterell. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 122–129, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.17. URL https://aclanthology.org/2021.acl-short.17.

A. Menon and C. S. Ong. Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning*, pages 304–313. PMLR, 2016.

A. K. Menon, A. S. Rawat, S. Reddi, S. Kim, and S. Kumar. A statistical perspective on distillation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7632–7642. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/menon21a.html.

C. Mereghetti and B. Palano. Threshold circuits for iterated matrix product and powering. *RAIRO-Theoretical Informatics and Applications*, 2000.

W. Merrill. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3901. URL https://www.aclweb.org/anthology/W19-3901.

W. Merrill, Y. Goldberg, R. Schwartz, and N. A. Smith. On the power of saturated Transformers: A view from circuit complexity. *arXiv:2106.16213*, 2021.

W. Merrill, A. Sabharwal, and N. A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample efficient world models. *arXiv:2209.00588*, 2022.

G. N. Milstein, J. G. Schoenmakers, V. Spokoiny, et al. Transition density estimation for stochastic differential equations via forward-reverse representations. *Bernoulli*, 10(2):281–312, 2004.

S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh. Improved knowledge distillation via teacher assistant. *AAAI Conference on Artificial Intelligence*, 2019. doi: 10.1609/AAAI.V34I04.5963.

A. Mitra, H. Khanpour, C. Rosset, and A. Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26:2265–2273, 2013.

A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

H. Mobahi, M. Farajtabar, and P. Bartlett. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

M. Mohri and A. Rostamizadeh. Rademacher complexity bounds for non-iid processes. In *Advances in Neural Information Processing Systems*, pages 1097–1104, 2009.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT Press, 2012.

D. Morwani, B. L. Edelman, C. Oncescu, R. Zhao, and S. M. Kakade. Feature emergence via margin maximization: case studies in algebraic tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=i9wDX850jR.

E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375, 2005.

A. Mukherjee and A. Basu. Lower bounds over boolean inputs for deep neural networks with ReLU gates. *arXiv:1711.03073*, 2017.

V. Nagarajan. Explaining generalization in deep learning: progress and fundamental limits. *arXiv preprint arXiv: 2110.08922*, 2021.

V. Nagarajan, A. K. Menon, S. Bhojanapalli, H. Mobahi, and S. Kumar. On student-teacher deviations in distillation: does it pay to disobey?, 2024.

N. Nanda and T. Lieberum. A mechanistic interpretability analysis of grokking. *Alignment Forum*, 2022. URL https://www.alignmentforum.org/posts/N6WM6hs7RQMKDhYjB/a-mechanistic-interpretability-analysis-of-grokking.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, Nov. 2-3 2019. Association for Computational Linguistics. URL https://aclanthology.org/2019.iwslt-1.17.

E. Nichani, Y. Bai, and J. D. Lee. Identifying good directions to escape the ntk regime and efficiently learn low-degree plus sparse polynomials. *Advances in Neural Information Processing Systems*, 35: 14568–14581, 2022.

E. Nichani, A. Damian, and J. D. Lee. How transformers learn causal structure with gradient descent. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jNM4imlHZv.

R. Nogueira, Z. Jiang, and J. Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv:2102.13019*, 2021.

M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv:2112.00114*, 2021a.

M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021b.

R. O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting recurrent neural networks for long sequences. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2303.06349.

D. Pareek, S. S. Du, and S. Oh. Understanding the gains from repeated self-distillation. *arXiv preprint arXiv: 2407.04600*, 2024.

A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das. Totto: A controlled table-to-text generation dataset. *Conference On Empirical Methods In Natural Language Processing*, 2020. doi: 10.18653/v1/2020.emnlp-main.89.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. K"opf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 2019.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv: 2305.13048*, 2023.

A. Pensia, S. Rajput, A. Nagle, H. Vishwakarma, and D. Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in neural information processing systems*, 33:2599–2610, 2020.

J. Pérez, P. Barceló, and J. Marinkovic. Attention is turing complete. *The Journal of Machine Learning Research*, 22(1):3463–3497, 2021.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv:1802.05365*, 2018.

F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? *Conference On Empirical Methods In Natural Language Processing*, 2019. doi: 10.18653/v1/D19-1250.

S. Polu and I. Sutskever. Generative language modeling for automated theorem proving. *arXiv:2009.03393*, 2020.

S. Prasanna, A. Rogers, and A. Rumshisky. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.259. URL https://aclanthology.org/2020.emnlp-main.259.

O. Press, N. A. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

S. Purushwalkam and A. Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

L. Quirke, L. Heindrich, W. Gurnee, and N. Nanda. Training dynamics of contextual n-grams in language models. *arXiv preprint arXiv:2311.00863*, 2023.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019a.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019b.

A. Raganato and J. Tiedemann. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5431. URL https://aclanthology.org/W18-5431.

P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

J. H. Reif and S. R. Tate. On threshold circuits and polynomial computation. *SIAM Journal on Computing*, 1992.

L. Ren, Y. Liu, Y. Lu, Y. Shen, C. Liang, and W. Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv: 2406.07522*, 2024.

Y. Ren, S. Guo, and D. J. Sutherland. Better supervisory signals by observing learning paths. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Iog0djAdbHj.

B. Rhodes, K. Xu, and M. U. Gutmann. Telescoping density-ratio estimation. *arXiv preprint arXiv:2006.12204*, 2020.

J. Rhodes, C. L. Nehaniv, and M. W. Hirsch. *Applications of automata theory and algebra: via the mathematical theory of complexity to biology, physics, psychology, philosophy, and games*. World Scientific, 2010.

L. Riou-Durand, N. Chopin, et al. Noise contrastive estimation: Asymptotic properties, formal comparison with mc-mle. *Electronic Journal of Statistics*, 12(2):3473–3518, 2018.

J. Robinson, L. Sun, K. Yu, K. Batmanghelich, S. Jegelka, and S. Sra. Can contrastive learning avoid shortcut solutions? *Advances in Neural Information Processing Systems*, 2021.

A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

I. Safran, R. Eldan, and O. Shamir. Depth separations in neural networks: what is actually being separated? In *Conference on Learning Theory*, pages 2664–2666. PMLR, 2019.

G. Samorodnitsky et al. Long range dependence. *Foundations and Trends® in Stochastic Systems*, 1(3): 163–257, 2007.

C. Sanford, D. Hsu, and M. Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=QCZabhKQhB.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

A. Saparov and H. He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *International Conference On Learning Representations*, 2022. doi: 10.48550/arXiv.2210.01240.

Y. Sarrof, Y. Veitsman, and M. Hahn. The expressive capacity of state space models: A formal language perspective. *arXiv preprint arXiv: 2405.17394*, 2024.

N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637, 2019.

N. Saunshi, S. Malladi, and S. Arora. A mathematical exploration of why language models help solve downstream tasks. In *International Conference on Learning Representations*, 2020.

R. E. Schapire. The strength of weak learnability. *Machine learning*, 5:197–227, 1990.

C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

T. Schuster, A. Kalyan, A. Polozov, and A. Kalai. Programming puzzles. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 1965.

A. Schwarzschild, E. Borgnia, A. Gupta, F. Huang, U. Vishkin, M. Goldblum, and T. Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, 2021.

M. Schützenberger. On context-free languages and push-down automata. *Information and Control*, 6(3):246–264, 1963. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(63)90306-1. URL https://www.sciencedirect.com/science/article/pii/S0019995863903061.

S. Serrano and N. A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1282. URL https://aclanthology.org/P19-1282.

Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2302.00618.

N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. Chi, N. Schärli, and D. Zhou. Large language models can be easily distracted by irrelevant context. *arXiv preprint arXiv:2302.00093*, 2023.

W. Shi, Y. Song, H. Zhou, B. Li, and L. Li. Follow your path: a progressive method for knowledge distillation, 2021.

H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 440–449, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130432. URL https://doi.org/10.1145/130385.130432.

M. Soltanolkotabi, A. Javanmard, and J. D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

D. Soudry and E. Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv preprint arXiv: 1702.05777*, 2017.

D. A. Spielman, H. Wang, and J. Wright. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, pages 37–1. JMLR Workshop and Conference Proceedings, 2012.

A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, USA, 1st edition, 2012. ISBN 0521190177.

S. Sukhbaatar, D. Ju, S. Poff, S. Roller, A. D. Szlam, J. Weston, and A. Fan. Not all memories are created equal: Learning to forget by expiring. *International Conference On Machine Learning*, 2021.

K. Sun and A. Marasović. Effective attention sheds light on interpretability. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4126–4135, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.361. URL https://aclanthology.org/2021.findings-acl.361.

S. Sun, K. Krishna, A. Mattarella-Micke, and M. Iyyer. Do long-range language models actually use long-range context? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 807–822, Online and Punta Cana, Dominican Republic, nov 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.62. URL https://aclanthology.org/2021.emnlp-main.62.

R. Sutton. The bitter lesson. 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

M. Suzgun, Y. Belinkov, S. Shieber, and S. Gehrmann. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL https://www.aclweb.org/anthology/W19-3905.

A. Tamkin, V. Liu, R. Lu, D. Fein, C. Schultz, and N. Goodman. Dabs: A domain-agnostic benchmark for self-supervised learning. *arXiv preprint arXiv:2111.12062*, 2021.

R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez,

P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv: 2403.08295*, 2024.

M. Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *ACL*, 2019.

R. Tian, S. Narayan, T. Sellam, and A. P. Parikh. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv: 1910.08684*, 2019.

Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding, 2020a.

Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020b.

Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839, 2020c.

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive learning, multi-view redundancy, and linear models, 2020a.

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive estimation reveals topic posterior information to linear models. *arXiv preprint arXiv:2003.02234*, 2020b.

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive estimation reveals topic posterior information to linear models. *arXiv preprint arXiv:2003.02234*, 2020c.

C. Tosh, A. Krishnamurthy, and D. Hsu. Contrastive learning, multi-view redundancy, and linear models, 2020d.

H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers and distillation through attention. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/touvron21a.html`.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

C.-P. Tsai, A. Prasad, S. Balakrishnan, and P. Ravikumar. Heavy-tailed streaming statistical estimation. *arXiv preprint arXiv:2108.11483*, 2021.

M. Uehara, T. Kanamori, T. Takenouchi, and T. Matsuda. A unified statistically efficient estimation framework for unnormalized models. In *International Conference on Artificial Intelligence and Statistics*, pages 809–819. PMLR, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. A. Hechtman, and J. Shlens. Scaling local self-attention for parameter efficient visual backbones. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

J. Vig. Visualizing attention in transformer-based language representation models. *arXiv:1904.02679*, 2019.

J. Vig and Y. Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4808. URL https://aclanthology.org/W19-4808.

E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

J. Von Kügelgen, Y. Sharma, L. Gresele, W. Brendel, B. Schölkopf, M. Besserve, and F. Locatello. Self-supervised learning with data augmentations provably isolates content from style. *Advances in neural information processing systems*, 34:16451–16467, 2021.

M. Vuffray, S. Misra, A. Lokhov, and M. Chertkov. Interaction screening: Efficient and sample-optimal learning of ising models. *Advances in neural information processing systems*, 29, 2016.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022a.

K. R. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020a.

T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020b.

X. Wang, X. Chen, S. S. Du, and Y. Tian. Towards demystifying representation learning with non-contrastive self-supervision. *arXiv preprint arXiv:2110.04947*, 2021.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv: Arxiv-2203.11171*, 2022b.

Z. Wang, S. Wei, D. Hsu, and J. D. Lee. Transformers provably learn sparse token selection while fully-connected nets cannot. *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2406.06893.

C. Wei, J. D. Lee, Q. Liu, and T. Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Wei, S. M. Xie, and T. Ma. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *Advances in Neural Information Processing Systems*, 34, 2021.

C. Wei, Y. Chen, and T. Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022a.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv:2201.11903*, 2022c.

G. Weiss, Y. Goldberg, and E. Yahav. On the practical computational power of finite precision rnns for language recognition. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 740–745. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-2117. URL https://aclanthology.org/P18-2117/.

G. Weiss, Y. Goldberg, and E. Yahav. Thinking like transformers. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

K. Wen, Y. Li, B. Liu, and A. Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars. *Neural Information Processing Systems*, 2023.

K. Wen, X. Dang, and K. Lyu. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. *arXiv preprint arXiv: 2402.18510*, 2024.

Z. Wen and Y. Li. Toward understanding the feature learning process of self-supervised contrastive learning. *International Conference on Machine Learning*, page 11112–11122, 2021.

S. Wiegreffe and Y. Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1002. URL https://aclanthology.org/D19-1002.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*, 2019.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016.

Y. Wu, M. N. Rabe, W. Li, J. Ba, R. B. Grosse, and C. Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, pages 11251–11262. PMLR, 2021.

Y. Wu, M. N. Rabe, D. S. Hutchins, and C. Szegedy. Memorizing transformers. *International Conference On Learning Representations*, 2022. doi: 10.48550/arXiv.2203.08913.

Z. Wu, Y. Chen, B. Kao, and Q. Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.383. URL https://aclanthology.org/2020.acl-main.383.

Y. Xiao, L. Wu, J. Guo, J. Li, M. Zhang, T. Qin, and T.-y. Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *arXiv:2204.09269*, 2022.

S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

K. Xu, M. Zhang, J. Li, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv:2009.11848*, 2020.

G. Yang and E. J. Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv: 2011.14522*, 2020.

G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv: 2203.03466*, 2022.

S. Yang, B. Wang, Y. Zhang, Y. Shen, and Y. Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv: 2406.06484*, 2024.

S. Yao, B. Peng, C. Papadimitriou, and K. Narasimhan. Self-attention networks can process bounded hierarchical languages. *arXiv preprint arXiv:2105.11115*, 2021a.

S. Yao, B. Peng, C. H. Papadimitriou, and K. Narasimhan. Self-attention networks can process bounded hierarchical languages. In *Association for Computational Linguistics*, 2021b.

W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 2021.

T. Yi, D. Mostafa, B. Dara, and M. Donald. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2021.

B. Yu. Rates of convergence for empirical processes of stationary mixing sequences. *The Annals of Probability*, pages 94–116, 1994.

L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

S. Yu, T. Chen, J. Shen, H. Yuan, J. Tan, S. Yang, J. Liu, and Z. Wang. Unified visual transformer compression. *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2203.08243.

L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3903–3911, 2020.

X. Yue, X. Qu, G. Zhang, Y. Fu, W. Huang, H. Sun, Y. Su, and W. Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

H. P. Zeiger. Cascade synthesis of finite-state machines. *Information and Control*, 1967.

X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

C. Zhang, M. Raghu, J. Kleinberg, and S. Bengio. Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *arXiv:2107.12580*, 2021a.

J. Zhang, Y. Zhao, H. Li, and C. Zong. Attention with sparsity regularization for neural machine translation and summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):507–518, 2018.

L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021b.

M. Zhang, H. Marklund, N. Dhawan, A. Gupta, S. Levine, and C. Finn. Adaptive risk minimization: Learning to adapt to domain shift. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23664–23678, 2021c. URL https://proceedings.neurips.cc/paper/2021/hash/c705112d1ec18b97acac7e2d63973424-Abstract.html.

S. D. Zhang, C. Tigges, S. Biderman, M. Raginsky, and T. Ringer. Can transformers learn to solve problems recursively? *arXiv preprint arXiv: 2305.14699*, 2023.

T. Zhang and T. Hashimoto. On the inductive bias of masked language modeling: From statistical to syntactic dependencies. *arXiv preprint arXiv:2104.05694*, 2021.

Y. Zhang, A. Backurs, S. Bubeck, R. Eldan, S. Gunasekar, and T. Wagner. Unveiling Transformers with LEGO: a synthetic reasoning task. *arXiv:2206.04301*, 2022.

H. Zhao, A. Panigrahi, R. Ge, and S. Arora. Do transformers parse while predicting the masked word? *Conference on Empirical Methods in Natural Language Processing*, 2023. doi: 10.48550/arXiv. 2303.08117.

C. Zheng, Z. Liu, E. Xie, Z. Li, and Y. Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv.2304.09797*, 2023a.

K. Zheng and E.-H. Yang. Knowledge distillation based on transformed teacher matching. *arXiv preprint arXiv: 2402.11148*, 2024.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023b.

Z. Zhong, Z. Liu, M. Tegmark, and J. Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *arXiv preprint arXiv: 2306.17844*, 2023.

H. Zhou, L. Song, J. Chen, Y. Zhou, G. Wang, J. Yuan, and Q. Zhang. Rethinking soft labels for knowledge distillation: A bias–variance tradeoff perspective. In *International Conference on Learning Representations*, 2020.

H. Zhou, A. Nova, H. Larochelle, A. Courville, B. Neyshabur, and H. Sedghi. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022.

K.-H. Zimmermann. On Krohn-Rhodes theory for semiautomata. *arXiv:2010.16235*, 2020.

R. S. Zimmermann, Y. Sharma, S. Schneider, M. Bethge, and W. Brendel. Contrastive learning inverts the data generating process. In *International Conference on Machine Learning*, pages 12979–12990. PMLR, 2021.