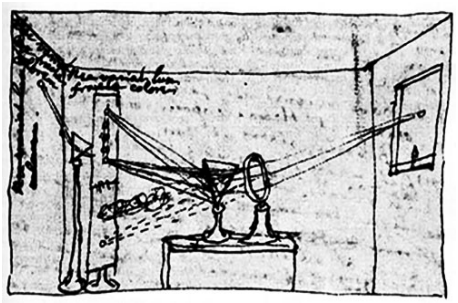
GENERALIZATION THROUGH RICHER SUPERVISION

ALEXANDER C. LI



May 2025
CMU-ML-25-103

Machine Learning Department
School of Computer Science
Carnegie Mellon University

THESIS COMMITTEE:
Deepak Pathak, Chair
Zico Kolter
Ruslan Salakhutdinov
Alexei A. Efros (UC Berkeley)

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

This research was sponsored by Defense Advanced Research Projects Agency contract number N660011924034 and National Science Foundation Graduate Research Fellowship Program award numbers 1745016 and 2140739.

Keywords: machine learning, deep learning, generalization, generative models, distribution shift, diffusion models, language models, Transformer, exploration, data curation, Internet, self-supervised learning, contrastive learning

ACKNOWLEDGMENTS

I'm deeply grateful to my advisor, Deepak Pathak, for his continuous guidance throughout my PhD. Alyosha Efros has also been instrumental in my research journey, providing valuable insights and collaboration since the start of my PhD. They've helped me grow so much as a researcher, from scrutinizing the smallest details ("look at the pixels!") to thinking big, no matter how crazy the idea. I'm also very grateful for my undergraduate advisor, Pieter Abbeel, as well as my undergraduate mentors Lerrel Pinto and Carlos Florensa, for inspiring me to love research and getting me started on this journey. Finally, thank you to my committee members Zico Kolter and Ruslan Salakhutdinov for thoughtful comments and fruitful discussions throughout the years.

I'm lucky to have worked with such an incredible set of collaborators during my PhD: Ellis Brown, Mihir Prabhudesai, Shivam Duggal, Tsung-Wei Ke, Katerina Fragkiadaki, Ananya Kumar, Xinlei Chen, Yuandong Tian, and Beidi Chen. I'd also like to thank my labmates Russell Mendonca, Shikhar Bahl, Kenny Shaw, Ananye Agarwal, Murtaza Dalal, and Lili Chen for many wonderful conversations. Thank you to all the researchers in the Machine Learning Department and Robotics Institute, who have fostered such a vibrant community at CMU. I'm grateful for my friends Rishi Veerapaneni, Patrick Chao, and many more who have provided much-needed balance, perspective, and laughter. I'd also like to thank Christina Baek for her constant support, endless patience, and ability to bring joy during the most challenging moments of this journey. Most importantly, I thank my family. Their unwavering love and support has made everything possible.

ABSTRACT

Machine learning methods have achieved superhuman performance in limited domains, but they still fall short in their ability to generalize to new scenarios. In this thesis, we suggest that this shortcoming stems from the fact that training objectives are *too easy*. Models can quickly overfit to training examples without learning general principles that apply broadly. We propose addressing this issue by recasting problems to be *richer and more complex*, encouraging models to understand the underlying structure rather than memorizing surface statistics. We present three complementary solutions: autonomous data acquisition, better learning objectives, and careful algorithm design.

Chapter 2 develops a curiosity-based system that continually learns by searching the Internet for data that it is least knowledgeable about. Our Internet Explorer method learns to generate targeted Google queries and selectively trains on retrieved data, outperforming alternatives while requiring 32x less training time and 180x less data. Chapters 3 and 4 explore how to extract more value from existing datasets by revisiting generative classifiers, which have the harder task of modeling both inputs and labels jointly. By implementing these classifiers with modern generative architectures, we achieve significant improvements in compositional reasoning and out-of-distribution generalization. Finally, Chapter 5 advances sequence modeling algorithms by recasting masked discrete diffusion as a generalization of autoregressive models and developing efficient architectures for any-order generative modeling. This approach enables training on all possible sequence permutations, resulting in better performance on algorithmic reasoning tasks as well as substantially lower sampling latency.

CONTENTS

1

INTRODUCTION

Machine learning methods have achieved superhuman performance in limited domains like game-playing [172], competition programming [110], and protein folding [89]. However, in more realistic settings, models exhibit a puzzling failure to generalize. Object detection models are overly sensitive to tiny changes in position [155]; classification models are susceptible to background changes or small corruptions [14, 75]; the list of failures is endless. Failure to generalize can be life-threatening: models trained to recognize pneumonia or cancer fail on images from new hospitals [2, 205], and self-driving cars can crash when encountering a novel road condition [149]. Naively training on more data, up to hundreds of billions of words, can even hurt generalization [121]. Why do models fail to generalize, and how can we prevent this?

Consider a toy classification task where the objective is to classify whether an image shows a camel or a cow. This appears easy enough to humans. However, models can learn pathologically bad solutions, as their training objective only requires them to confidently and correctly predict the training set. First, models can overfit, using features that work on the training data but don't generalize to the validation dataset [109, 188]. Second, models can also rely on shortcut solutions [57], which appear to generalize to the validation set but actually utilize features that are not causally related to the label. Shortcut solutions, *e.g.*, background color or high-frequency texture [58, 160], can then cause huge drops in performance under distribution shift, when these shortcuts no longer hold. Notably, models can learn to rely on shortcut solutions even when they *aren't perfectly predictive on the training set*. In the cow versus camel example, the training set contains a strong correlation between the label and the background color of sand versus grass, though there are counterexamples like the cow on the beach or the camel in the grasslands.

Figure 1.1: Camel vs cow toy classification task

Models are still perfectly "happy" to use the background to get most of the training examples correct, then overfit to the remaining difficult examples where this cue doesn't hold.

This suggests that our models' failure to generalize may stem from the fact that our training objective is too *easy*: there are too many solutions that just don't generalize. Even if there are generalizing solutions (in this case, animal shape) that work on all training and test examples, models have little incentive to find them during training. Perhaps, then, we need to make our problems *harder* in order to get the learning signal necessary for models to better generalize in- and out-of-distribution.

In this thesis, we explore three types of approaches to train our models with *richer and more complex supervision*. First, we develop a method to autonomously acquire hard data for training. Next, we revisit the paradigm of generative classifiers as a better training objective. Finally, we explore generalizations of autoregressive models as new algorithms for language modeling.

1.1 AUTONOMOUSLY ACQUIRING HARD DATA

One of the most successful approaches for improving generalization is to pretrain on a much broader data distribution [96, 146, 147], before finetuning on a desired target task. Fang et al. [52] show that the diversity of the pretraining set primarily drives the increase in performance. This makes sense: as we scale the number of "rare" examples, it becomes harder to overfit to all of them, and it also becomes more likely that we'll see similar training examples to any test point. Thus, a reliable approach to improve performance has been to continuously scale up pretraining datasets. However, such approaches require scaling up the amount of compute to the point that training runs last for months and cost millions of dollars. This is impractical for most applications and researchers.

In Chapter 2, we propose an alternate approach. If we know the target task that we care about, we shouldn't waste training compute on examples that are too easy or irrelevant. Instead, we should focus on collecting as much hard, relevant data as possible in order to train a specialized model. Our Internet Explorer method has access to all of the data on the Internet and learns to make queries that yield useful training images for self-supervised representation learning. It determines this by identifying the downloaded images that increase the self-supervised training loss on the target task images the most. This process is similar to active learning [40], but Internet Explorer only requires a small, unlabeled set of images from our target task and does not need labels. We show that this is highly effective: Internet Explorer outperforms CLIP [147] with 32x less training time and 180x less data.

1.2 HARDER LEARNING OBJECTIVES

While Chapter 2 focuses on improving generalization by finding hard training data examples that provide more learning signal, we can also improve generalization given a fixed amount of data if we train on a more difficult objective. There are many popular techniques for doing so, such as data augmentation to enforce certain invariances [203, 207] or regularization to reduce overfitting [127, 178]. Self-supervised objectives like CPC [134], MoCo [71], DINO [25], and MAE [70] force the model to differentiate between images or reconstruct missing patches, and are designed to encourage the model to learn features that are also useful for the supervised task.

While these methods do improve generalization, they have their limitations. It's difficult to hand-design effective augmentation strategies, since we need to find a balance between adding invariances and keeping the transformed data similar enough to the original distribution. Furthermore, there are many invariances that are difficult to easily enforce via augmentation. For example, it's not clear how to transform images so that models avoid using a watermark shortcut feature [111] on ImageNet. Regularization is useful but limited, as models still often find a way to overfit or learn shortcuts. Finally, auxiliary self-supervised objectives have to be carefully tuned [29], otherwise their learned features are too irrelevant for the task at hand.

These techniques are particularly ineffective for improving out-of-distribution generalization *trends*. Taori et al. [181] and Miller et al. [125] evaluate the in-distribution (ID) and out-of-distribution (OOD) accuracy of hundreds of models, trained on the same data with varying augmentation, regularization, and auxiliary objective. Surprisingly, if graphed with ID accuracy on the x-axis and OOD accuracy on the y-axis, on most distribution shifts the performances of all of these models fall on a single line. This is concerning: all methods so far only improve OOD accuracy by improving ID accuracy, and none have a better scaling trend for OOD accuracy! The only approach to get OOD performance above the trend line is to train on additional data, which is often infeasible.

We suggest that the issue with all of these previous approaches is that they're trying to compensate for a weak discriminative objective. Minimizing the cross-entropy loss $-\log p_\theta(y^* \mid x)$ is too easy. Going back to the camel versus cow example, the model has little incentive to learn the causal "shape" features if it can minimize loss just via an easy-to-learn background feature and overfitting on the examples where this shortcut doesn't hold. Even if the model has somehow learned the causal features, models trained with cross-entropy often still prefer to use the spurious features [93, 128, 156, 177].

Motivated by the idea that the discriminative objective $-\log p_\theta(y \mid x)$ is too easy, we revisit the idea of generative classifiers, which train models to learn the condi-

tional likelihood $p_\theta(x \mid y)$. This is a much harder objective, as it requires modeling the entire input x, conditioned on the label y. This is desirable. In the camel versus cow example, a generative classifier cannot just learn to model the grass and sand. It must eventually understand how to generate the shapes of each animal. And once it learns the causal shape features, it should prefer to use them since they are the most consistently correlated with the label.

Generative classifiers are not new, dating at least as far back as Fischer discriminant analysis [55]. However, they had fallen in popularity, likely due to the lack of strong generative modeling algorithms at the time. Today, though, we have powerful generative models for multiple modalities: diffusion models [76, 173] for images and autoregressive Transformers [190] for text. Thus, this is a good time to design efficient and accurate classifiers using these new generative models and explore their learning advantages. In Chapter 3, we propose Diffusion Classifier, a way to turn class-conditional diffusion models into strong classifiers. We find that Diffusion Classifier does especially well on harder tasks like compositional reasoning. In Chapter 4, we show that generative classifiers do indeed generalize better on distribution shifts. In fact, we find that generative classifiers are the first approach to have a qualitatively better ID vs OOD trend line ("effective robustness") without using any additional training data. These findings hold across image, text, and low-dimensional Gaussian benchmarks.

1.3 Dynamically choosing problem formulation at test-time

Finally, we turn our attention to how we can use richer supervision to improve language modeling. Today's most ubiquitous language models are autoregressive [21, 186], predicting text one word at a time from left-to-right by modeling $p_\theta(x_t \mid x_{1:t})$. This formulation dates back to Shannon [170], which introduced language modeling using n-grams. Since then, RNNs [158], LSTMs [79], and Transformer [190] text generation models have all been autoregressive, though there have been alternate algorithms proposed [59, 187]. Left-to-right autoregressive prediction is easy to train, works well on natural language, and easily handles variable-length sequences. For Transformers in particular, autoregressive prediction also enables efficient sampling due to key-value caching.

However, autoregression in a fixed order has drawbacks. First, it may hurt generalization. Some problems are hard to solve from left-to-right, but easy to solve in a different ordering. For example, Sudoku is relatively easy to solve when filling in the next most constrained square, but essentially impossible if you have to solve it from left-to-right, top-to-bottom. Sampling from autoregressive models is also slow, as we

can only generate a single token per forward pass. This could be a big drawback in latency-sensitive scenarios.

Discrete diffusion models are a promising alternative generative modeling framework [6, 173] that could solve these problems. These models start with a sequence that has been corrupted with noise and learn to gradually denoise it step by step. Unlike continuous diffusion used in image generation, discrete diffusion operates on discrete tokens and uses various corruption processes such as masking tokens, replacing them with random tokens, or applying more complex transitions. Recent approaches have shown promise but still face challenges in training efficiency and inference speed compared to traditional autoregressive models [6, 117, 162]. Despite these challenges, discrete diffusion models enable flexible conditioning, parallel token generation, and potentially improved reasoning capabilities.

In Chapter 5, we provide new insights into discrete diffusion that lead to better generalization and more efficient models. We first establish a theoretical equivalence between absorbing discrete diffusion and any-order autoregressive models. We propose Prism, our improved discrete diffusion language model, and train it as an autoregressive model over random permutations of each sequence. This costs more to pretrain than autoregressive models, which only have to learn the single left-to-right ordering, but this extra training supervision has several advantages. Given a test prompt, Prism can use a confidence-based decoding strategy to dynamically decide which order it should sample tokens in. This substantially improves performance on algorithmic reasoning tasks over existing models. Training over all data permutations also enables Prism to predict multiple tokens in parallel. This reduces sampling latency by up to $4-8\times$.

Part I

AUTONOMOUS DATA ACQUISITION

2

INTERNET EXPLORER: TARGETED REPRESENTATION LEARNING ON THE OPEN WEB

Suppose you have a small dataset and need to train a model for some task, say classification. A pipeline that has become standard today is to download the latest pre-trained deep network and fine-tune it on your own small dataset. This pre-trained model used to be ImageNet-based [43, 72] and now would probably be CLIP [146]. The implicit goal set by the community for such pre-trained models is that they should transfer well to any kind of downstream task not known in advance. This has led to a race to build ultra-large-scale models in terms of computation, model size, and dataset size. But is this goal of building an "omniscient" pre-trained model that can work on any future downstream task even feasible? Perhaps not, as our world is continually changing. Although the size of the pretraining datasets has grown from 1.2M [43] to 5B [166] images, what has not changed at all is their nature: these datasets are curated and, more importantly, ***static***. For instance, the portion of ImageNet curated before 2007 has no idea what an iPhone is. Furthermore, although a few hundred million images represent a staggering quantity of visual data, they are minuscule compared to the entire Internet, where billions of new photos are uploaded every day. Thus, current static datasets, however big they become, fail to capture the richness and dynamic nature of the data available on the Internet. Moreover, as our static datasets grow, they require increasingly inaccessible amounts of compute.

In this section, we rethink the idea of ***generic*** large-scale pretraining and propose an alternate paradigm: train a small-scale but up-to-date model geared towards the ***specific*** downstream task of interest. To do so, we look beyond static datasets and *treat the Internet itself as a dynamic, open-ended dataset*. Unlike conventional datasets, which are expensive to expand and grow stale with time, the Internet is dynamic, rich, grows automatically, and is always up to date. Its continuously evolving nature also means we cannot hope to ever download it or train a model, whether large or small, on all of it.

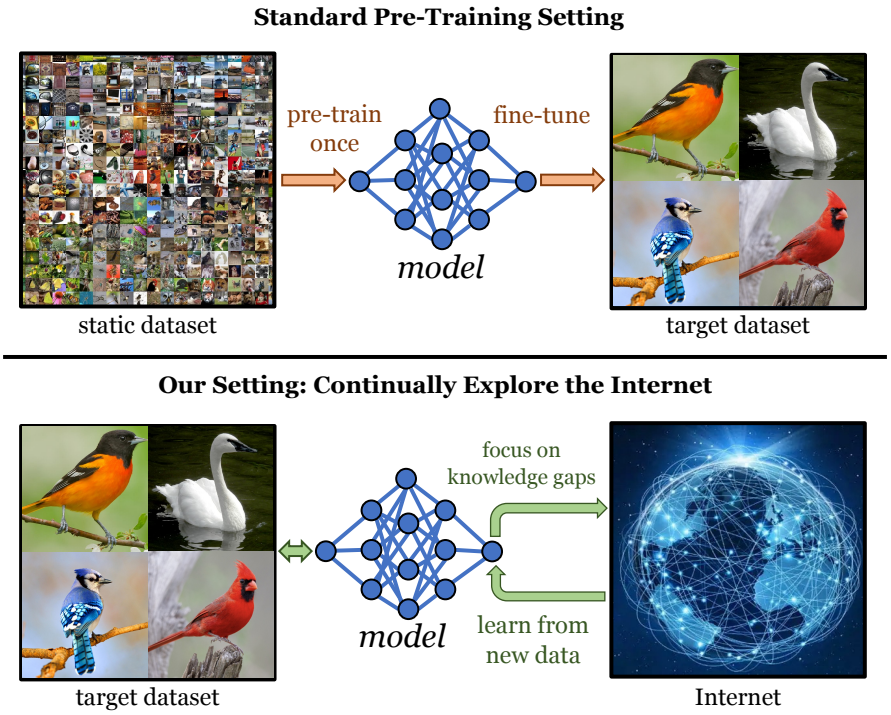


Figure 2.1: Given unlabeled data for a target task, our approach, Internet Explorer, searches the Internet to progressively find more and more relevant training data via self-supervised exploration.

We propose that the Internet can be treated as a special kind of dataset—one that exists out there, ready to be queried as needed to quickly train a customized model for a desired task. We draw an analogy to reinforcement learning, where even though the task is known, finding a policy that can generate the desired behavior is non-trivial due to the high complexity of the state space. Hence, most approaches rely on some form of exploration to figure out what actions the agent should take so that it quickly finds high-reward states. Inspired by this analogy, we formulate a disembodied, online agent we call *Internet Explorer*, that actively queries standard search engines to find relevant visual data that improve feature quality on a target dataset (see Figure 2.1). The agent's actions are text queries made to search engines, and the observations are the data obtained from the search.

The queries made by Internet Explorer improve over time. It cycles between searching for images on the Internet with text queries, self-supervised training on downloaded images, determining which images are relevant to the target dataset, and prioritizing what to search for next (see Figure 2.2). We also bootstrap Internet Explorer using existing pre-trained models such as MoCo-v3 [71] and obtain a significant boost on the target datasets.

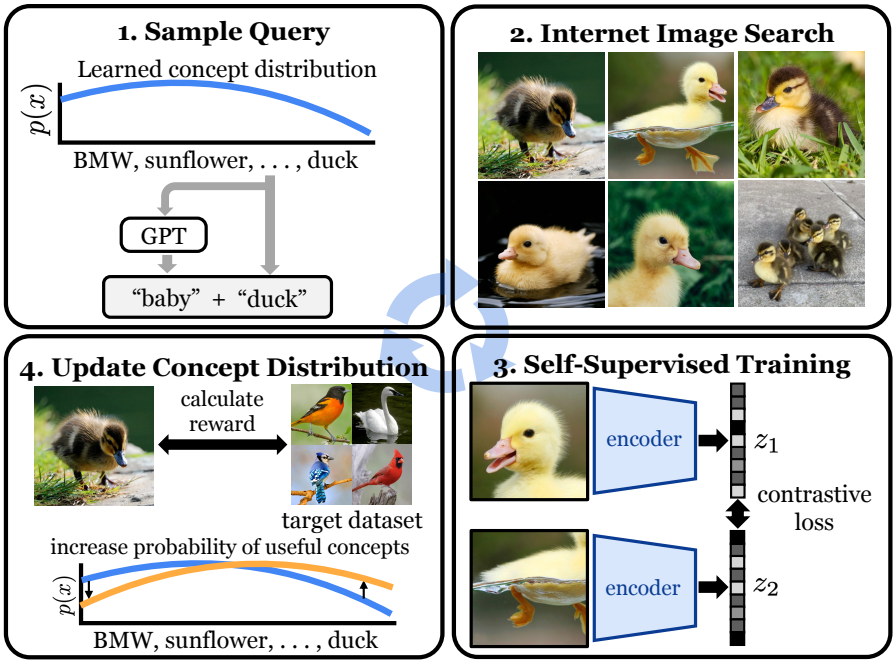


Figure 2.2: **Overview of Internet Explorer.** Our goal is to efficiently search the Internet for images that improve our performance on a target dataset. In each iteration, we first generate text queries by combining a concept sampled from a learned distribution with a GPT-generated descriptor (§2.1.2, §2.1.7). Next, we query search engines with the resulting phrase and download the top 100 image results (§2.1.1, 2.3.5). We add these images to the set of previously downloaded images and perform self-supervised training on the combined dataset (§2.1.3). Finally, we evaluate the relevance of the new images and update our concept distribution to increase the likelihood of similar queries if their images were similar to the target dataset (§2.1.4, §2.1.5).

Our setting is different from active learning [169], where the goal is to selectively obtain labels for data points from a fixed dataset. In contrast, Internet Explorer continually expands the size of its dataset and requires no labels for training, even from the target dataset. Some prior works have also discussed ways to leverage the Internet as an additional source of data. NELL [24] proposed a way to continually scrape web pages to learn new concepts and relationships, which are periodically curated by a human in the loop. NEIL [32] builds on NELL's dictionary to search visual data and develop visual relationships. Both are semi-supervised methods to gather general "common-sense" knowledge from the Internet. In contrast, we perform an actively improving *directed* search to perform well on target data, in a fully self-supervised manner.

2.1 INTERNET EXPLORER: AN ONLINE AGENT

We focus on the problem of efficiently improving representations for some target dataset by acquiring Internet data. We make as few assumptions as possible and assume that we have only unlabeled training data from the target dataset. Successful representation learning in this setting would lead to better performance on the target dataset distribution for standard tasks like classification and detection, and potentially others where the labels are not semantic (e.g., depth prediction or robotics). An overview of the Internet Explorer method is depicted in Figure 2.2 and described in Algorithm 1.

2.1.1 *Text-to-image Search*

We discover and download images from the full breadth of the Internet by querying text-to-image search engines, which return images based on their captions and surrounding text. Text-to-image search is fast, finds diverse images from across the Internet, and enables searches for vastly different queries simultaneously. Note that text-to-image search is noisy and makes use of weak supervision (the image-text pairing on webpages). Thus, we only perform self-supervised training on the downloaded images. We use a public codebase to query Google Images, which can download the top 100 images for each query [38, 189]. We also try other search engines in Section 2.3.5.

2.1.2 *Text Query Generation*

As text queries are our only input interface with the Internet, it is crucial that we can generate diverse queries that correspond to a variety of visual categories. Specificity is also important. Once a useful visual category is identified, generating fine-grained variants of the query is necessary to obtain data for all visual variations in the category. We construct queries by combining two components:

1. *Concepts* specify semantic categories such as people, places, or objects.
2. *Descriptors* are modifiers that generate variations in appearance.

We draw our concepts from the WordNet hierarchy [124], which consists of 146,347 noun lemmas. Not all of these lemmas are visual, but the vocabulary still covers an incredible range of topics. To generate a text query, we first sample a concept from a learned distribution over our vocabulary. This discrete distribution is defined by our estimates of how relevant each concept in the vocabulary is at the current time (see Section 2.1.4 for details on estimating rewards and Section 2.1.7 for the distribution). Given a sampled concept, we can generate a descriptor by prompting

a GPT-J language model [192] with examples of descriptor-concept pairs. Finally, as shown in Step 1 of Figure 2.2, we concatenate the concept and descriptor. If our concept is "duck" and the GPT-generated descriptor is "baby," our search engine query is "baby duck."

2.1.3 *Self-supervised Training*

We use self-supervised learning (SSL) to learn useful representations from the unlabeled images that we download from the Internet. Internet Explorer is compatible with any SSL algorithm that uses images or image-text pairs, including contrastive [29, 71], non-contrastive [12, 25, 64, 204], masking-based [9, 70], or multimodal [146] approaches. For speed and stability reasons, we use the MoCo-v3 algorithm [33], which trains encoders f_q and f_k on augmentations (x_1, x_2) of the same image to output vectors $q = f_q(x_1)$ and $k = f_k(x_2)$. f_q is trained to minimize the InfoNCE loss [134]:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)} \tag{2.1}$$

k^+ corresponds to f_k's output on the other augmentation of the image used to compute q, and the set of negative examples $\{k^-\}$ corresponds to f_k's output on other images in the batch. The temperature τ is set to 1 by default. f_k consists of a base encoder, a projection MLP, and a prediction head, whereas f_q is the exponential moving average of the base encoder and projection MLP from f_k. By training q and k^+ to be similar across image augmentations, MoCo-v3 encourages the network to learn high-level semantic features.

Before turning to the Internet, we initialize a ResNet-50 model [72] using a MoCo-v3 checkpoint trained offline for 100 epochs on ImageNet and then fine-tuned on the target dataset. Without using labels, we select the best starting checkpoint by early stopping on the SSL loss, which highly correlates with target accuracy [105]. In each iteration of our method, we use MoCo-v3 to fine-tune our encoder on a mixture of newly downloaded, previously downloaded, and target dataset images.

2.1.4 *Image Relevance Reward*

We want to rank newly downloaded images by how much they improve our features for the target dataset. This allows us to (a) prioritize taking gradient steps on useful images, and (b) understand what to search for in subsequent iterations. Unfortunately, it is challenging to directly measure the effect of an individual training example on performance. Numerous techniques have been proposed [53, 85, 94, 140],

but they all require extensive and repeated training on new images to estimate their impact.

Instead of trying to precisely measure what is learned from each image, we use its similarity to the target dataset as a proxy for being relevant to training. We rank the downloaded images by their similarity in representation space to the target dataset images; those most similar to the target dataset induce larger contrastive loss since each $\exp(q \cdot k^-)$ term in the denominator of Eq. 2.1 is larger when the negative examples $\{k^-\}$ are closer to q. These "hard negatives" [56, 68, 133, 153, 165, 196] yield larger and more informative gradients and should result in the biggest improvement in representation quality. Thus, overloading notation for k, we compute the reward for a particular image as its representation's average cosine similarity to its k closest neighbors in the target dataset. Given an image encoder $f_k : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^d$, an unlabeled target dataset $\mathcal{D} = \{x_i\}_{i=1}^N$, and a new image y to evaluate, the reward is calculated:

$$r(f_k, \mathcal{D}, y) = \max_{I \subset \{1,\ldots,N\};\, |I|=k} \frac{1}{k} \sum_{i \in I} S_{\cos}(f_k(x_i), f_k(y)) \tag{2.2}$$

where $S_{\cos}$ is the cosine similarity. A previous metric for identifying relevant data [87] used $k = 1$ nearest neighbors, but we found that this was too noisy and allowed high rewards for outlier target images to distract our search. We instead use $k = 15$ to improve the accuracy of our relevance estimation. In Section 2.3.6, we compare our reward to alternatives and explore their failure modes. This reward is used for two purposes: determining which of the downloaded images to train on and, subsequently, which concepts would be useful to search for next.

WHICH IMAGES TO TRAIN ON. Many newly downloaded images are not worth training on, since they come from unrelated queries or are noisy results from the search engine. Thus, at the end of each iteration, we rank the newly downloaded images by their reward and save the top 50% to a replay buffer that we maintain across iterations. In subsequent iterations, we continue training on this filtered data.

DETERMINING WHICH CONCEPTS ARE USEFUL. When we search for a concept and get back Q image results $\{I_i\}_{i=1}^Q$, we take the average of the top 10 image-level rewards $r_i = r(f_k, \mathcal{D}, I_i)$ and use that as a *concept-level score*. This gives us an accurate measure of the relevance of a particular query and reduces the impact of noisy search results.

Algorithm 1 `Internet Explorer`

Input: target dataset $\mathcal{D}$, SSL algorithm $\mathbb{A}$, search engine `SE`, encoder $f : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^d$, image reward function r, vocabulary $\mathcal{V} = \{c_i\}_{i=1}^{C}$, # concepts/itr M, # query results/search Q, GPT-based concept $\to$ descriptor function `GPTDesc`, concept distribution function `CalcProb`
Initialize replay buffer $\mathcal{B} \leftarrow \emptyset$
Initialize concept distribution $p = \text{Uniform}\{1, C\}$
for iteration $= 1, 2, \ldots$ **do**
  for $i = 1, \ldots, M$ **do**
    Sample concept $c_i \sim p(\mathcal{V})$ (§2.1.2)
    Sample descriptor $d_i \leftarrow$ `GPTDesc`(c_i)
    Image search $\{I_j^i\}_{j=1}^{Q} \leftarrow$ `SE`$(d_i + c_i, Q)$ (§2.1.1)
    Calc. reward $r_{c_i} \leftarrow \frac{1}{Q} \sum_{j=1}^{Q} r(f, \mathcal{D}, I_j^i)$ (§2.1.4)
  end for
  $\mathcal{B}_{\text{new}} = \{I_j^1\}_{j=1}^{Q} \cup \cdots \cup \{I_j^M\}_{j=1}^{Q}$
  SSL training: $\mathbb{A}(f, \mathcal{D} \cup \mathcal{B} \cup \mathcal{B}_{\text{new}})$ (§2.1.3)
  Add to buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup$ `Top50%`$(\mathcal{B}_{\text{new}}, r)$
  Predict all concept rewards $\mathbf{r}_{\text{concept}}$ from $\{r_{c_i}\}$ (§2.1.5)
  Update concept dist $p \leftarrow$ `CalcProb`$(\mathbf{r}_{\text{concept}})$ (§2.1.7)
end for

2.1.5 *Estimating Reward for Unseen Concepts*

Since our vocabulary contains hundreds of thousands of concepts, it is inefficient to search to test whether a query yields relevant images. Luckily, we can estimate the quality of a query by using the observed rewards of the queries used so far. Humans can do this effortlessly due to our understanding of what each concept means. To us, it is obvious that if querying "golden retriever" yielded useful images for this dataset, then "labrador retriever" probably should as well. To give our method the same understanding of concept meaning, we embed our 146,347 WordNet concepts into a 384-dimensional space using a pre-trained sentence similarity model [151].

We use Gaussian process regression (GPR) [195] over the text embeddings $\{\mathbf{e}_i\}$ to predict the concept-level reward $r(\mathbf{e}_i)$ for untried concepts. GPR models the function outputs for any set of inputs $\{r(\mathbf{e}_i)\}$ as jointly Gaussian random variables, and it estimates a Gaussian posterior with mean $\mu(\mathbf{e}')$ and variance $\sigma(\mathbf{e}')^2$. We encourage exploration by setting the score of unobserved concepts to $\mu(\mathbf{e}_i) + \sigma(\mathbf{e}_i)$.

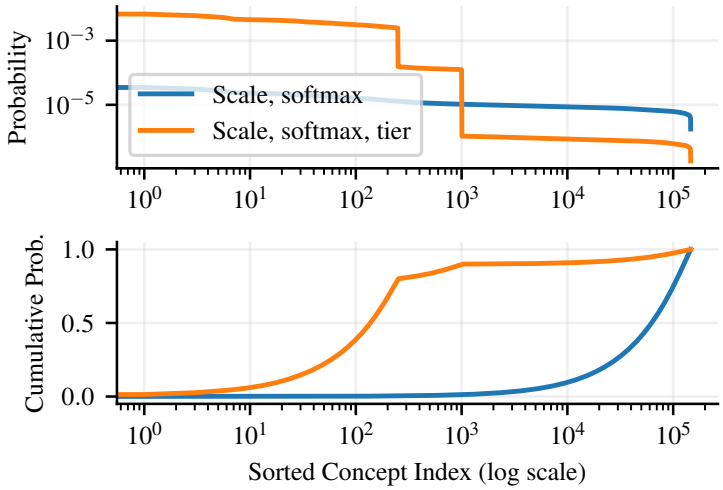


Figure 2.3: **Learned concept sampling distribution.** Given estimated scores for each of the 146,347 concepts, we need to choose how often to sample each one in order to balance exploration and exploitation. **Top:** we scale our scores to a desired temperature, then take the softmax to obtain a distribution over concepts. Finally, we create tiers so that the top 250 concepts have 80% of the probability mass, and the next 750 have 10%. This ensures that we sample enough from the top 1,000 concepts while still exploring other concepts with lower scores. **Bottom:** the top 1,000 concepts are only sampled a tiny fraction of the time without tiering.

2.1.6 *Provable speedup in relevant query identification*

Only a small subset of our vocabulary of n concepts is relevant to the target dataset. We assume that the relevant concepts are partitioned into c disjoint clusters of size s, with $cs \ll n$. We want to discover every relevant concept by sampling concepts uniformly at random (with replacement) to test. We assume that sampling a concept conclusively tells us whether it is relevant. Furthermore, we assume that we could optionally use an algorithm (e.g., Gaussian process regression) that, if we have sampled a relevant concept, tells us that all concepts in its cluster are also relevant. Then, Lemma 2.1.1 shows that the Gaussian process drastically reduces the time required to identify all relevant concepts.

Lemma 2.1.1. *Let T_{base} be the expected time to identify every relevant concept without the GPR, and T_{GPR} be the expected time when exploiting the additional knowledge from the GPR. Then, $T_{base} = nH_{c \cdot s}$, $T_{GPR} = \frac{nH_c}{s}$, and the speedup from GPR is $\frac{T_{base}}{T_{GPR}} \approx s \log s$.*

For our vocabulary and target datasets, we estimate $s \approx 100$, for a total speedup of $200 - 300\times$. This shows that a predictive model like GPR is crucial for quickly identifying all useful concepts.

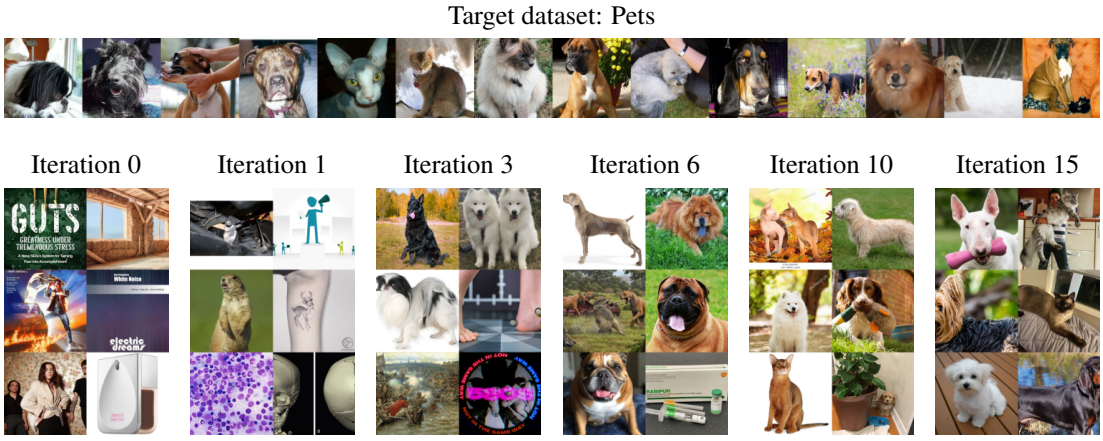


Figure 2.4: **Progression of downloaded images across training. Top:** samples of Oxford-IIIT Pets images. **Bottom:** samples of images queried by Internet Explorer across iterations. As it learns, it makes queries that are progressively more relevant to the target dataset.

2.1.7 *Query sampling distribution*

Once we have estimates for the quality of each concept, how do we determine what to search for next? We face the age-old dilemma of exploration versus exploitation: we need to sample the top concepts frequently enough to get relevant training data for SSL, while at the same time, we need sufficient exploration of promising untried concepts.

We use a sampling-based approach based on Boltzmann exploration [180]. Boltzmann exploration samples based on a scaled softmax distribution $p(c_i) \propto \exp(r(c_i)/\tau)$, where τ is the temperature scaling. However, with a large vocabulary (action space) of 146,347 concepts, it becomes difficult to tune τ so that we sample the top concepts frequently enough without being too skewed. Thus, we define a "tiering function" to adjust the probability mass in specified intervals of our distribution. Given a sorted discrete probability distribution p, interval boundaries $T_0 = 0 < T_1 < \cdots < T_n$, and interval masses $\Delta_0, \ldots, \Delta_{n-1}$ such that $\sum_i \Delta_i = 1$, tiering computes a new distribution:

$$p_i^{\text{tier}} = \Delta_j \frac{p_i}{\sum_{k=T_j}^{T_{j+1}} p_k} \quad \text{for } j \text{ s.t. } T_j \leqslant i < T_{j+1} \tag{2.3}$$

p^{tier} is a new distribution such that $\sum_{k=T_j}^{T_{j+1}} p^{\text{tier}} = \Delta_j$. We use $T_0 = 0$, $T_1 = 250$, $T_2 = 1{,}000$, $T_3 = 146{,}347$, $\Delta_0 = 0.8$, $\Delta_1 = 0.1$, and $\Delta_2 = 0.1$. Simply put: we give the highest-ranked 250 concepts 80% of the probability mass, the next 750 concepts 10%,

and all remaining concepts 10%. Figure 2.3 shows that tiering the scaled softmax distribution samples frequently enough from the top concepts while a vanilla scaled softmax distribution does not.

2.2 EXPERIMENTAL SETTING

2.2.1 *Self-supervised Exploration*

We assume that we have an unlabeled target dataset of images for which we would like to learn useful visual features. We compare three methods:

1. Random: sample concepts uniformly from the vocab.
2. Ours: sample concepts from our learned distribution.
3. Ours++: additionally use GPT-generated descriptors.

2.2.2 *Label Set-guided Exploration*

We may sometimes know the set of labels for our task (e.g., "golden retriever," etc.) even if we do not have image-label pairs. Knowing the label set greatly accelerates learning on the Internet, because it acts as a strong prior on what could be useful. Using our text similarity model, we reduce the size of the vocabulary by selecting the top 10% (14,635 concepts) with the largest average top-k similarity to the label set in text embedding space. We set k to a third of the size of the label set to reduce the impact of outliers. Reducing the size of the vocabulary strengthens our baselines by ensuring that they only search for potentially useful concepts. We compare 4 methods:

1. Labels: only search for labels.
2. Labels + relevant: search for labels half of the time, and random concepts from the pruned vocabulary the other half of the time.
3. Ours: sample labels half of the time and sample from our learned concept distribution the other half.
4. Ours++: additionally use GPT-generated descriptors.

We call this setting "label set-guided," since we have additional supervision in the form of the label set.

2.2.3 *Datasets and Metrics*

We evaluate Internet Explorer on 4 popular small-scale fine-grained classification datasets: Birdsnap [15], Flowers-102 [131], Food101 [18], and Oxford-IIIT Pets [139]. These small datasets consist of 2,040 to 75,750 training examples, making them ideal

Model	Birdsnap	Flowers	Food	Pets	VOC2007	IN100	FMoW*	Images	GPU hrs.
Fixed dataset, lang. supervision									
CLIP ResNet-50 (**oracle**)	57.1	96.0	**86.4**	88.4	**86.7**	**89.3**	44.9	400×10^6	4,000
Fixed dataset, self-supervised									
MoCo-v3 (ImageNet pre-train)	26.8	83.2	70.5	79.6	–	–	40.8	1.2×10^6	72
MoCo-v3 (ImageNet + target)	39.9	94.6	78.3	85.3	58.0†	84.7†	52.5	1.2×10^6	72 + 12
No label set information									
Random exploration	39.6 (−0.3)	95.3 (+0.7)	77.0 (−1.3)	85.6 (+0.3)	70.2 (+12.2)	85.7 (+1.0)	54.3 (+1.8)	2.2×10^6	84 + 40
Ours	43.4 (+3.5)	97.1 (+2.5)	80.5 (+2.2)	86.8 (+1.5)	68.5 (+10.5)	86.2 (+1.5)	– –	2.2×10^6	84 + 40
Ours++	54.4 (+14.5)	98.4 (+3.8)	82.2 (+3.9)	89.6 (+4.3)	80.1 (+22.1)	86.4 (+1.7)	54.1 (+1.6)	2.2×10^6	84 + 40
Use label set information									
Search labels only	47.1 (+7.2)	96.3 (+1.7)	80.9 (+2.6)	85.7 (+0.4)	61.8 (+3.8)	85.7 (+1.0)	53.5 (+1.0)	2.2×10^6	84 + 40
Labels + relevant terms	49.9 (+10.0)	98.0 (+3.4)	81.2 (+2.9)	87.0 (+1.7)	67.5 (+9.5)	86.3 (+1.6)	54.1 (+1.6)	2.2×10^6	84 + 40
Ours	52.0 (+12.1)	97.6 (+3.0)	81.2 (+2.9)	87.3 (+2.0)	70.3 (+14.3)	86.4 (+1.7)	– –	2.2×10^6	84 + 40
Ours++	**62.8** (+22.9)	**99.1** (+4.5)	84.6 (+6.3)	**90.8** (+5.5)	79.6 (+21.6)	87.1 (+2.4)	**54.5** (+2.0)	2.2×10^6	84 + 40

Table 2.1: **Linear probing accuracy**. Our method significantly improves the starting checkpoint performance in just 40 additional hours of training. We show the performance change from the starting MoCo-v3 (ImageNet + target) initialization in green/red. CLIP numbers correspond to linear probe (which is higher than its zero-shot accuracy). Internet Explorer reaches or often surpasses CLIP (oracle with 2x params) performance on each dataset while using 2.5% as much compute and 0.5% as much data. †For VOC2007 and IN100, we do not do ImageNet pre-training because ImageNet is too similar and obscures the effect. *For FMoW-WILDS, we use a hand-crafted list of domain-specific descriptors common to all models

for testing whether Internet Explorer can efficiently find relevant useful data. We also evaluate on PASCAL VOC 2007 (Cls) [51], a coarse-grained multi-label classification task, and ImageNet-100 [185]. Finally, we try FMoW [36], a satellite domain classification task. We compare the representation quality of our model *w*.r.t. its target dataset using two metrics: k-nearest neighbors (k-NN) accuracy and linear probe accuracy.

2.3 RESULTS AND ANALYSIS

2.3.1 *Self-supervised Results*

Internet Explorer improves the k-NN accuracy more efficiently than sampling queries uniformly at random from the concept vocabulary. In fact, random sampling occasionally decreases accuracy, likely due to the fact that Internet images can generally be unsuitable for pre-training due to issues such as watermarks, images containing text, and overly photogenic images [30, 123]. Table 2.1 shows that our method significantly improves on the starting MoCo-v3 (ImageNet + target) checkpoint and can outperform a CLIP [146] model of the same size while using much less compute and

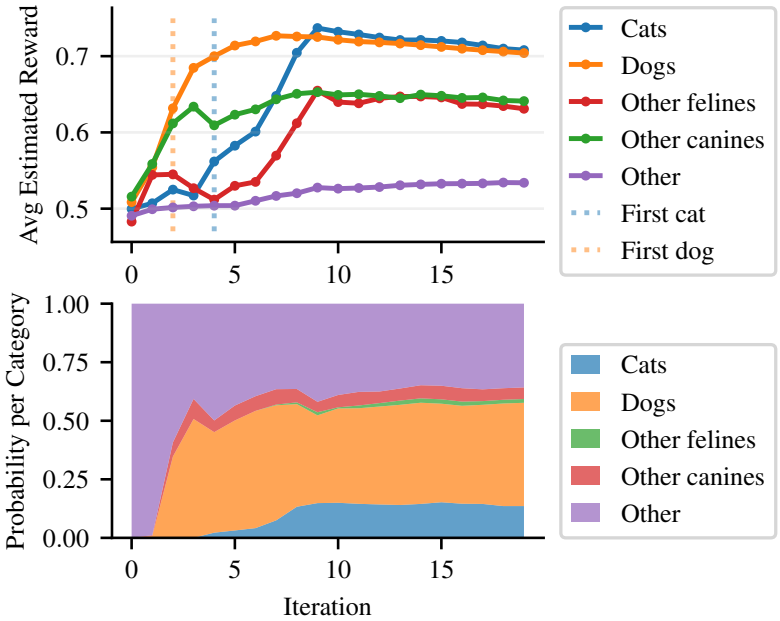


Figure 2.5: **Self-supervised concept discovery on Pets dataset.** When targeting the Pets dataset, self-supervised Internet Explorer quickly estimates high reward for concepts from the cat category (82 concepts) and dog category (246 concepts). It is also able to identify felines that are not cats (e.g., tiger) and canines that are not dogs (e.g., wolf), although it gives them lower reward on average. Finding these categories is especially challenging since they comprise only 460/146,347 = 0.3% of the vocabulary.

data. This is impressive as CLIP can be considered an oracle since its training set contains up to 20k Bing image search results for each WordNet lemma (in addition to other queries). Using GPT-generated descriptors in "Ours++" also significantly improves performance by enabling Internet Explorer to generate diverse views of the most useful concepts.

2.3.2 *Self-supervised Exploration Behavior*

Figure 2.5 shows the progression of Internet Explorer (Ours++) behavior on the Pets dataset in the self-supervised setting. Since Pets consists of cat and dog breeds, to analyze the results, we use the WordNet hierarchy to divide concepts in our vocabulary into 5 meaningful categories: cats, dogs, non-cat felines (e.g., lion), non-dog canines (e.g., wolf), and other. This categorization is only done for this post hoc analysis and is not provided during training. Figure 2.5 (top) shows that Internet Explorer rapidly identifies the roughly 0.3% of concepts that are useful for Pets. During the

first two iterations, the average estimated reward for each category is roughly the same. However, after the first dog concept is searched in iteration #2, the estimated reward and probability mass for dogs and other canines rapidly increases. The same happens for cats after the first cat is searched in iteration #4. Interestingly, while "other felines" and "other canines" have higher average reward than the "other" category, they still have much lower reward than cats and dogs. This indicates that our model understands that other felines and canines (mostly large, wild predators) are only moderately relevant for house pet cats and dogs.

Figure 2.4 shows how Internet Explorer downloads progressively more useful images over time. It shows 8 random images that were downloaded in iteration #0, #1, #3, #6, #10, and #15 in the self-supervised setting. Iteration #0 contains mostly useless data, like graphics or screenshots, but Pets-relevant images already make up most of the downloads by iteration #3.

2.3.3 *Label Set-guided Results*

Internet Explorer significantly outperforms the stronger baselines in the label set-guided setting where we additionally have knowledge of the label set. Searching for the label set continuously provides useful data and helps us rapidly identify other useful concepts. Together with the diversity promoted by GPT descriptors, Ours++ outperforms CLIP in 4/7 datasets and approaches its performance in the other 3, using just 2.5% of the time and 0.5% the data.

2.3.4 *Domain dataset results*

To test if Internet Explorer is effective when the target dataset contains very specific domain knowledge, we apply it to FMoW-WILDS [36]—a popular satellite imaging domain dataset—by hand-designing a dozen search prompts that help induce satellite image results. Even though the WordNet vocabulary is not particularly suited for this dataset, Internet Explorer still improves the LP accuracy by 2 percentage points (see Table 2.1). Notably, all of our methods dramatically outperform CLIP here, likely because the distribution of satellite data is very different than the data used to train CLIP. This demonstrates the wide flexibility of our method to be applied to arbitrary domains.

2.3.5 *Learning from other sources of data*

We primarily obtain images by querying Google Images, but Internet Explorer is compatible with any text-to-image search engine. To measure the effect of the choice

Model	Flowers			Food			Pets		
	Google	Flickr	LAION	Google	Flickr	LAION	Google	Flickr	LAION
Fixed dataset									
MoCo-v3 (IN)	83.2	83.2	83.2	70.5	70.5	70.5	79.6	79.6	79.6
MoCo-v3 (IN + target)	94.6	94.6	94.6	78.3	78.3	78.3	85.3	85.3	85.3
Undirected search									
Random exploration	95.3	95.2	94.8	77.0	80.0	80.2	85.6	84.4	85.1
Internet Explorer									
Ours++ (no label set)	98.4	98.1	94.6	81.2	80.3	80.9	87.3	88.4	85.9
Ours++ (with label set)	**99.1**	**99.0**	**95.8**	**84.6**	**81.9**	**81.0**	**90.8**	**89.1**	**86.7**

Table 2.2: **Linear probe accuracy with other search engines**. Internet Explorer improves its performance using any search engine, including Flickr and our custom text-based LAION search engine.

of search engine, we also test Internet Explorer with the Flickr photo search API and a custom search engine we built on top of a subset of LAION-5B [166]. LAION-5B consists of noisy web-scraped (text, image) pairs, and our custom LAION search engine searches using approximate nearest neighbors in *text embedding space*. Thus, it tests whether Internet Explorer can still improve even when the search engine has little inductive bias. Table 2.2 shows that Internet Explorer consistently improves over time, regardless of the search engine we use. Google consistently does best, followed by Flickr, then LAION (which has the smallest pool of images to draw from). Using Internet Explorer to search LAION-5B consistently performs *better* than random exploration—indicating that Internet Explorer is effective even for selecting data from a static dataset.

2.3.6 *Effect of image reward type*

Reward Type	Food
MoCo loss	81.2
1-NN sim	83.2
15-NN sim (ours)	**84.6**

Table 2.3: **Ablation on type of image reward.** MoCo loss does not identify relevant concepts, and $k = 1$ similarity is too noisy to identify useful concepts.

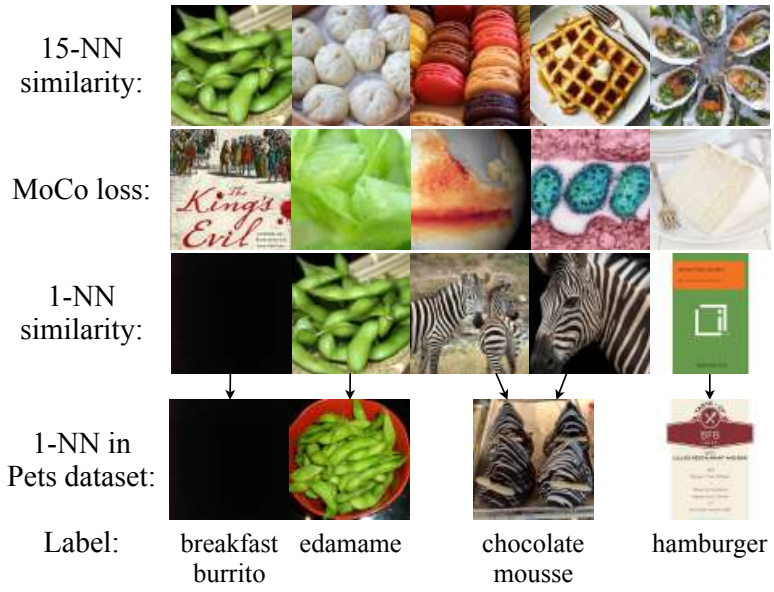


Figure 2.6: **Most preferable images under different rewards.** We show the top 5 downloaded images ranked by 3 possible image rewards for adversarial Food101 examples. MoCo loss encourages noisy out-of-distribution images; 15-NN (ours) prefers a wide variety of food images, whereas outliers in the Food dataset throw off 1-NN, causing it to reward black images, text, and zebras.

We run an ablation on the type of image relevance reward. Instead of calculating the image reward based on the average similarity to the $k = 15$ nearest neighbors in representation space (as in Section 2.1.3), we also try using $k = 1$ or the MoCo contrastive loss as the reward. Table 2.3 compares these three metrics in the label set-guided setting and shows that $k = 15$ does best. We explain this result by qualitatively comparing the behavior of various metrics on Food101 in Figure 2.6. The MoCo loss does not identify relevant concepts, instead preferring images that are difficult to align across augmentations. Representation similarity with $k = 1$ also fails, as it prefers images of `zebras` and `books` because they are highly similar to a few outlier images in Food101. Our proposed reward with $k = 15$ eliminates the influence of outliers and avoids this problem.

2.3.7 *Comparison to image-to-image search*

An alternate approach to finding relevant Internet data is to use image-to-image search: for each image in the target dataset, directly retrieve images that are visually similar.

SCIENTIFIC AND PRACTICAL ISSUES Image-to-image search uses strong visual representations from pretrained models in order to identify similar images. This defeats the primary purpose of Internet Explorer: learning useful representations when none exist beforehand (e.g., a new iPhone is released that is out-of-distribution for existing vision models). Text-based search avoids this issue by using additional supervision (e.g., caption and surrounding text) that makes it easier to index new images. Image-to-image search also relies on paid APIs that can cost thousands of dollars.

COMPARISON TO TEXT-BASED SEARCH Regardless of the concerns above, we do a controlled comparison between Internet Explorer and image-based search over LAION-5B. For each image in a target training set, we compute its CLIP ViT-L/14 representation and find its N nearest neighbors in LAION-5B. We choose N so that we download a total of 1 million new images, which matches how many images Internet Explorer downloads. We then train a MoCo-v3 model on a 1:1 mix of the target dataset and the downloaded images with the exact same hyperparameters (e.g., learning rate, number of steps, etc) as Internet Explorer. Interestingly, Table 2.4 shows that the image-to-image approach consistently learns worse features than Internet Explorer, despite taking advantage of strong, pretrained vision features from CLIP. We hypothesize that image-to-image search finds images that are too similar to the target images, resulting in less additional information that was not already present in the target dataset. In contrast, using text (concepts and descriptors) as an intermediate bottleneck encourages Internet Explorer to download novel images that generalize along useful axes.

2.4 RELATED WORK

Many papers use self-supervised or weakly-supervised learning on large-scale static datasets collected from the Internet, such as YFCC-100M [183], Instagram-1B [118], or LAION-5B [166]. However, these are usually impractically expensive since they train on all of the data, not just the subset relevant to a target dataset. Concurrent work [135] attempts to address this by adding a "one-time" automatic data curation step that keeps only the most relevant images from a static web crawl dataset. This approach works well but is limited as the selection process does not use the most up-to-date learned features or adjust its searches on-the-fly to focus on especially useful data.

Other approaches obtain additional training data by searching for predetermined queries. Fergus et al. [54] create a supervised training dataset from the Google image search results for a list of known classes. Kamath et al. [90] improve a visual question-

	Flowers	Pets	VOC2007
Image-to-image	96.6	81.6	67.8
Internet Explorer (ours)	**98.8**	**87.0**	**76.1**

Table 2.4: k**-NN accuracy across search methods**. Image-to-image search uses CLIP ViT-L/14 vision features to acquire the nearest neighbors of each target dataset image. Despite using strong pretrained features and the same source data (LAION-5B), number of downloaded images, and other hyperparameters as Internet Explorer, the image-to-image approach learns worse features.

answering model using a set of predetermined Bing queries. However, these approaches query the internet just once, which is susceptible to noise in the search results, and the total amount of data is limited to the relevant search terms known *a priori*. Internet Explorer's self-supervised approach bypasses these problems. It can learn useful features from noisy yet relevant data, and it only needs an initial image collection to identify relevant search queries. This enables it to *continually* explore the Internet via a potentially unbounded number of searches.

Finally, some approaches continuously interact with the Internet to find useful data. NELL [24, 126] extracts text from web pages to form beliefs, and NEIL [32] uses images downloaded from Google Image Search to learn visual concepts. However, both methods are undirected (i. e., they do not modify their exploration behavior to prioritize specific data), which means that learning is slow and will not necessarily improve performance on a desired task. In contrast, Internet Explorer continually uses *targeted* exploration on the Internet to find data for self-supervised training.

2.5 DISCUSSION

We show that interactively exploring the Internet is an efficient source of highly useful training data—if one knows how to search for it. In just 30–40 hours of training on a single GPU, Internet Explorer significantly outperforms or closely matches the performance of compute-heavy *oracle* models like CLIP trained on huge, static datasets, as well as strong baselines that search the Internet in an undirected manner.

Part II

HARDER LEARNING OBJECTIVES

3

LEARNING GENERATIVE MODELS FOR CLASSIFICATION

Classification is a fundamental problem in machine learning, and AlexNet's strong results on the ImageNet classification challenge arguably jumpstarted the popularity of deep learning as a field. Today, deep learning classification algorithms have largely converged to using a neural network to output logits, which attempt to minimize some variant of a cross-entropy loss.

This is a *discriminative* approach to classification, and it stands in stark contrast to how humans do it. Significant evidence from cognitive science [202] indicates that humans have a *generative* understanding of a world – we can classify cats vs. dogs because we know what they look like, from head to tail. In contrast, discriminative models cannot "imagine" what a dog would look like. Internally, they may be looking for a small set of discriminative features, perhaps a particular eye shape or fur texture. But they do not learn what each class looks like as a whole. Perhaps this difference in objective, generative vs discriminative, is responsible for current shortcomings of deep learning, such as sensitivity to distribution shift?

Generative classifiers have had a long history in machine learning, since early methods like linear discriminant analysis and Naive Bayes [130], but had been limited due to the lack of flexible generative models. Today's generative modeling algorithms are far more powerful. In this section, we design efficient and accurate classifiers using these new generative models and explore their learning advantages.

3.1 CLASSIFICATION VIA DIFFUSION MODELS

Diffusion models are a recent class of likelihood-based generative models that model the data distribution via an iterative noising and denoising procedure [76, 173]. These models are trained via a variational objective, which maximizes an evidence lower bound (ELBO) [16] of the log-likelihood. They have recently achieved state-of-the-art performance [45] on several text-based content creation and editing tasks [78, 143, 157, 161, 214]. However, diffusion models have been underexplored for discriminative tasks like classification. In this paper, we use diffusion models, the current

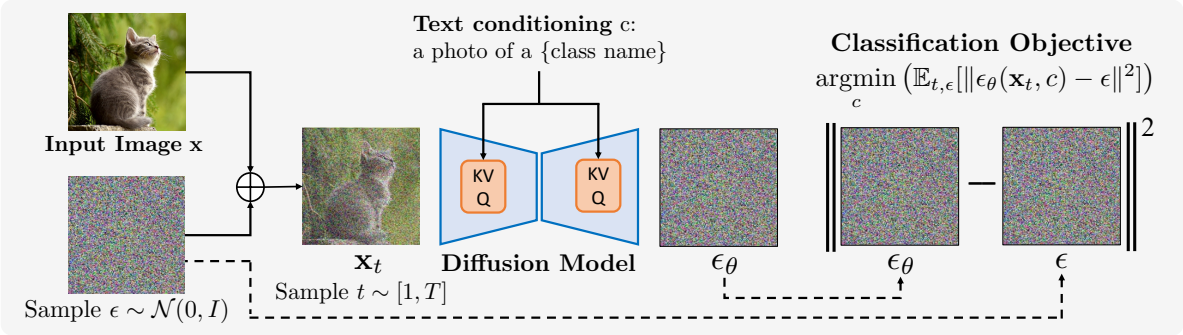


Figure 3.1: **Overview of our Diffusion Classifier approach:** Given an input image $\mathbf{x}$ and a set of possible conditioning inputs (e.g., text for Stable Diffusion or class index for DiT, an ImageNet class-conditional model), we use a diffusion model to choose the one that best fits this image. Diffusion Classifier is theoretically motivated through the variational view of diffusion models and uses the ELBO to approximate $\log p_\theta(\mathbf{x} \mid \mathbf{c})$. Diffusion Classifier chooses the conditioning $\mathbf{c}$ that best predicts the noise added to the input image. *Diffusion Classifier can be used to extract a zero-shot classifier from Stable Diffusion and a standard classifier from DiT without any additional training.*

state-of-the-art generative model family, to revisit the classic generative vs. discriminative classifier debate.

3.1.1 *Method: Diffusion Classifier*

We describe our approach for calculating class conditional density estimates in a practical and efficient manner using diffusion models. We first provide an overview of diffusion models (Sec. 3.1.2), discuss the motivation and derivation of our Diffusion Classifier method (Sec. 3.1.2), and finally propose techniques to improve its accuracy (Sec. 3.1.3).

3.1.2 *Diffusion Model Preliminaries*

Diffusion probabilistic models ("diffusion models" for short) [76, 173] are generative models with a specific Markov chain structure. Starting at a clean sample $\mathbf{x}_0$, the fixed forward process $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ adds Gaussian noise, whereas the learned reverse process $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$ tries to denoise its input, optionally conditioning on a variable $\mathbf{c}$. In our setting, $\mathbf{x}$ is an image and $\mathbf{c}$ represents a low-dimensional text embedding

(for text-to-image synthesis) or class index (for class-conditional generation). Diffusion models define the conditional probability of $\mathbf{x}_0$ as:

$$p_\theta(\mathbf{x}_0 \mid \mathbf{c}) = \int_{\mathbf{x}_{1:T}} p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) \, d\mathbf{x}_{1:T} \tag{3.1}$$

where $p(\mathbf{x}_T)$ is typically fixed to $\mathcal{N}(0, I)$. Directly maximizing $p_\theta(\mathbf{x}_0)$ is intractable due to the integral, so diffusion models are instead trained to minimize the variational lower bound (ELBO) of the log-likelihood:

$$\log p_\theta(\mathbf{x}_0 \mid \mathbf{c}) \geqslant \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T}, \mathbf{c})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right] \tag{3.2}$$

Diffusion models parameterize $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$ as a Gaussian and train a neural network to map a noisy input $\mathbf{x}_t$ to a value used to compute the mean of $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$. Using the fact that each noised sample $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1 - \bar{\alpha}_t}\epsilon$ can be written as a weighted combination of a clean input $\mathbf{x}$ and Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, diffusion models typically learn a network $\epsilon_\theta(\mathbf{x}_t, \mathbf{c})$ that estimates the added noise. Using this parameterization, the ELBO can be written as:

$$-\mathbb{E}_\epsilon \left[\sum_{t=2}^{T} w_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2 - \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1, \mathbf{c}) \right] + C \tag{3.3}$$

where C is a constant term that does not depend on $\mathbf{c}$. Since $T = 1000$ is large and $\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1, \mathbf{c})$ is typically small, we choose to drop this term. Finally, [76] find that removing w_t improves sample quality metrics, and many follow-up works also choose to do so. We found that deviating from the uniform weighting used at training time hurts accuracy, so we set $w_t = 1$. Thus, this gives us our final approximation that we treat as the ELBO:

$$-\mathbb{E}_{t,\epsilon} \left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2 \right] + C \tag{3.4}$$

CLASSIFICATION WITH DIFFUSION MODELS In general, classification using a conditional generative model can be done by using Bayes' theorem on the model predictions $p_\theta(\mathbf{x} \mid \mathbf{c}_i)$ and the prior $p(\mathbf{c})$ over labels $\{\mathbf{c}_i\}$:

$$p_\theta(\mathbf{c}_i \mid \mathbf{x}) = \frac{p(\mathbf{c}_i) \; p_\theta(\mathbf{x} \mid \mathbf{c}_i)}{\sum_j p(\mathbf{c}_j) \; p_\theta(\mathbf{x} \mid \mathbf{c}_j)} \tag{3.5}$$

A uniform prior over $\{\mathbf{c}_i\}$ (i.e., $p(\mathbf{c}_i) = \frac{1}{N}$) is natural and leads to all of the $p(\mathbf{c})$ terms cancelling. For diffusion models, computing $p_\theta(\mathbf{x} \mid \mathbf{c})$ is intractable, so we use

Algorithm 2 `Diffusion Classifier`

Input: test image $\mathbf{x}$, conditioning inputs $\{\mathbf{c}_i\}_{i=1}^n$ (e. g., text embeddings), # of trials T per input
Initialize $\texttt{Errors}[\mathbf{c}_i] = \text{list}()$ for each $\mathbf{c}_i$
for trial $j = 1, \ldots, T$ **do**
  Sample $t \sim [1, 1000]$; $\epsilon \sim \mathcal{N}(0, I)$
  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1 - \bar{\alpha}_t}\epsilon$
  for conditioning $\mathbf{c}_k \in \{\mathbf{c}_i\}_{i=1}^n$ **do**
    $\texttt{Errors}[\mathbf{c}_k].\texttt{append}(\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_k)\|^2)$
  end for
end for
return $\arg\min_{\mathbf{c}_i \in \mathcal{C}} \texttt{mean}(\texttt{Errors}[\mathbf{c}_i])$

the ELBO in place of $\log p_\theta(\mathbf{x} \mid \mathbf{c})$ and use Eq. 3.4 and Eq. 3.5 to obtain a posterior distribution over $\{\mathbf{c}_i\}_{i=1}^N$:

$$p_\theta(\mathbf{c}_i \mid \mathbf{x}) = \frac{\exp\{-\mathbb{E}_{t,\epsilon}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_i)\|^2]\}}{\sum_j \exp\{-\mathbb{E}_{t,\epsilon}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_j)\|^2]\}} \tag{3.6}$$

We compute an unbiased Monte Carlo estimate of each expectation by sampling N (t_i, ϵ_i) pairs, with $t_i \sim [1, 1000]$ and $\epsilon \sim \mathcal{N}(0, I)$, and computing:

$$\frac{1}{N}\sum_{i=1}^{N} \left\| \epsilon_i - \epsilon_\theta(\sqrt{\bar{\alpha}_{t_i}}\mathbf{x} + \sqrt{1 - \bar{\alpha}_{t_i}}\epsilon_i, \mathbf{c}_j) \right\|^2 \tag{3.7}$$

By plugging Eq. 3.7 into Eq. 3.6, we can extract a classifier from any conditional diffusion model. We call this method **Diffusion Classifier**. *Diffusion Classifier is a powerful, hyperparameter-free approach to extracting classifiers from pretrained diffusion models without any additional training*. Diffusion Classifier can be used to extract a zero-shot classifier from a text-to-image model like Stable Diffusion [154], to extract a standard classifier from a class-conditional diffusion model like DiT [141], and so on. We outline our method in Algorithm 2 and show an overview in Figure 3.1.

3.1.3 *Variance Reduction via Difference Testing*

At first glance, it seems that accurately estimating $\mathbb{E}_{t,\epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2\right]$ for each class $\mathbf{c}$ requires prohibitively many samples. Indeed, a Monte Carlo estimate even using thousands of samples is not precise enough to distinguish classes reliably. However, a key observation is that classification only requires the *relative* differences between

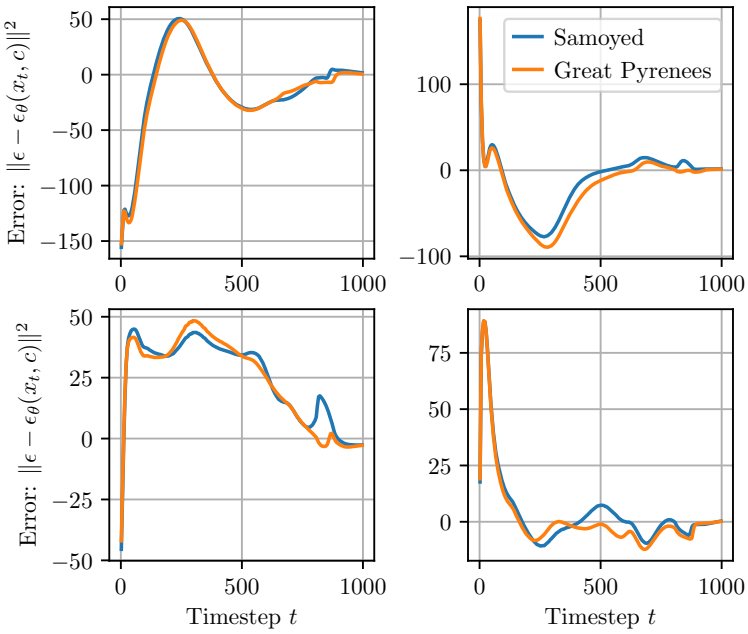


Figure 3.2: We show the ϵ-prediction error for an image of a Great Pyrenees dog and two prompts ("Samoyed" and "Great Pyrenees"). Each subplot corresponds to a single ϵ_i, with the error evaluated at every $t \in \{1, 2, ..., 1000\}$. Errors are normalized to be zero-mean at each timestep across the 4 plots, and lower is better. Variance in ϵ-prediction error is high across different ϵ, but the variance in the error difference between prompts is much smaller.

the prediction errors, not their *absolute* magnitudes. We can rewrite the approximate $p_\theta(\mathbf{c}_i \mid \mathbf{x})$ from Eq. 3.6 as:

$$\frac{1}{\sum_j \exp\left\{\mathbb{E}_{t,\epsilon}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_i)\|^2 - \|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_j)\|^2]\right\}} \tag{3.8}$$

Eq. 3.8 makes apparent that we only need to estimate the *difference* in prediction errors across each conditioning value. Practically, instead of using different random samples of (t_i, ϵ_i) to estimate the ELBO for each conditioning input $\mathbf{c}$, we simply sample a fixed set $S = \{(t_i, \epsilon_i)\}_{i=1}^N$ and use the same samples to estimate the ϵ-prediction error for every $\mathbf{c}$. This is reminiscent of paired difference tests in statistics, which increase their statistical power by matching conditions across groups and computing differences.

In Figure 3.2, we use 4 fixed ϵ_i's and evaluate $\|\epsilon_i - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\epsilon_i, \mathbf{c})\|^2$ for every $t \in 1, \ldots, 1000$, two prompts ("Samoyed dog" and "Great Pyrenees dog"), and a fixed input image of a Great Pyrenees. Even for a fixed prompt, the ϵ-prediction error varies wildly across the specific ϵ_i used. However, the error difference between

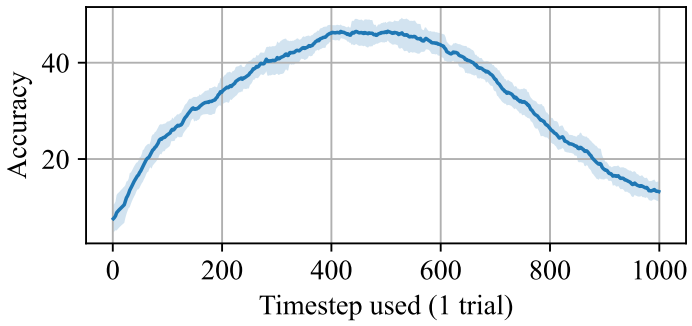


Figure 3.3: **Pets accuracy, evaluating only a single timestep per class**. Small t corresponds to less noise added, and large t corresponds to significant noise. Accuracy is highest when an intermediate amount of noise is added ($t = 500$).

each prompt is much more consistent for each ϵ_i. *Thus, by using the same* (t_i, ϵ_i) *for each conditioning input, our estimate of* $p_\theta(\mathbf{c}_i \mid \mathbf{x})$ *is much more accurate.*

3.2 PRACTICAL CONSIDERATIONS

Our Diffusion Classifier method requires repeated error prediction evaluations for every class in order to classify an input image. These evaluations naively require significant inference time, even with the technique presented in Section 3.1.3. In this section, we present further insights and optimizations that reduce our method's runtime.

3.2.1 *Effect of timestep*

Diffusion Classifier, which is a theoretically principled method for estimating $p_\theta(\mathbf{c}_i \mid \mathbf{x})$, uses a uniform distribution over the timestep t for estimating the ϵ-prediction error. Here, we check if alternate distributions over t yield more accurate results. Figure 3.3 shows the Pets accuracy when using only a single timestep evaluation per class. Perhaps intuitively, accuracy is highest when using intermediate timesteps ($t \approx 500$). This begs the question: can we improve accuracy by oversampling intermediate timesteps and undersampling low or high timesteps?

We try a variety of timestep sampling strategies, including repeatedly trying $t = 500$ with many random ϵ, trying N evenly spaced timesteps, and trying the middle $t - N/2, \ldots, t + N/2$ timesteps. The tradeoff between different strategies is whether to try a few t_i repeatedly with many ϵ or to try many t_i once. Figure 3.4 shows that all strategies improve when taking using average error of more samples, but simply

using evenly spaced timesteps is best. We hypothesize that repeatedly trying a small set of t_i scales poorly since this biases the ELBO estimate.

3.2.2 *Efficient Classification*

A naive implementation of our method requires $C \times N$ trials to classify a given image, where C is the number of classes and N is the number of (t, ϵ) samples to evaluate to estimate each conditional ELBO. However, we can do better. Since we only care about $\arg\max_c p(\mathbf{c} \mid \mathbf{x})$, we can stop computing the ELBO for classes we can confidently reject. Thus, one option to classify an image is to use an upper confidence bound algorithm [5] to allocate most of the compute to the top candidates. However, this requires assuming that the distribution of $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_j)\|^2$ is the same across timesteps t, which does not hold.

We found that a simpler method works just as well. We split our evaluation into a series of stages, where in each stage we try each remaining $\mathbf{c}_i$ some number of times and then remove the ones that have the highest average error. This allows us to efficiently eliminate classes that are almost certainly not the final output and allocate more compute to reasonable classes. For example, on the Pets dataset, we have $N_{\text{stages}} = 2$. We try each class 25 times in the first stage, then prune to the 5 classes with the smallest average error. Finally, in the second stage we try each of the 5 remaining classes 225 additional times. With this evaluation strategy, classifying one Pets image requires 18 seconds on a RTX 3090 GPU.

Further reducing inference time could be a valuable avenue for future work. Inference is still impractical when there are many classes. Classifying a single ImageNet image, with 1000 classes, takes about 1000 seconds with Stable Diffusion at 512×512 resolution, even with our adaptive strategy. Table B.2 shows inference times for each dataset, and we discuss promising approaches for speedups in Section 3.5.

3.3 EXPERIMENTAL DETAILS

We provide setup details, baselines, and datasets for zero-shot and supervised classification.

3.3.1 *Zero-shot Classification*

DIFFUSION CLASSIFIER SETUP: Zero-shot Diffusion Classifier utilizes Stable Diffusion 2.0 [154], a text-to-image latent diffusion model trained on a filtered subset of LAION-5B [166]. Additionally, instead of using the squared ℓ_2 norm to compute the ϵ-prediction error, we leave the choice between ℓ_1 and ℓ_2 as a per-dataset inference

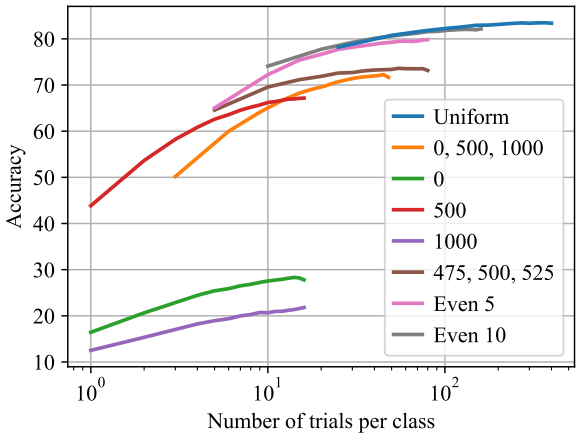


Figure 3.4: **Zero-shot scaling curves for different timestep sampling strategies**. We evaluate a variety of strategies for choosing the timesteps at which we evaluate the ϵ-prediction error. Each strategy name indicates which timesteps it uses— e.g., "0" only uses the first timestep, "0, 500, 1000" uses only the first, middle and last, "Even 10" uses 10 evenly spaced timesteps. We allocate more ϵ evaluations at the chosen timesteps as the number of trials increases. Strategies that repeatedly sample from a restricted set of timesteps, like "475, 500, 525", scale poorly with trials. Using timesteps uniformly from the full range [1, 1000] scales best.

hyperparameter. See Appendix B.3 for more discussion. We also use the adaptive Diffusion Classifier from Algorithm 3.

BASELINES: We provide results using two strong discriminative zero-shot models: (a) CLIP ResNet-50 [146] and (b) OpenCLIP ViT-H/14 [35]. We provide these for reference only, as these models are trained on different datasets with very different architectures from ours and thus cannot be compared apples-to-apples. We further compare our approach against two alternative ways to extract class labels from diffusion models: (c) **Synthetic SD Data**: We train a ResNet-50 classifier on synthetic data generated using Stable Diffusion (with class names as prompts), (d) **SD Features**: This baseline is not a zero-shot classifier, as it requires a **labeled dataset** of real-world images and class-names. Inspired by Label-DDPM [10], we extract Stable Diffusion features (mid-layer U-Net features at a resolution $[8 \times 8 \times 1024]$ at timestep $t = 100$), and then fit a ResNet-50 classifier on the extracted features and corresponding ground-truth labels. Details are in Appendix B.6.4.

DATASETS: We evaluate the zero-shot classification performance across eight datasets: Food-101 [19], CIFAR-10 [100], FGVC-Aircraft [119], Oxford-IIIT Pets [138], Flowers102 [131], STL-10 [39], ImageNet [43], and ObjectNet [11]. Due to computational

	Zero-shot?	Food	CIFAR10	Aircraft	Pets	Flowers	STL10	ImageNet	ObjectNet
Synthetic SD Data	✓	12.6	35.3	9.4	31.3	22.1	38.0	18.9	5.2
SD Features	✗	73.0	84.0	**35.2**	75.9	**70.0**	87.2	56.6	10.2
Diffusion Classifier (ours)	✓	**77.7**	**88.4**	26.4	**87.3**	66.3	**95.4**	**61.4**	**43.4**
CLIP ResNet-50	✓	81.1	75.6	19.3	85.4	65.9	94.3	58.2	40.0
OpenCLIP ViT-H/14	✓	92.7	97.3	42.3	94.6	79.9	98.3	76.8	69.2

Table 3.1: **Zero-shot classification performance**. Our zero-shot Diffusion Classifier method (which utilizes Stable Diffusion) significantly outperforms the zero-shot diffusion model baseline that trains a classifier on synthetic SD data. Diffusion Classifier also generally outperforms the baseline trained on Stable Diffusion features, despite "SD Features" using the entire training set to train a classifier. Finally, although making a fair comparison is difficult due to different training datasets, our generative approach surprisingly outperforms CLIP ResNet-50 and is competitive with OpenCLIP ViT-H. We report average accuracy or mean-per-class accuracy in accordance with [98].

constraints, we evaluate on 2000 test images for ImageNet. We also evaluate zero-shot compositional reasoning ability on the Winoground benchmark [184].

3.3.2 *Supervised Classification*

DIFFUSION CLASSIFIER SETUP: We build Diffusion Classifier on top of Diffusion Transformer (DiT) [141], a class-conditional latent diffusion model trained only on ImageNet-1k [43]. We use DiT-XL/2 at resolution 256^2 and 512^2 and evaluate each class 250 times per image.

BASELINES: We compare against the following discriminative models trained with cross-entropy loss on ImageNet-1k: ResNet-18, ResNet-34, ResNet-50, and ResNet-101 [72], as well as ViT-L/32, ViT-L/16, and ViT-B/16 [49].

DATASETS: We evaluate models on their in-distribution accuracy on ImageNet [43] and out-of-distribution generalization to ImageNetV2 [150], ImageNet-A [74], and ObjectNet [11]. ObjectNet accuracy is computed on the 113 classes shared with ImageNet. Due to computational constraints, we evaluate Diffusion Classifier accuracy on 10,000 validation images for ImageNet. We compute the baselines' ImageNet accuracies on the same 10,000 image subset.

3.4 EXPERIMENTAL RESULTS

In this section, we conduct detailed experiments aimed at addressing the following questions:

1. How does Diffusion Classifier compare against zero-shot state-of-the-art classifiers such as CLIP?
2. How does our method compare against alternative approaches for classification with diffusion models?
3. How well does our method do on compositional reasoning tasks?
4. How well does our method compare to discriminative models trained on the same dataset?
5. How robust is our model compared to discriminative classifiers over various distribution shifts?

3.4.1 *Zero-shot Classification Results*

Table 3.1 shows that Diffusion Classifier significantly outperforms the Synthetic SD Data baseline, an alternate zero-shot approach of extracting information from diffusion models. This is likely because the model trained on synthetically generated data learns to rely on features that do not transfer to real data. Surprisingly, our method also generally outperforms the SD Features baseline, which is a classifier trained in a *supervised* manner using the entire *labeled training set* for each dataset. In contrast, our method is zero-shot and requires no additional training or labels. Finally, while it is difficult to make a fair comparison due to training dataset differences, our method outperforms CLIP ResNet-50 and is competitive with OpenCLIP ViT-H.

This is a major advancement in the performance of generative approaches, and there are clear avenues for improvement. First, we performed no manual prompt tuning and simply used the prompts used by the CLIP authors. Tuning the prompts to the Stable Diffusion training distribution should improve its recognition abilities. Second, we suspect that Stable Diffusion classification accuracy could improve with a wider training distribution. Stable Diffusion was trained on a subset of LAION-5B [166] filtered aggressively to remove low-resolution, potentially NSFW, or unaesthetic images. This decreases the likelihood that it has seen relevant data for many of our datasets. The rightmost column in Table 3.2 shows that only 0-3% of the test images in CIFAR10, Pets, Flowers, STL10, ImageNet, and ObjectNet would remain after applying all three filters. *Thus, many of these zero-shot test sets are completely out-of-distribution for Stable Diffusion.* Diffusion Classifier performance would likely improve significantly if Stable Diffusion were trained on a less curated training set.

Dataset	Resolution	Aesthetic	SFW	A + S	R + A + S
Food	61.5	90.5	99.9	90.5	56.3
CIFAR10	0.0	3.4	90.3	3.2	0.0
Aircraft	98.6	95.7	100.0	95.6	94.4
Pets	1.1	89.1	100.0	89.1	0.9
Flowers	0.0	82.4	100.0	82.4	0.0
STL10	0.0	31.6	93.1	30.6	0.0
ImageNet	4.5	84.1	98.0	82.5	3.4
ObjectNet	98.8	20.5	98.8	20.3	20.2

Table 3.2: **How in-distribution is each test set for Stable Diffusion?** We show the percentage of each test set that would remain after the Stable Diffusion 2.0 data filtering process. The first three columns show the percentage of images that pass resolution ($\geqslant 512^2$), aesthetic ($\geqslant 4.5$), and safe-for-work ($\leqslant 0.1$) thresholds, respectively. The last two columns show the proportion of images that pass multiple filters, and the last column (R + A + S) corresponds to the actual filtering criteria used to train SD 2.0.

3.4.2 *Improved Compositional Reasoning Abilities*

Large text-to-image diffusion models are capable of generating samples with impressive compositional generalization. In this section, we test whether this generative ability translates to improved compositional *reasoning*.

WINOGROUND BENCHMARK: We compare Diffusion Classifier to contrastive models like CLIP [146] on Winoground [184], a popular benchmark for evaluating the visio-linguistic compositional reasoning abilities of vision-language models. Each example in Winoground consists of 2 (image, caption) pairs. Notably, both captions within an example contain the exact same set of words, just in a different order. Vision-language multimodal models are scored on Winoground by their ability to match captions C_i to their corresponding images I_i. Given a model that computes a score for each possible pair $\texttt{score}(C_i, I_j)$, the *text score* of a particular example $((C_0, I_0), (C_1, I_1))$ is 1 if and only if it independently prefers caption C_0 over caption C_1 for image I_0 and vice-versa for image I_1. Precisely, the model's text score on an example is:

$$\begin{aligned} \mathbb{I}[&\texttt{score}(C_0, I_0) > \texttt{score}(C_1, I_0) \text{ AND} \\ &\texttt{score}(C_1, I_1) > \texttt{score}(C_0, I_1)] \end{aligned} \tag{3.9}$$

Achieving a high text score is extremely challenging. Humans (via Mechanical Turk) achieve 89.5% accuracy on this benchmark, but even the best models do barely above

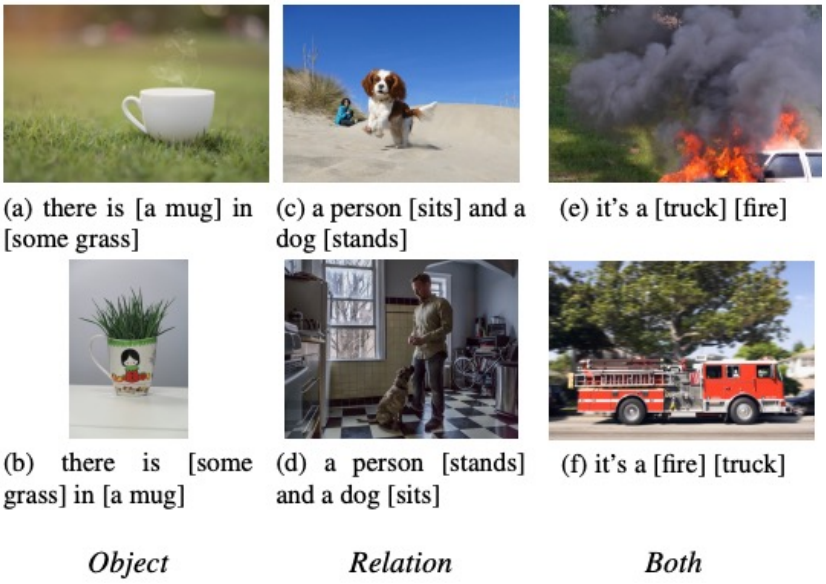


Figure 3.5: **Example visualizations of Winoground swap types.** Each category corresponds to a different type of linguistic swap in the caption. Object swaps noun phrases, Relation swaps verbs, adjectives, or adverbs, and Both can swap entities of both kinds.

chance. Models can only do well if they understand compositional structure within each modality. CLIP has been found to do poorly on this benchmark since its embeddings tend to be more like a "bag of concepts" that fail to bind subjects to attributes or verbs [198].

Each example is tagged by the type of linguistic swap (object, relation and both) between the two captions:

1. Object: reorder elements like noun phrases that typically refer to real-world objects/subjects.
2. Relation: reorder elements like verbs, adjectives, prepositions, and/or adverbs that modify objects.
3. Both: a combination of the previous two types.

We show examples of each swap type in Figure 3.5.

RESULTS Table 3.3 compares Diffusion Classifier to two strong contrastive baselines: OpenCLIP ViT-H/14 (whose text embeddings Stable Diffusion conditions on) and CLIP ViT-L/14. *Diffusion Classifier significantly outperforms both discriminative approaches on Winoground*. Our method is stronger on all three swap types, even the challenging "Relation" swaps where the contrastive baselines do no better than random guessing. This indicates that Diffusion Classifier's generative approach exhibits

Model	Object	Relation	Both	Average
Random Chance	25.0	25.0	25.0	25.0
CLIP ViT-L/14	27.0	25.8	57.7	28.2
OpenCLIP ViT-H/14	39.0	26.6	57.7	33.0
Diffusion Classifier (ours)	**46.1**	**29.2**	**80.8**	**38.5**

Table 3.3: **Compositional reasoning results on Winoground**. Diffusion Classifier obtains signficantly better text score (Eq. 3.9) than the contrastive baselines for all three swap categories.

better compositional reasoning abilities. Since Stable Diffusion uses the same text encoder as OpenCLIP ViT-H/14, this improvement comes from better cross-modal binding of concepts to images. Overall, we find it surprising that Stable Diffusion, trained with only sample generation in mind, can be repurposed into such a strong classifier and reasoner without any additional training.

3.4.3 *Supervised Classification Results*

We compare Diffusion Classifier, leveraging the ImageNet-trained DiT-XL/2 model [141], to ViTs [49] and ResNets [72] trained on ImageNet. This setting is particularly interesting because it enables a fair comparison between models trained on the same dataset. Table 3.4 shows that Diffusion Classifier outperforms ResNet-101 and ViT-L/32. Diffusion Classifier achieves ImageNet accuracies of 77.5% and 79.1% at resolutions 256^2 and 512^2 respectively. *To the best of our knowledge, we are the first to show that a generative model trained to learn* $p_\theta(\mathbf{x} \mid \mathbf{c})$ *can achieve ImageNet classification accuracy comparable to highly competitive discriminative methods.*

3.4.3.1 *Better Out-of-distribution Generalization*

We find that Diffusion Classifier surprisingly has stronger out-of-distribution (OOD) performance on ImageNet-A than all of the baselines. In fact, our method shows qualitatively different and better OOD generalization behavior than discriminative approaches. Previous work [181] evaluated hundreds of discriminative models and found a tight linear relationship between their in-distribution (ID) and OOD accuracy — for a given ID accuracy, no models do better OOD than predicted by the linear relationship. For models trained on only ImageNet-1k (no extra data), none of a wide variety of approaches, from adversarial training to targeted augmentations to different architectures, achieve better OOD accuracy than predicted. We show the relationship between ID ImageNet accuracy (subsampled to the classes that over-

Method	ID	OOD		
	IN	IN-V2	IN-A	ObjectNet
ResNet-18	70.3	57.3	1.1	27.2
ResNet-34	73.8	61.0	1.9	31.6
ResNet-50	76.7	63.2	0.0	36.4
ResNet-101	77.7	65.5	4.7	39.1
ViT-L/32	77.9	64.4	11.9	32.1
ViT-L/16	80.4	67.5	16.7	36.8
ViT-B/16	**81.2**	**69.6**	20.8	**39.9**
Diffusion Classifier 256^2	77.5	64.6	20.0	32.1
Diffusion Classifier 512^2	79.1	66.7	**30.2**	33.9

Table 3.4: **Standard classification on ImageNet.** We compare Diffusion Classifier (using DiT-XL/2 at 256^2 and 512^2 resolutions) to discriminative models trained on ImageNet. We highlight cells where Diffusion Classifier does better. All models (generative and discriminative) have only been trained on ImageNet.

lap with ImageNet-A) and OOD accuracy on ImageNet-A for these discriminative models as the blue points ("standard training") in Figure 3.6. The OOD accuracy is described well by a piecewise linear fit, with a kink at the ImageNet accuracy of the ResNet-50 model used to identify the hard images that comprise ImageNet-A. No discriminative models show meaningful "effective robustness," which is the gap between the actual OOD accuracy of a model and the OOD accuracy predicted by the linear fit [181].

However, in contrast to these hundreds of discriminative models, Diffusion Classifier achieves much higher OOD accuracy on ImageNet-A than predicted. Figure 3.6 shows that Diffusion Classifier lies far above the linear fit and achieves an effective robustness of 15-25%. *To the best of our knowledge, this is the first approach to achieve significant effective robustness without using any extra data during training.*

There are a few caveats to our finding. Diffusion Classifier does not show improved effective robustness on the ImageNetV2 or ObjectNet distribution shifts, though perhaps the nature of those shifts is different from that of ImageNet-A. Diffusion Classifier may do better on ImageNet-A since its predictions could be less correlated with the (discriminative) ResNet-50 used to find hard examples for ImageNet-A. Nevertheless, the dramatic improvement in effective robustness on ImageNet-A is exciting and suggests that generative classifiers are promising approaches to achieve better robustness to distribution shift.

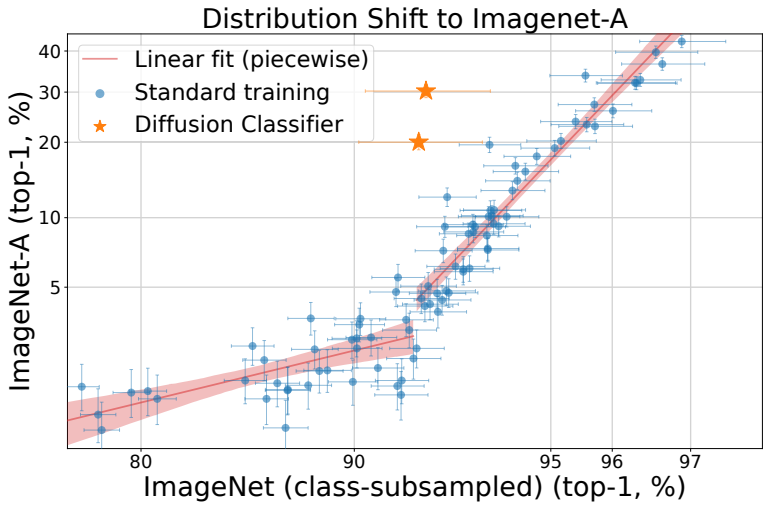


Figure 3.6: **Diffusion Classifier exhibits effective robustness without using extra labeled data**. Compared to discriminative models trained on the same amount of labeled data ("standard training"), Diffusion Classifier achieves much higher ImageNet-A accuracy than predicted by its ImageNet accuracy. Diffusion Classifier points correspond to DiT-XL/2 at resolution 256^2 and 512^2. Points are shown with 99.5% Clopper-Pearson confidence intervals. The red lines show the linear relationship between ID and OOD accuracy for discriminative models, with a "break" at the accuracy of the model used to create ImageNet-A. The axes were adjusted using logit scaling, since accuracies fall within $[0, 100]$.

3.4.3.2 *Stable Training and No Overfitting*

Diffusion Classifier's ImageNet accuracy is especially impressive since DiT was trained with *only random horizontal flips*, unlike typical classifiers that use RandomResizedCrop, Mixup [207], RandAugment [41], and other tricks to avoid overfitting. Training DiT with more advanced augmentations should further improve its accuracy. Furthermore, DiT training is stable with fixed learning rate and no regularization other than weight decay [141]. This stands in stark contrast with ViT training, which is unstable and frequently suffers from NaNs, especially for large models [69]. These results indicate that the generative objective $\log p_\theta(\mathbf{x} \mid \mathbf{c})$ could be a promising way to scale up training to even larger models without overfitting or instability.

3.4.3.3 *Choice of classification objective*

While Stable Diffusion parameterizes $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$ as a Gaussian with fixed variance, DiT learns the variance $\Sigma_\theta(\mathbf{x_t}, \mathbf{c})$. A single network outputs ϵ_θ and Σ_θ, but they are trained via two separate losses. ϵ_θ is trained via a uniform weighting of ℓ_2 errors

Resolution	Objective	IN	IN-v2	IN-A	ObjectNet
256^2	ℓ_2	**77.5**	**64.6**	**20.0**	**33.9**
	VLB	71.6	57.7	17.9	24.7
	ℓ_2 + VLB	**77.5**	**64.6**	**20.0**	33.8
512^2	ℓ_2	**79.1**	**66.7**	**30.2**	**33.9**
	VLB	74.0	59.1	24.9	24.7
	ℓ_2 + VLB	79.0	66.6	**30.2**	33.8

Table 3.5: **Effect of classification objective**. DiT trains ϵ_θ with the uniformly weighted ℓ_2 loss to evaluate $\sum_t w_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2$ from Eq. 3.3. DiT also trains the learned variance Σ_θ of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ with the exact variational lower bound, which weights timesteps unevenly. Since both of these weightings are involved in DiT training, we try each objective, as well as their sum, to see which one achieves the best accuracy. We find that uniformly weighting ℓ_2 errors across timesteps performs best.

$\mathbb{E}_{\epsilon,t}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2]$, as this is found to improve sample quality. In contrast, Σ_θ is trained with the exact variational lower bound. This keeps the timestep-dependent weighting term w_t in Eq. 3.3 and weights the ϵ-prediction errors by the inverse of the variances Σ_θ (see [141] for more details). Since both losses are used at training time, we run an experiment to see which objective yields the best accuracy as an inference-time objective. Instead of choosing the class with the lowest error based on uniform ℓ_2 weighting, as is done in Algorithm 2, we additionally try using the variational bound or the sum of the uniform weighting and the variational bound. Table 3.5 shows that the uniform ℓ_2 weighting does best across all datasets. This justifies the approximation we made to the ELBO in Eq. 3.4. The sum of the uniform ℓ_2 and the variational bound does almost as well, likely because the magnitude of the variational bound is much smaller than that of the uniformly weighted ℓ_2, so their sum is dominated by the ℓ_2 term.

3.5 DISCUSSION

We investigated the zero-shot and standard classification abilities of diffusion models by leveraging them as conditional density estimators. By performing a simple, unbiased Monte Carlo estimate of the learned conditional ELBO for each class, we extract **Diffusion Classifier**—a *powerful approach to turn any conditional diffusion model into a classifier without any additional training.* We find that this classifier narrows the gap with state-of-the-art discriminative approaches on zero-shot and standard classi-

fication and significantly outperforms them on multimodal compositional reasoning. Diffusion Classifier also exhibits far better "effective robustness" to distribution shift.

ACCELERATING INFERENCE While inference time is currently a practical bottleneck, there are several clear ways to accelerate Diffusion Classifier. Decreasing resolution from the default 512×512 (for SD) would yield a dramatic speedup. Inference at 256×256 is at least $4\times$ faster, and inference at 128×128 would be over $16\times$ faster. Another option is to use a weak discriminative model to quickly eliminate classes that are clearly incorrect. Appendix B.2 shows that this would simultaneously improve accuracy and reduce inference time. Gradient-based search could backpropagate through the diffusion model to solve $\arg\max_c \log p(x \mid c)$, which could eliminate the runtime dependency on the number of classes. New architectures could be designed to only use the class conditioning c toward the end of the network, enabling reuse of intermediate activations across classes. Finally, note that the error prediction process is easily parallelizable. With sufficient scaling or better GPUs in the future, all Diffusion Classifier steps can be done in parallel with the *latency of a single forward pass*.

ROLE OF DIFFUSION MODEL DESIGN DECISIONS Since we don't change the base diffusion model of Diffusion Classifier, the choices made during diffusion training affect the classifier. For instance, Stable Diffusion [154] conditions the image generation on the text embeddings from OpenCLIP [84]. However, the language model in OpenCLIP is much weaker than open-ended large-language models like T5-XXL [148] because it is only trained on text data available from image-caption pairs, a minuscule subset of total text data on the Internet. Hence, we believe that diffusion models trained on top of T5-XXL embeddings, such as Imagen [161], should display better zero-shot classification results, but these are not open-source to empirically validate. Other design choices, such as whether to perform diffusion in latent space (e.g. Stable Diffusion) or in pixel space (e.g. DALLE 2), can also affect the adversarial robustness of the classifier and present interesting avenues for future work.

In conclusion, while generative models have previously fallen short of discriminative ones for classification, today's pace of advances in generative modeling means that they are rapidly catching up. Our strong classification, multimodal compositional reasoning, and robustness results represent an encouraging step in this direction.

4

GENERATIVE CLASSIFIERS AVOID SHORTCUT SOLUTIONS

Ever since AlexNet [101], classification with neural networks has mainly been tackled with discriminative methods, which train models to learn $p_\theta(y \mid x)$. This approach has scaled well for in-distribution performance [49, 72], but these methods are susceptible to shortcut learning [57], where they output solutions that work well on the training distribution but may not hold even under minor distribution shift. The brittleness of these models has been well-documented [150, 181], but beyond scaling up the diversity of the training data [146] so that everything becomes in-distribution, no approaches so far have made significant progress in addressing this problem.

In this paper, we examine whether this issue can be solved with an alternative approach, called generative classifiers [130, 202, 210]. This method trains a class-conditional generative model to learn $p_\theta(x \mid y)$, and it uses Bayes' rule at inference time to compute $p_\theta(y \mid x)$ for classification. We hypothesize that generative classifiers may be better at avoiding shortcut solutions because their objective forces them to model the input x in its entirety. This means that they cannot just learn spurious correlations the way that discriminative models tend to do; they must eventually model the core features as well. Furthermore, we hypothesize that generative classifiers may have an inductive bias towards using features that are *consistently predictive*, *i.e.*, features that agree with the true label as often as possible. These are exactly the core features that models should learn in order to do well under distribution shift.

Generative classifiers date back at least as far back as Fischer discriminant analysis [55]. Generative classifiers like Naive Bayes had well-documented learning advantages [130] but were ultimately limited by the lack of good generative modeling techniques at the time. Today, however, we have extremely powerful generative models [21, 154], and some work is beginning to revisit generative classifiers with these new models [37, 106]. Li et al. [106] in particular find that ImageNet-trained diffusion models exhibit the first "effective robustness" [181] without using extra data, which suggests that generative classifiers are have fundamentally different (and perhaps better) inductive biases. However, their analysis is limited to ImageNet distribution shifts and does not provide any understanding. Our paper focuses on carefully com-

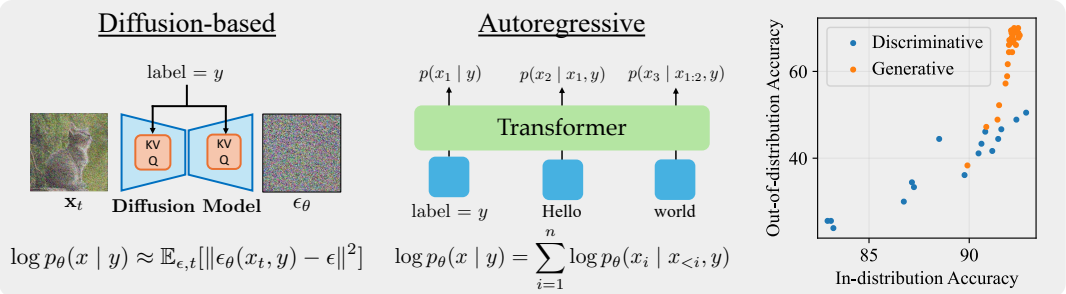


Figure 4.1: **Generative classifiers**. We repurpose today's best generative modeling algorithms for classification. Generative classifiers predict $\arg\max_y p_\theta(x \mid y)p(y)$. We use diffusion-based generative classifiers on image tasks and autoregressive generative classifiers on text tasks, and find that they scale better out-of-distribution than discriminative approaches.

paring deep generative classifiers against today's discriminative methods on a comprehensive set of distribution shift benchmarks. We additionally conduct a thorough analysis of the reasons and settings where they work. We list our contributions:

- **Show significant advantages of generative classifiers on realistic distribution shifts**. Generative classifiers are simple and effective compared to previous distribution shift mitigations. They utilize existing generative modeling pipelines, avoid additional hyperparameters or training stages, and do not require knowledge of the spurious correlations to avoid. We run experiments on standard distribution shift benchmarks across image and text domains and find that generative classifiers consistently do better under distribution shift than discriminative approaches. Most notably, they are the *first algorithmic approach* to demonstrate "effective robustness" [181], where they do better out-of-distribution than expected based on their in-distribution performance (see Figure 4.1, right). We also surprisingly find better in-distribution accuracy on most datasets, indicating that generative classifiers are also less susceptible to overfitting.

- **Understand why generative classifiers work**. We carefully test several hypotheses for why generative classifiers do better. We conclude that the generative objective $p(x \mid y)$ provides more consistent learning signal by forcing the model to learn all features of x.

- **Provide insights from Gaussian data**. We compare generative (linear discriminant analysis) and discriminative (logistic) classification methods on a simplified setting. We find the existence of "generalization phases" that show which kind of approach does better, depending on the strength of the spurious correlations

and noisy features in the data. These phases shed light on the inductive bias of generative classifiers towards low-variance features.

4.1 RELATED WORK

LEARNING IN THE PRESENCE OF SPURIOUS FEATURES It is well-known that deep networks trained with empirical risk minimization (ERM) have a tendency to rely on spurious correlations to predict the label, such as the background in an image or the presence of certain words [14, 57, 120, 152]. Notably, overfitting to these shortcuts causes a degradation in performance under distribution shift, since these spurious correlations may no longer be predictive [73, 155, 181]. The performance on rare ("minority") groups in particular tends to suffer [47, 159, 209], and this imbalance is aggravated in highly overparametrized models [160]. Theoretical works attribute this problem to the inductive bias of classifiers trained with cross-entropy loss; these classifiers prefer to find max-margin solutions, and thus fit spurious features even when they are not fully predictive like the core feature [128, 145]. To address these failures in discriminative models, people use objectives that try to balance learning across different groups [103, 159, 168], or add data augmentation to smooth out the spurious feature [171]. However, these methods still tend to fail to capture the core feature and often lead to degradations in in-distribution performance. Some approaches focus on identifying the specific spurious features, annotating which examples contain them, and using that to rebalance the data [60, 93, 197]. Unfortunately, these approaches require significant manual effort, are not as scalable, and may not work for problems where humans do not understand the learned features. Ideally, we find an approach with the right inductive bias to generalize well under distribution shift without requiring extra supervision.

CLASSIFICATION WITH GENERATIVE MODELS Few deep learning approaches have trained class-conditional generative models and used them directly for classification, perhaps due to the difficult task of modeling $p(x \mid y)$ with weaker generative models. However, recent generative models have significantly improved, especially with better techniques in diffusion probabilistic models [76, 173], and deep generative classification methods have recently been proposed [37, 106]. Li et al. [106] showed that ImageNet-trained class-conditional diffusion models are competitive with discriminative classifiers and achieve the first nontrivial "effective robustness" [181] on ImageNet-A [75] without using extra data. Prabhudesai et al. [144] show that a hybrid generative-discriminative classifier can use test-time adaptation to improve performance on several synthetic corruptions. Other work [37, 86] has shown that large pretrained generative models are more biased towards shape fea-

tures and more robust to *synthetic* corruptions, but this may be due to effect of pretraining on extra data, or the fact that diffusion specifically confers resilience to input perturbations. Other works have found that generative classifiers can improve adversarial robustness [27, 28, 63, 213]. However, *adversarial robustness* has been shown to not translate to *robustness to distribution shift* [164, 181]. Overall, it still remains unclear whether generative classifiers are more robust to the spurious correlations seen in *realistic* distribution shifts or *why* they might be better.

4.2 PRELIMINARIES

4.2.1 *Types of Distribution Shift*

We consider classification under two types of distribution shift. In subpopulation shift, there are high-level spurious features that are correlated with the label. For example, on CelebA [115], where the task is to predict whether a person's hair is blond or not blond, the spuriously correlated feature is the gender. This occurs because there are very few blond men in the dataset, so models typically learn to use the "man" feature. The spurious feature determines groups: the majority group contains examples where the spurious feature is correct, and the minority group contains examples where the spurious feature is incorrect. We also consider domain shift, where the test domain's data distribution is similar to the training domain's distribution. For example, training images in Camelyon17-WILDS [95] come from 3 hospitals, whereas the test images come from a disjoint 4th hospital. Spurious features that worked on the training distribution, *e.g.*, artifacts of the way slide staining or sample collection was done, may hurt accuracy under distribution shift [97, 182, 191]. We examine 5 common distribution shift benchmarks in total: besides CelebA and Camelyon, we use Waterbirds (Sagawa et al. [159]; subpopulation shift), FMoW (Koh et al. [95]; both subpopulation and domain shift), and CivilComments (Koh et al. [95]; subpopulation shift).

4.2.2 *Shortcomings of Discriminative Classifiers*

Discriminative classifiers, which seek to maximize $p_\theta(y \mid x)$, can overly rely on the spurious features and fall victim to shortcut solutions [57]. This is because they can use the spuriously correlated features to correctly and confidently fit the majority group examples. After this happens, the loss on these examples flattens out, and there is less gradient signal available to encourage the model to use core features [109, 142]. The model then overfits to the remaining minority examples where the spurious correlation does not help [128, 160]. These shortcut solutions often work

in-distribution but can fail, sometimes catastrophically, under even minor distribution shift. Significant effort has been put into preventing this, mainly by rebalancing the data so that the spurious correlation no longer holds [93, 112, 159, 168]. However, these methods all add additional hyperparameters and complexity to the training process, and often require knowledge of the exact distribution shift to counteract, which is impractical for realistic problems where there may be many spurious correlations.

4.3 GENERATIVE CLASSIFIERS

We now present generative classifiers, a simple paradigm for classification with class-conditional generative models. To classify an input x, generative classifiers first compute $p_\theta(x|y)$ with a class-conditional generative model and then utilize Bayes' theorem to obtain $p_\theta(y|x)$. This paradigm had been popular in machine learning with methods like linear discriminant analysis and Naive Bayes [130], but has fallen out of favor in the modern era of deep learning. We revisit this paradigm with deep learning architectures and show its advantages for robustness to distribution shift in Section 4.4. Algorithm 4 gives an overview of the generative classification procedure.

4.3.1 *Intuition*

Why could generative classifiers do better on these distribution shifts? In contrast to discriminative classifiers, which can minimize their training objective using just a few spurious features, generative classifiers need to model the entire input x. This means that they cannot stop at just the spurious features; their training objective requires them to learn both core and spurious features. This should translate to better training signal throughout the course of the training. We confirm this in Section 4.4.3. Note that learning both types of features does not mean that it uses them equally when classifying an input. The generative classifier should learn which type of features are more consistently correlated with the label and weight them accordingly. Section 4.5.3 and 4.5.4 demonstrates this inductive bias in a simple setting with Gaussian data.

4.3.2 *Diffusion-based Generative Classifier*

For image classification, we use diffusion models [76, 173], which are currently the state-of-the-art approach for image modeling. Diffusion models are trained to iteratively denoise an image and do not have an exact likelihood that can be computed in a single forward pass. They are typically trained with a reweighted variational lower

bound of $\log p_\theta(x \mid y)$. To use them in a generative classification framework, we use that value to approximate $\log p_\theta(x \mid y)$:

$$\log p_\theta(x \mid y) \approx \mathbb{E}_{\epsilon,t}[\|\epsilon_\theta(x_t, y) - \epsilon\|^2] \tag{4.1}$$

Training the class-conditional diffusion models is done as normal – we use off-the-shelf training pipelines to train diffusion models from scratch, without modifying any hyperparameters. At inference time, we follow the Diffusion Classifier algorithm from [106], which samples multiple noises ϵ, adds them to the image to obtain noised $x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon$, and does multiple forward passes through the network to obtain a Monte Carlo estimate of Eq. 4.1. This is done for each class, and the class with the highest conditional likelihood $\log p_\theta(x \mid y)$, which corresponds to the lowest denoising error, is chosen.

4.3.3 *Autoregressive Generative Classifier*

For text classification, we introduce generative classifiers built on autoregressive Transformer models, as they are the dominant architecture for text modeling. Since we need to now learn $p_\theta(x \mid y)$, where x is a sequence of text tokens and y is a label, we make a small modification to the training procedure. Instead of starting each sequence of text tokens with a "beginning of sequence" (BOS) token, we allocate C special class tokens in our vocabulary, one per class, and replace BOS with the desired class token. Obtaining $\log p_\theta(x \mid y)$ can be done in a single forward pass:

$$\log p_\theta(x \mid y) = \log\left(\prod_{i=1}^{n} p_\theta(x_i \mid x_{<i}, y)\right) = \sum_{i=1}^{n} \log p_\theta(x_i \mid x_{<i}, y) \tag{4.2}$$

We train our Transformer as usual using cross-entropy loss over the entire sequence, with the ground truth label y^* at the beginning. To classify a text sequence at inference time, we do C forward passes, one with each possible class token. We then choose the class token with the lowest cross-entropy loss over the entire sequence as our prediction. Figure 4.1 (middle) shows a diagram of this method.

Overall, generative classifiers can be easily trained using existing generative modeling pipelines and do not require any specialized architectures, extra hyperparameters, data augmentation, multi-stage training, or knowledge of the specific shortcuts to avoid. Training is also cheap: all of our generative models can be trained on a single GPU in 2-3 days.

Method	Waterbirds		CelebA		Camelyon		FMoW		CivilComments	
	ID	WG	ID	WG	ID	OOD	ID	OOD WG	ID	WG
ERM	88.8	32.2	92.4	50.5	95.2	78.3	51.1	27.5	**90.6**	53.3
LfF [129]	86.4	28.9	90.8	34.0	90.5	66.3	49.6	31.0	87.9	49.4
JTT [112]	88.1	32.9	91.9	42.1	88.1	65.8	52.1	31.8	89.2	55.6
RWY/DFR	90.8	31.6	**94.1**	68.9	95.2	78.3	39.3	26.1	90.1	58.1
Generative (ours)	**96.8**	**79.4**	91.2	**69.4**	**98.3**	**90.8**	**62.8**	**35.8**	79.8	**61.4**

Table 4.1: **Accuracy on distribution shift benchmarks**. We show in-distribution (ID) and either worst-group (WG) or out-of-distribution (OOD) accuracy, depending on the type of shift in each dataset. Our generative approach performs the best on all five distribution shifts and 3/5 ID datasets.

4.4 EXPERIMENTS

We now compare our generative classification approach to discriminative methods that are commonly used today. We aim to answer the following questions in this section. First, do generative classifiers have better robustness to distribution shift? If so, why are they more robust than discriminative methods? We test multiple hypotheses to determine which explanation is correct.

4.4.1 *Setup*

Benchmarks We use five standard benchmarks for distribution shift. Camelyon undergoes domain shift, so we report its OOD accuracy on the test data. Waterbirds, CelebA, and CivilComments undergo subpopulation shift, so we report worst group accuracy. FMoW has both subpopulation shift over regions and a domain shift across time, so we report OOD worst group accuracy. The first four are image benchmarks, while CivilComments is text classification. Waterbirds and CelebA are natural images, whereas Camelyon contains whole-slide images of cells and FMoW contains satellite images. In total, these benchmarks cover multiple shift types, modalities, and styles.

MODEL SELECTION We believe that it is unrealistic or impractical to know the exact distribution shift that will happen on the test set. Thus, we do not use knowledge of the spurious correlation or distribution shift when training or performing model selection, and instead tune hyperparameters and perform early stopping on the in-distribution validation accuracy, not the worst-group accuracy. This is the most

valuable setting, as it matches how models are often deployed in practice, and thus is a popular experimental setting for evaluating methods [95, 112, 168, 200]. We use class-balanced accuracy for model selection as it uniformly improves performance on each dataset for all methods [83].

BASELINES We compare generative classifiers against several discriminative baselines. ERM minimizes the average cross-entropy loss of the training set and is the standard method for training classifiers. We additionally evaluate several methods designed to combat spurious features. Learning from Failure (LfF) [129] simultaneously trains one network to be biased and uses it to identify samples that a second network should focus on. Just Train Twice (JTT) [112] is a similar two-stage method that first trains a standard ERM model for several epochs, and then heuristically identifies worst-group examples as training points with high loss under the first model. JTT then upsamples these points and trains a second classifier. DFR [93] fine-tunes a model on a training set that has been carefully balanced to make the spurious feature unpredictive. As in prior work [200], DFR samples data from each class equally when there are no spurious feature annotations. This is equivalent to RWY [83] and can help if there is class imbalance related to the spurious correlation. For fairness, we train all models, generative and discriminative, from scratch to eliminate the effect of differing pre-training datasets.

MODELS For image-based tasks, all discriminative baselines use ResNet-50, ResNet-101, and ResNet-152, whereas our generative classifier approach trains a class-conditional U-Net-based latent diffusion model [154]. For text-based tasks, all discriminative baselines use an encoder-only Transformer, whereas our generative classifier approach trains a Llama-style autoregressive language model [186] from scratch. See Appendix C.2 for more details.

4.4.2 *Results on Distribution Shift Benchmarks*

MAIN RESULTS Table 4.1 compares generative classifiers against discriminative baselines on the distribution shift benchmarks. Compared to the discriminative baselines, generative classifiers have better worst-group or OOD accuracy on all five datasets. Surprisingly, generative classifiers also achieve *significantly* better in-distribution accuracy on three of the five datasets, which indicates less overfitting. These results suggest that generative classifiers may have an advantage in both (a) learning core features that generalize across distribution shifts, and (b) learning features that generalize from the training set to the ID test set.

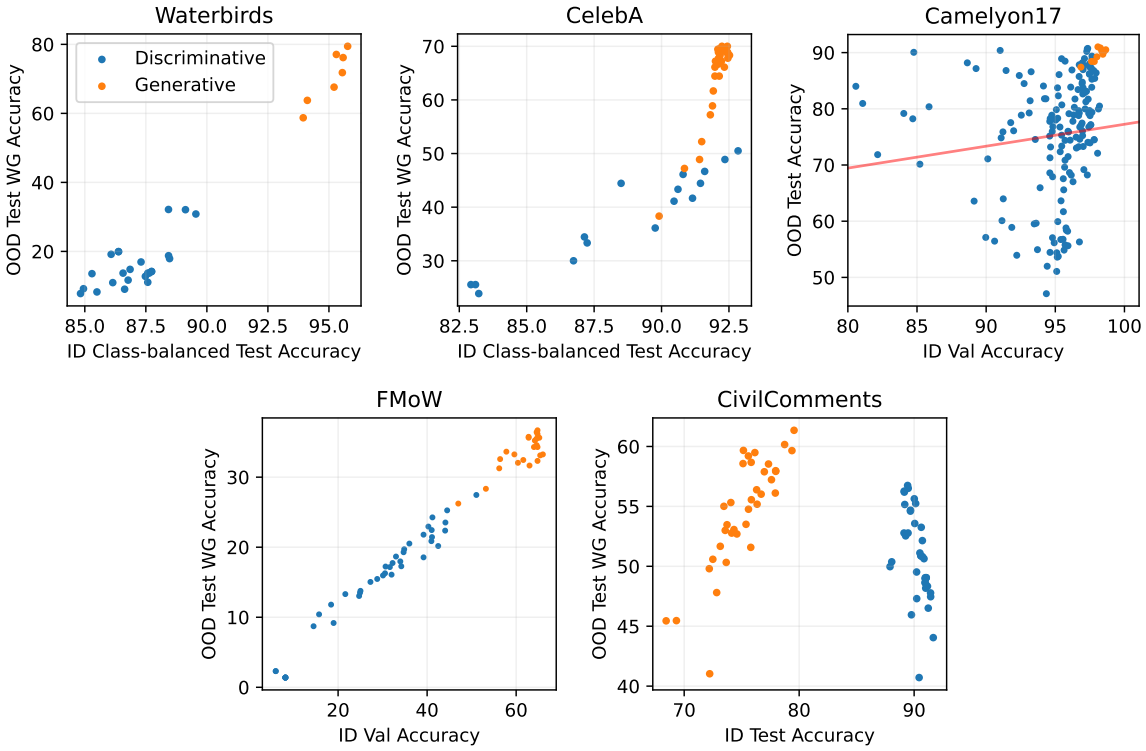


Figure 4.2: **In-distribution vs out-of-distribution accuracy** for each dataset. Each point corresponds to a separate training run, other than the diffusion-based generative classifier results, which are checkpoints of a run with default hyperparameters. We observe better OOD scaling trends (i.e., effective robustness) for generative classifiers on CelebA, CivilComments, and potentially Camelyon17, although results are noisy for this dataset (the red line in Camelyon17 denotes a linear fit for the relationship between ID and OOD accuracy for discriminative models). On the remaining two datasets, they follow the same trend and do better both ID and OOD.

ACCURACY ABOVE THE LINE Comparing the best generative classifier against the best discriminative classifier provides a one-dimensional understanding of each approach. To provide a better sense of which method may scale better in the future, Figure 4.2 plots the in-distribution and out-of-distribution accuracies of each family of methods. We can classify the benchmarks into two sets:

1. Generative classifiers are better both ID and OOD, and lay on the same trend line as discriminative models. This includes Waterbirds and FMoW.

2. Generative classifiers have a significantly better OOD performance *trend*, but are the same or worse in-distribution. This includes CelebA, CivilComments, and potentially Camelyon.

The second case, where generative classifiers have better OOD accuracy than discriminative classifiers at any ID accuracy, demonstrates "effective robustness" [181]. This suggests fundamentally better out-of-distribution behavior for generative classifiers in some scenarios and indicates that they may be the right approach to classification after further scaling. Our results corroborate findings from Li et al. [106], which found some signs of effective robustness on ImageNet. Section 4.5 examines a toy setting and provides insights into the cause of this "effective robustness."

4.4.3 *Why Do Generative Classifiers Do Better?*

We test several hypotheses for how generative classifiers outperform the discriminative baselines.

LEARNING MORE FROM MAJORITY EXAMPLES Our intuition is that the generative objective $\log p_\theta(x \mid y)$ provides more consistent learning signal across epochs. In contrast, discriminative models may use spurious features to make confident and correct predictions on the training set and lose the gradient signal necessary to use the core features. We test this by measuring the gradient norm on majority and minority examples across epochs. We compute the per-example gradient norm $\|\nabla_\theta \mathcal{L}(x_i, y_i)\|_2$ and average it over the majority and minority groups. We normalize this by the average majority group gradient norm at epoch 5 in order to fairly compare different architectures that have different loss landscapes. Figure 4.3 shows these metrics on CivilComments with toxic comments about the Black demographic as the minority group. For the discriminative model, the majority group gradient quickly vanishes, and the minority group gradient starts high but eventually decays. The generative classifier, however, has very similar gradient norm across the majority and minority groups, and the gradient norm actually slightly increases over training. These results support our intuition that the generative objective helps the model learn more from examples with and without the spurious features. Note: the *per-example* gradient norm for generative classifiers *does not* go to 0, since there is always a way to increase the likelihood of a single data point.

DOES AN UNCONDITIONAL OBJECTIVE $p(x)$ IMPROVE DISCRIMINATIVE PERFORMANCE? One hypothesis is that the generative classification objective $p_\theta(x \mid y)$ teaches the model better features in general, similar to how generative pre-training methods [44, 70] learn features that are useful for fine-tuning. We test this on Civil-

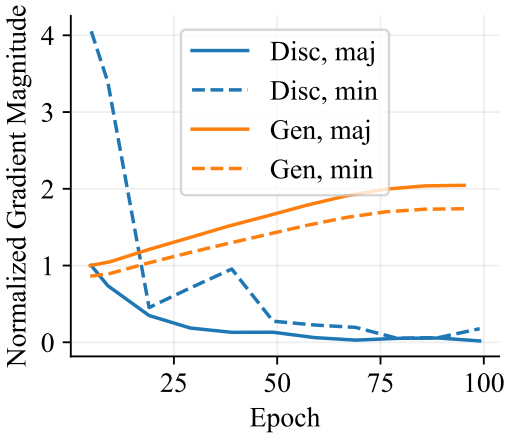


Figure 4.3: **Gradient norms**. Gradient for the discriminative model rapidly decays to 0, so its learning signal is reduced, while it does not decay for generative classifier.

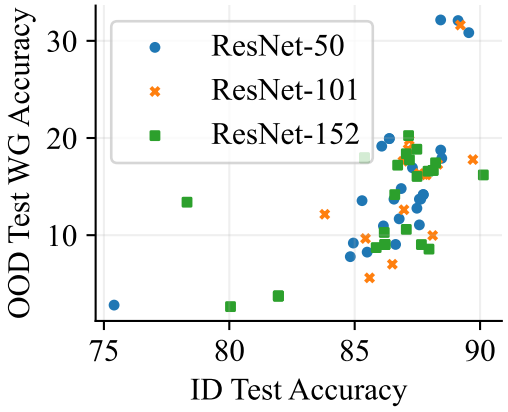


Figure 4.4: **Scaling discriminative model size** does not improve accuracy on Waterbirds. This shows that model size is *not* a confounder in our experiments.

Train Objective	ID	WG
$p(y \mid x)$	91.4	35.7
$p(x)$ and $p(y \mid x)$	**91.7**	35.4
$p(x \mid y)$ (ours)	79.8	**61.4**

Table 4.2: **Alternative training objectives** for an autoregressive model on CivilComments. $p(y \mid x)$ is a standard discriminative approach with cross-entropy loss, and "$p(y \mid x)$ and $p(x)$" tests if adding an unconditional generative modeling improves performance.

Comments, as the architecture makes it simple to add an unconditional generative objective $p(x)$. Instead of placing the class-specific token at the beginning of the sequence, we place it at the end. Predicting the text tokens of x now corresponds to predicting $p(x)$, and predicting the class-specific token at the end corresponds to $p(y \mid x)$. Table 4.2 shows that adding the objective $p(x)$ does not affect performance, so we reject this hypothesis.

MODEL SIZE In our image classification experiments, our generative classifier used a standard 395M parameter UNet [154], which is far more than the 26M parameters in its ResNet-50 [72] discriminative counterpart. Could the greater parameter

count could be responsible for the difference in performance and OOD behavior? We first note that the discriminative classifier used for CivilComments in Table 4.1 contains 67M parameters, which is more than the 42M parameters we use in our autoregressive generative classifier. Furthermore, the architectures and parameter counts of the discriminative $p(y \mid x)$ and generative $p(x \mid y)$ classifiers are exactly matched in Table 4.2. On image tasks, we test whether parameter count matters by scaling from ResNet-50 up to ResNet-152 [72]. We perform a sweep over model size and training hyperparameters (learning rate and weight decay). Figure 4.4 and Figure C.1 show that increasing discriminative model size does not improve performance. This aligns with previous work: Sagawa et al. [160] found that increasing discriminative model size can actually *hurt* OOD performance.

4.5 ILLUSTRATIVE SETTING

We explore a simplified Gaussian data setting and find that linear generative classifiers can also display better robustness to distribution shift compared to their discriminative counterparts. We rigorously explore this behavior in order to understand the inductive bias of generative classifiers. Finally, we connect our findings back to practice and explain the varying empirical behavior for generative vs discriminative classifiers.

4.5.1 *Data*

Consider binary classification with label $y \in \{-1, +1\}$. The features are $x = (x_{\text{core}}, x_{\text{spu}}, x_{\text{noise}}) \in \mathbb{R}^d$, where:

$$x_{\text{core}} \mid y = \mathcal{N}(y, \sigma^2) \in \mathbb{R} \tag{4.3}$$

$$x_{\text{spu}} \mid y = y\mathcal{B} \text{ w.p. } \rho, \text{ else } -y\mathcal{B} \in \mathbb{R} \tag{4.4}$$

$$x_{\text{noise}} \mid y = \mathcal{N}(0\,, \sigma^2_{\text{noise}}) \in \mathbb{R}^{d-2} \tag{4.5}$$

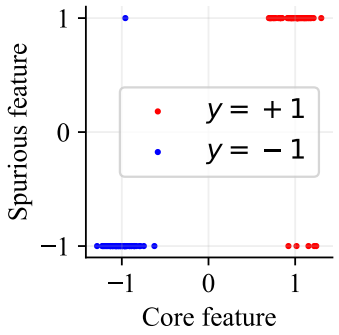


Figure 4.5: Visualization of features (noise dims not shown).

We set the spurious correlation ratio $\rho = 0.9$ and core feature standard deviation $\sigma = 0.15$, which is small enough that the data can be perfectly classified by using only the core feature x_{core} and ignoring the remaining features. Figure 4.5 shows a visualization of the core and spurious features. The majority groups consist of samples where the spurious and core features agree (top right and bottom left of Fig. 4.5), and the minority groups consist of samples where the spurious and core features disagree (top left and bottom right).

This synthetic dataset has previously been used to understand the failure modes of discriminative classifiers in previous work [83, 160, 168] and is a natural simplified setting for us to study the advantages of generative classifiers.

4.5.2 *Algorithms*

DISCRIMINATIVE We analyze unregularized logistic regression, as is done in previous work [128, 160]. Since the data is linearly separable, logistic regression learns the max-margin solution when trained via gradient descent [177].

GENERATIVE We use linear discriminant analysis (LDA), a classic generative classification method that models each class as a multivariate Gaussian. It fits separate class means μ_{-1} and μ_{+1} but learns a shared covariance matrix Σ for both classes. Assuming balanced classes, LDA predicts:

$$\arg\max_{y} p(x \mid y) = \text{sign}\left(\log \frac{p(x \mid y = +1)}{p(x \mid y = -1)}\right) = \text{sign}\left(\log \frac{\mathcal{N}(x \mid \mu_{+1}, \Sigma)}{\mathcal{N}(x \mid \mu_{-1}, \Sigma)}\right) \tag{4.6}$$

This corresponds to a linear decision boundary with coefficients $w_{\text{LDA}} = \Sigma^{-1}(\mu_{+1} - \mu_{-1})$. We chose LDA because it has the least inductive bias among common linear generative classifiers (*e.g.*, it is equivariant to rotations). For this reason, we rejected methods like Naive Bayes, which learns an axis-aligned generative model, even though theoretical analysis would have been easier.

4.5.3 *The Inductive Bias of LDA*

We carefully examine a setting where the generative approach has the same in-distribution performance, but it outperforms the discriminative approach on the worst group (OOD). Figure 4.6 compares the behavior of LDA and logistic regression on toy data with data dimension $d = 1026$ and noise variance $\sigma^2_{\text{noise}} = 0.36$. We find that both methods have similar in-distribution accuracies, but LDA does significantly better on the minority group. In fact, Figure 4.6 (middle) shows that LDA has essentially no performance gap between the majority and minority groups, which indicates that it *does not use the spurious feature at all*. In contrast, logistic regression has a large performance gap between the groups. This can be explained by looking at the linear coefficients learned by both methods. Figure 4.6 (right) shows the ratio $|w_{\text{spu}}|/|w_{\text{core}}|$ between the weights on the shortcut and core features. Ideally, this ratio goes to 0 as fast as possible as the model sees more data. Logistic regression, however, places significant weight on the spurious feature until it gets thousands of training examples. LDA is far more data-efficient and places almost no weight on

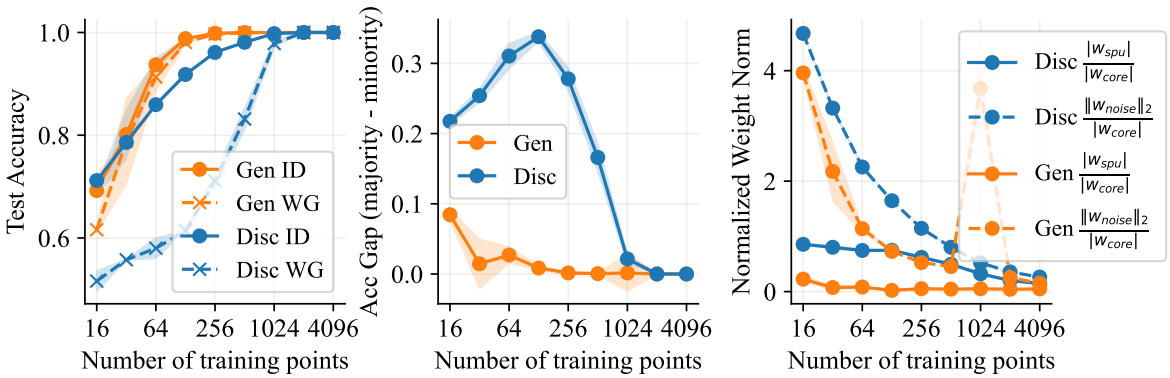


Figure 4.6: **Illustrative setting for shortcut learning. Left**: in-distribution accuracies are roughly the same between generative (LDA) and discriminative (logistic regression) methods, but LDA achieves much higher minority group accuracy. **Middle**: the difference between the majority and minority test accuracies as a function of the number of training examples. The generative method displays better robustness to the spurious correlation. **Right**: spurious feature weights $|w_{\mathrm{spu}}|$ and noise feature weights $\|w_{\mathrm{noise}}\|_2$, normalized by the magnitude of the core feature weight w_{core}. LDA puts much less weight on the spurious feature, even with very little training data. Logistic regression puts more weight on the noise, and only achieves good in-distribution accuracy by using the spurious feature. Shading denotes ± 1 standard deviation over 25 seeds.

the spurious feature with as few as 16 training examples. Interestingly, logistic regression puts more weight on the noisy features than LDA does. It is only by putting significant weight on the spurious feature that it achieves good in-distribution performance, though this hurts worst group accuracy.

These differences in behavior indicate that LDA has a significantly different inductive bias. We suspect that the most important factor is the core feature variance σ^2. We increased σ from 0.15 to 0.6 and reran the same analysis. Figure C.7 shows that LDA now consistently underperforms logistic regression, both in-distribution and out-of-distribution. Why is this? Intuitively, when the learned probability $p_\theta(x_i \mid y^*)$ of feature x_i is low (*i.e.*, the feature is not consistently correlated with the label) compared to other features, the generative classifier downweights this feature in its prediction. This helps improve robustness to distribution shift, since, by definition, we believe that the core feature x_{core} should be the most consistently predictive.

4.5.4 *Generalization Phase Diagrams*

The feature dimension d, spurious feature scale $\mathcal{B}$, noisy feature variance σ^2_{noise}, and core feature variance σ^2 influence the solutions that models prefer to learn. Intuitively, the spurious feature scale $\mathcal{B}$ controls the saliency of the shortcut feature, and larger $\mathcal{B}$ makes it easier for the model to learn this shortcut. The noisy feature variance σ^2_{noise} controls how easy it is for a model to overfit to training examples [128]. The core feature variance σ^2 controls how consistently the core feature predicts the label. Varying these properties of the data creates a family of datasets, and we use them to understand when generative classifiers outperform their discriminative counterparts.

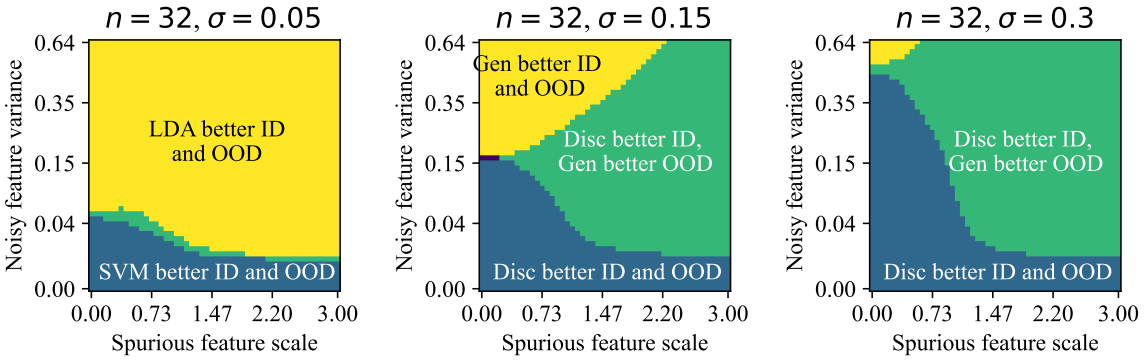


Figure 4.7: **Generalization phase diagrams**. We vary the scale $\mathcal{B}$ of the spurious feature and the variance σ^2_{noise} of the noise features and evaluate their effect on the ID and OOD test accuracy of generative classifiers (LDA) vs discriminative classifiers (logistic regression). Each plot corresponds to a different variance σ^2 of the core feature, and the color of each pixel denotes which classifier does better for a particular combination of $\mathcal{B}$ and σ^2_{noise}. We observe three main phases of generalization: (1) discriminative has better ID and OOD accuracy, (2) generative has better ID and OOD accuracy, and (3) discriminative does better ID and generative does better OOD. Each plot corresponds to a different standard deviation σ of the core feature. As σ increases, the core feature becomes less reliable, and the generative classifier uses the spurious and noise features more. This shows the inductive bias of generative classifiers: they prefer *consistently predictive features*.

Each plot in Figure 4.7 varies $\mathcal{B}$ and σ^2_{noise}, for a given number of training examples $n = 32$ and $d = 1024$. Each successive subplot corresponds to increasingly larger variance σ^2 in the core feature, and each plot is divided into regions depending on which method does better ID or OOD for the given $(\mathcal{B}, \sigma^2_{\text{noise}})$ at that location. We call this a *generalization phase diagram*, since it resembles a phase diagram which

shows the impact of pressure and temperature on the physical state of a substance. In our case, there are four possible generalization phases:

1. The generative classifier is better both ID and OOD. This typically happens at high σ^2_{noise}, since the discriminative model overfits using the noise features.

2. The discriminative classifier is better both ID and OOD. This happens at low σ^2_{noise}.

3. The discriminative classifier is better ID, but the generative classifier is better OOD. This phase occurs at a sweet spot of $\mathcal{B}$ and σ^2_{noise}. Moderate noise allows some overfitting, but the spurious feature is strong enough for the discriminative model to achieve good ID accuracy. However, its heavy reliance on the spurious feature reduces its OOD performance.

4. The generative classifier is better ID, but the discriminative classifier is better OOD. This is exceedingly rare (see the dark, unlabeled regions in Figure 4.7).

Notably, there is no free lunch. Even in this setting, neither generative nor discriminative classifiers are uniformly better than the other. However, we do note that $\mathcal{B}$ and σ^2_{noise} are unbounded above, and generative classifiers should do comparatively better as the strength of shortcuts or noise increases.

Finally, while it is hard to map $\mathcal{B}$ and σ^2_{noise} directly onto a realistic image or text dataset, they do offer insights on important properties of the data that determine which method is suitable for a given task. Indeed, we can categorize the distribution shift benchmarks into these phases based on their generative vs discriminative behavior. Waterbirds and FMoW fall in phase 1 (generative better ID and OOD), CelebA and CivilComments fall in phase 3 (discriminative better ID and generative better OOD), and Camelyon lies on the transition boundary between phase 1 and 3, since the generative classifier achieves better OOD and similar ID accuracy compared to discriminative baselines.

4.6 DISCUSSION

Discriminative approaches to classification have dominated the field since AlexNet catalyzed the widespread adoption of deep learning. Despite their prevalence, these methods face increasing limitations, including vulnerability to distribution shift and escalating data requirements. In this thesis, we present a simple alternative. We revisit the paradigm of generative classifiers and show that they have significant advantages in both in-distribution and out-of-distribution performance on realistic distribution shift benchmarks. We carefully analyze their behavior, and finally show

insights from a simplified, illustrative setting into when generative classifiers can be expected to do better.

As deep generative classifiers have not been well-explored, there is significant room for future work. The inference cost of these generative classifiers, especially diffusion-based ones, is currently impractically high. It is also unclear how to incorporate complex augmentations, such as Mixup, into generative classifiers. Finally, the ideas from this work may be useful in other contexts, such as language modeling. Tasks like sentiment analysis, code completion, and reasoning are currently being done in a discriminative approach: given a context x, predict the correct answer y by sampling from $p_\theta(y \mid x)$. Improving the performance and out-of-distribution robustness of these models by doing a generative approach $p(x \mid y)$ would be a particularly exciting direction.

Part III

ALGORITHM DESIGN

5

DISCRETE DIFFUSION IS GENERALIZED AUTOREGRESSION

Language models today are extremely powerful, general-purpose algorithms that can solve a wide variety of tasks, from coding to mathematics to other natural language problems [21, 110]. These models are typically autoregressive, which generate sequences one token at a time, from left to right. This typically works well in practice but has several drawbacks. First, the left-to-right ordering may hurt generalization. It can be difficult to properly put information in the context, such as when performing infilling [13]. Some problems are also too difficult to solve from left-to-right, such as Sudoku or addition. Second, sampling is slow. Generating one token per forward pass makes sampling memory-bound, as the weights of the model have to be loaded into accelerator memory once per generated token. This may be unacceptable in certain latency-sensitive applications.

Diffusion language models [6, 62, 66, 108] offer a potential solution to both problems. They can flexibly condition on any input, and they can generate an entire sequence in relatively few steps. Diffusion language models can either generate sequences in a continuous latent space [66, 108], or they can directly predict discrete tokens in token space [6, 23, 117, 122]. Despite significant research into this area, it is still not clear what the right formulation is.

In this chapter, we aim to make theoretical, architectural, and algorithmic improvements to discrete diffusion language models and show that they offer a compelling alternative to today's autoregressive models. We first show that autoregression is a special case of discrete diffusion and prove that the most successful type of discrete diffusion – absorbing diffusion – can be cast as autoregression over a generalized sequence ordering. Using this view, we propose Prism, a language model that can generate sequences in any desired ordering (Figure 5.1). We measure scaling laws and find that Prism is more expensive to train than autoregressive models, but the extra training compute spent training on generalized sequence orderings buys several advantages. Prism can generate sequences with lower latency than autoregressive models and higher throughput than previous diffusion models. Finally, we show im-

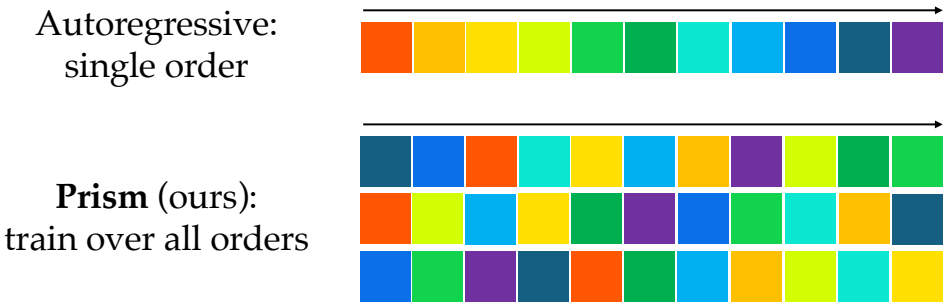


Figure 5.1: Our method, Prism, learns to generate sequences in any order. We show in Section 5.1 that this is equivalent to absorbing diffusion models, but this perspective has significant computational advantages.

proved performance on reasoning tasks, especially when using a confidence-based decoding strategy to determine which sampling order to use.

5.1 CONNECTING DISCRETE DIFFUSION AND AUTOREGRESSION

5.1.1 *Preliminaries*

In general, diffusion models are latent variable generative models with a corresponding forward and reverse process. The forward process $q(x_t \mid x_{t-1})$ gradually adds noise to the original data x_0, whereas the learned reverse process $p_\theta(x_t \mid x_{t+1})$ learns to undo this noise. Diffusion models are typically trained by minimizing a variational upper bound on the negative log-likelihood $-\log p_\theta(x)$ [76]:

$$L_{vb} = \mathbb{E}_{q(x_0)}\Big[D_{KL}\left(q(x_T \mid x_0) \,\|\, p(x_T)\right) - \mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0 \mid x_1)\right] \tag{5.1}$$

$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\left[D_{KL}\left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)\right]\Big] \tag{5.2}$$

For discrete diffusion, x_0 corresponds to a sequence of discrete variables, and the forward and reverse process are Markov processes that transition to other discrete sequences. The most common type of discrete diffusion is absorbing (*i.e.*, masked) diffusion, which gradually replaces elements with a masked token M in each step until the entire sequence has been masked. Empirically, this is the most effective type of discrete diffusion [6, 117, 162] and we focus on it in this paper. For simplicity, we consider a form of absorbing diffusion where a single unmasked token is randomly chosen to become masked in every step. Figure 5.2 shows a visualization of the absorbing forward process.

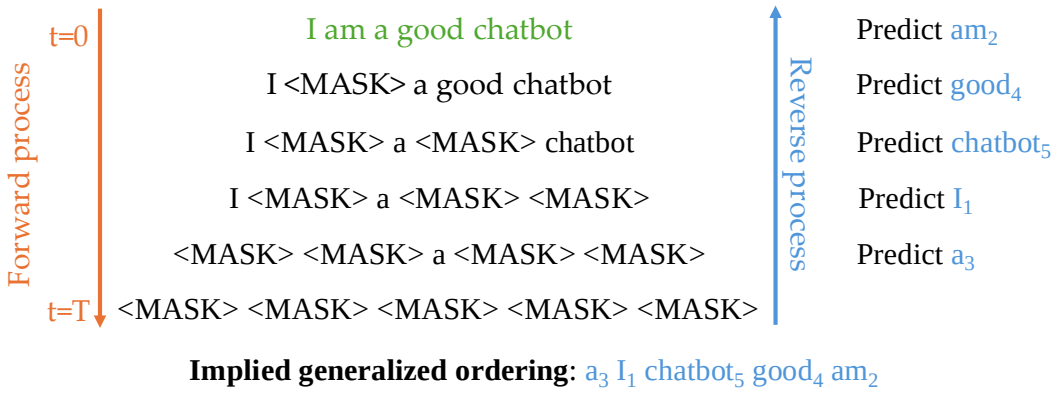


Figure 5.2: **Equivalence of absorbing diffusion and any-order autoregressive prediction.** Absorbing diffusion masks random tokens in the forward process, which requires the model to predict masked tokens in the reverse process. Intuitively, this reverse process is actually any-order autoregressive prediction: the model has to predict the next token in a generalized ordering, where the ordering is determined by the order that tokens were masked out during the forward masking process!

5.1.2 *Equivalence to a Generalized Autoregressive Ordering*

Absorbing diffusion may look complicated, but we can think intuitively about what is really happening. Figure 5.2 shows that the forward process masks out elements of x one-by-one, and the backward process roughly learns to fill the masked elements back in, one-by-one. If we arrange these tokens predicted during the backwards pass in order, it corresponds to predicting the next token in some permutation of the original sequence x. This is very interesting: it says that absorbing diffusion is an autoregressive model that predicts sequences permuted in any-order! Note that Hoogeboom et al. [82] has a similar insight. We formalize this connection below. First, we introduce notation for generalized autoregression:

Definition 5.1.1. (Generalized Ordering). A sequence $x = (z_1, \ldots, z_T)$ can be shuffled following permutation π to produce $x_\pi = (z_{\pi(1)}, \ldots, z_{\pi(T)})$. There are $T!$ unique orderings π, of which the standard left-to-right ordering $(1, \ldots, T)$ is just one, so we call x_π a generalized ordering of x.

Definition 5.1.2. (Generalized Autoregression). Generalized autoregression consists of factorizing $p_\theta(x)$ autoregressively along all possible orderings:

$$p_\theta(x) = \mathbb{E}_\pi \left[\sum_{t=1}^{T} p(z_{\pi(t)} \mid z_{\pi(1)}, \cdots, z_{\pi(t-1)}) \right] \tag{5.3}$$

By Jensen's inequality, this has variational lower bound:

$$\log p_\theta(x) \geqslant \sum_{t=1}^{T} \mathbb{E}_\pi \left[\log p(z_{\pi(t)} \mid z_{\pi(1)}, \cdots, z_{\pi(t-1)}) \right] \tag{5.4}$$

For simplicity, we will hereafter denote the sequence $z_{\pi(1)}, \cdots, z_{\pi(t-1)}$ as $z_{\pi(1:t-1)}$.

Now, we can formally show the equivalence between absorbing diffusion and any-order autoregressive prediction.

Theorem 5.1.3. *Absorbing discrete diffusion is equivalent to autoregression over a generalized ordering of the sequence. Both processes have the same variational lower bound for their log-likelihoods.*

First, note that $q(x_T \mid x_0) = p(x_T) = (M, \ldots, M)$, so the $D_{KL}(q(x_T \mid x_0) \,\|\, p(x_T))$ term in Equation (5.1) is 0. Next, note that the $\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0 \mid x_1)]$ term corresponds to predicting a single element, conditioned on the rest, which is equivalent to the $t = T$ term in the any-order variational bound (Equation (5.4)). Finally, note that $q(x_{t-1} \mid x_t, x_0)$ corresponds a uniform distribution over t possibilities, each of which is x_t with a single element unmasked into the correct original element in x_0. Treating this newly unmasked element as the next element in the permutation, $D_{KL}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))$ is then equivalent to $\mathbb{E}_\pi \left[\log p(z_{\pi(T-t+1)} \mid z_{\pi(1:T-t)}) \right]$. □

In some sense, absorbing diffusion is a generalization of standard autoregressive models, which only predict sequences in a single left-to-right order. This connection may be why absorbing diffusion models do relatively well in practice compared to other diffusion processes. In the next section, we translate the insights from this section into a more efficient absorbing diffusion model.

5.2 METHODS

Existing absorbing diffusion models [6, 117, 162] are implemented as Transformers [190] with bidirectional attention. These models take as input a partially masked sequence $(x_t^1, x_t^2, \cdots, x_t^T)$ and make predictions at each of the masked locations. Sampling with these models is computationally expensive compared to autoregressive models, which can utilize key-value (KV) caching. KV caching only works with causal attention masks, since the intermediate activations at earlier indices do not depend on the intermediate activations at later indices. Keys and values at each location can be computed once and then saved for use in future forward passes for later indices. Bidirectional attention allows all tokens to attend to all tokens, so it is incompatible with KV caching. A forward pass to sample new tokens now costs

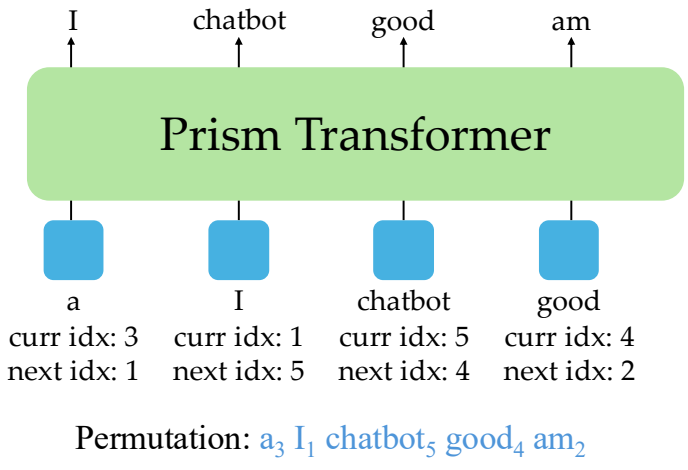


Figure 5.3: Prism is both an absorbing diffusion model and an any-order autoregressive model. It can predict the next token in any desired ordering of a sequence. At each location, Prism takes as input a token, the true index of that token, and the index of the next token to predict. Prism is a Transformer with one simple modification to the positional embedding: the Double RoPE presented in Section 5.2.2.

$O(T^2)$ operations, instead of $O(T)$ operations with KV caching. This is impractically expensive, especially as context lengths today begin to exceed 1 million tokens [114].

It is clear that adding some form of KV-caching is crucial for discrete diffusion to scale. One potential approach divides a sequence into blocks of fixed size b, and does absorbing diffusion within a block and autoregressive prediction across blocks [4]. This allows us to use KV caching for completed blocks but still has issues. Each forward pass costs $O(bT^2)$. If b is too small, then it loses the benefits of discrete diffusion, and if b too big, then each forward pass is too expensive.

How can we address this main limitation of discrete diffusion? We now show that we can develop a better discrete diffusion model using insights drawn in the previous section.

5.2.1 *Generalized Autoregressive Transformer*

We showed in Section 5.1 that absorbing diffusion is equivalent to any-order autoregressive prediction. Crucially, autoregressive prediction over a randomized ordering enables using a causal attention mask over the permuted sequence. This unlocks several new advantages. During training, the model can use all tokens as inputs and all tokens as outputs, which reduces the variance of the training objective and makes each step more efficient. During sampling, we can reduce the cost of a forward pass by employing KV-caching. Due to these advantages, we propose Prism, an absorbing

Positional embedding	Validation loss
Double Sinusoidal [137]	4.06
Double RoPE (ours)	**3.70**

Table 5.1: Our proposed Double RoPE positional embedding is more effective than previous positional embeddings for any-order prediction.

discrete diffusion model that is implemented as an any-order autoregressive model. We show a schematic of Prism's inputs and outputs in Figure 5.3.

5.2.2 *Double RoPE Positional Encoding*

In left-to-right autoregressive models, the index of the next token to predict is implicitly provided: it is the current index $+1$. However, when we do any-order prediction, the index of the next token to predict is no longer obvious, and we have to provide it to the model. σ-GPT handles this issue by adding two sinusoidal positional embeddings to each token [137]. One corresponding to the index of the token, and another corresponding to the index of the next token to predict. However, additive positional embeddings may not have enough inductive bias. Most language models use rotary positional embeddings (RoPE) [179], which rotates the queries and keys of each token based on its index m or n within the sequence. Specifically, RoPE uses a $\mathbb{R}^D \to \mathbb{R}^d$ rotation matrix R^d_m. Then, the attention scores are determined by $(R^d_m q_m)^\top (R^d_n k_n) = q_m^\top (R^{d\top}_m R^d_n) k_n$, which is only affected by the relative distance $m - n$ between the two indices m and n. We'd like to take advantage of similar inductive biases for Prism.

To make RoPE compatible with next-token prediction, we need to modify RoPE to use both the current and next index. We do this by splitting the query q in half, then rotating the first half based on the current index and the second half based on the next index. The key is rotated based on the current index only. This ensures that the attention score now depends on the relative distance between the query's "next index" and the query's "current index" as well as the standard relative distance between the current indices. We call this Double RoPE, since it uses twice as many indices into account. Table 5.1 shows that Double RoPE is better than the double sinusoidal approach, and Prism uses it for all following experiments.

Method	Sequential passes ($\propto$ latency) $\downarrow$	Normalized flops ($\propto 1/\text{throughput}$) $\downarrow$
Autoregressive	N	$C + N$
Bidirectional diffusion	F	$F \times (C + N)$
Prism (ours)	F	$C + (2N - F)$

Table 5.2: **Theoretical inference efficiency and speed.** We consider the task of generating N tokens from a prompt of length C. We display normalized flops, which define as dividing by the cost to predict 1 token with an autoregressive model of the same size. A standard autoregressive Tranformer requires N sequential forward passes to generate N tokens, whereas diffusion-based approaches can generate N tokens in F sequential forward passes, where F can be significantly less than N. However, previous bidirectional diffusion models cannot use KV caching, so each forward pass costs a full $C + N$ flops. Prism utilizes KV caching and roughly matches the flop count of autoregression while offering potentially massive generation speedups.

5.2.3 *Data orderings*

We showed in Theorem 5.1.3 that the standard absorbing diffusion process is equivalent to training over all $T!$ permutations. However, some orderings are unnecessarily hard. A fractal ordering like $(1, \frac{T}{2}, \frac{T}{4}, \frac{3T}{4}, \dots)$ requires the model to predict distant tokens, which can generally be difficult for the model [137]. Decoding using these orderings produces low-quality samples with issues like incoherence, so we should avoid sampling or spending training compute on such orderings. Luckily, we can train Prism over any distribution of data orderings we want and exclude those that are too hard.

Natural text can be predicted in any order at local scales (perhaps on the order of a single sentence) but may have some left-to-right structure at coarser scales. Thus, for language models we may want to permute indices within small blocks of size b, but still have blocks arranged in their original left-to-right ordering. More specifically, this blockwise permutation may transform a sequence of indices from $(1, \dots, b, b+1, \dots, 2b, \dots, T)$ into a new ordering $\text{shuffle}(1, \dots, b) + \text{shuffle}(b+1, \dots, 2b) + \cdots + \text{shuffle}(T-b+1, \dots, T)$. This corresponds to an any-order view of the discrete diffusion process in Arriola et al. [4]. This blockwise permutation strategy is a good fit for modeling text, but for reasoning tasks in Section 5.3.3 we still use all possible permutations.

5.2.4 *Accelerated sampling*

Many real-world applications are highly latency sensitive. For example, sequence models used in robotics need to sample quickly enough for the robot to respond to events in real time. Latency on typical hardware accelerators like GPUs and TPUs is mainly determined by the number of sequential forward passes needed to generate a sequence. Especially for autoregressive models, which typically generate a single token per sequential forward pass, latency is constrained by the time to load the weights into accelerator memory, which happens once per forward pass, rather than the time spent executing matrix or vector operations.

Absorbing discrete diffusion models can reduce sampling latency by unmasking multiple tokens per forward pass, reducing the total number of sequential forward passes to generate T tokens from T to F. Prism, as an instance of an absorbing discrete diffusion model, can also predict multiple tokens per forward pass. Interestingly, our implementation of Prism as a causally masked autoregressive model also enables accelerated *exact* sampling via speculative decoding. We highlight the advantages of each approach below.

APPROXIMATE SAMPLING VIA ABSORBING DIFFUSION VIEW To sample multiple tokens in a parallelized forward pass with Prism, we first fix an ordering π. Given C tokens in the existing context and k new tokens that we want to predict, we can sample each new token at location $\pi(C+i)$ independently via $p_\theta(z_{\pi(C+i)} \mid z_{\pi(1:C)})$. This can be done in a single parallel forward pass by appropriately changing the attention mask so that none of the k new tokens attend to each other. Note that we do not have to recompute anything for any of the C tokens in context, since we can use KV-caching. However, before the next forward pass, we do need to compute proper KVs at $k-1$ locations $\pi(C+2), \ldots, \pi(C+k)$. This is because Prism, when generating subsequent tokens, expects location $\pi(C+k)$ to have attended to $\pi(1:C+k-1)$ (and so on) when computing keys and values for that location. We do not have to recompute KVs for $\pi(C+1)$ because it already conditioned on $\pi(1:C)$, which is everything that it needs. The computational cost of one of these parallel forward passes is $O((2k-1)T)$, and generating N tokens in F forward passes costs $O((2N-F)T)$ compute. Note that $k = \frac{N}{F}$ must be less than the block size b (from Section 5.2.3) since the model does not know how to predict more than b tokens out. Overall, the absorbing diffusion view accelerates sampling by a factor of k tokens in exchange for incurring discretization error.

EXACT SAMPLING VIA GENERALIZED AUTOREGRESSIVE VIEW In general, causally masked autoregressive models can use a trick called speculative decoding to re-

duce sampling latency [104, 208]. Given C context tokens, speculative decoding typically uses a small autoregressive model to quickly generate a draft continuation $(z_{C+1}, \ldots, z_{C+K})$. Then, the large base model can verify this sequence in a single parallelized forward pass, performing rejection sampling to properly adjust for the difference in distribution. Speculative decoding has the desirable property of being exact: it samples exactly from the distribution of the base model. It reduces latency in exchange for extra computation spent on the small draft model and wasted flops spent on verifying tokens with the base model that follow any rejections. Speculative decoding only relies on the fact that these models are causally masked, so the base model can verify all generated tokens in a single forward pass.

Prism utilizes causal attention masks, so it can utilize speculative decoding, just as other any-order autoregressive models can [137]. Interestingly, Prism does not need a small draft model to quickly generate a continuation; it can self-generate k draft tokens in a single forward pass, as done above. Verification works similarly to standard speculative decoding, except we are guaranteed 2 acceptances instead of just 1. We are now guaranteed that the sample at location $\pi(C+1)$ is always accepted, since it was generated from the true model distribution $p_\theta(z_{\pi(C+1)} \mid z_{\pi(1:C)})$. Overall, self-speculative decoding can generate between 2 to $k+1$ tokens without incurring any discretization error. It is also not much more expensive than approximate sampling, since the verification computes the KVs with the appropriate conditioning structure. Thus, if we draft k tokens and accept j tokens on average per step, self-speculative decoding costs $O(2kT)$ per step, and requires $\frac{k}{j}$ as many steps as the approximate sampling approach. The main downside is that sometimes users are willing to pay the discretization error in exchange for more tokens generated per forward pass.

We compare Prism's latency and compute requirements against autoregressive and bidirectional discrete diffusion models in Table 5.2.

5.2.5 *Confidence-based decoding*

How should we choose the data ordering π at sampling time? It turns out that absorbing diffusion (and Prism by equivalence) will naively randomly choose the next location to predict. We formalize this below:

Theorem 5.2.1. *(Absorbing diffusion does not preferentially choose decoding locations). Given a partially unmasked sequence x_t, sampling from the true reverse process $p(x_{t-1} \mid x_t)$ for absorbing diffusion will decode a token at a location that is uniformly chosen from the currently masked indices.*

First, note that masking during the forward process happens uniformly across all locations. For a sequence of length T, $p(x_t^i = M) = t/T$ for all i, since there are t out of T total locations are masked. Then, we have:

$$p(x_{t-1}^i \neq M \mid x_t^i = M) = \frac{p(x_t^i = M \mid x_{t-1}^i \neq M)p(x_{t-1}^i \neq M)}{p(x_t^i = M)} \tag{5.5}$$

$$= \frac{1/(T-t+1)(1-(t-1)/T)}{t/T} \tag{5.6}$$

$$= 1/t \tag{5.7}$$

Thus, each of the t masked locations is equally likely to be unmasked in the reverse process. □

CHOOSING THE DECODING ORDER In many cases, there is indeed an ordering that makes the problem easier to solve [7, 212]. For instance, multi-digit addition is much simpler when adding from right-to-left, since carrying happens locally. There are also problems where the easiest ordering depends on the specific instance that is being solved. For example, Sudoku is easier when filling in the next most constrained square.

Unfortunately, previous approaches cannot easily take advantage of these "easy" orderings. Autoregressive models always predict from left-to-right, and absorbing diffusion models use a random ordering that is unlikely to be optimal. How do we fix this?

Even though we train on all orderings, we do not have to choose a random ordering at sampling time. Inspired by MaskGIT [26], we use the model's predictions to determine the next location to decode at. Specifically, we get the model's confidence in the most likely token at each masked location. Then, we simply choose the location with the highest confidence, and fix the most likely token at that location. We test this in Section 5.3.3 and find that this is especially effective for algorithmic problems like Sudoku or addition.

However, this does require us to make a prediction at each remaining masked location, which could be expensive. One solution we found is to add a head that predicts the confidence at each remaining location. After the initial pretraining phase, we use a small calibration set and compute the confidences at each masked location. Then, we freeze all original weights and train a linear head that uses the final representations (before the logits) to predict the confidences at each of the T locations. This allows us to perform confidence-based decoding without the $O(T^2)$ cost of obtaining true confidences. Instead, we can use the approximate confidences at just $O(T)$ cost. This empirically works well in practice.

Method	Parameters	Train FLOPs	Validation NLL
Autoregressive [162]	110M	3.46×10^{20}	2.86
SEDD [117]	110M	3.46×10^{20}	$\leqslant 3.18$
MDLM [162]	110M	3.46×10^{20}	$\leqslant 3.13$
BD3-LMs $b = 16$ [4]	110M	3.98×10^{20}	$\leqslant 3.10$
BD3-LMs $b = 8$ [4]	110M	3.98×10^{20}	$\leqslant 3.08$
BD3-LMs $b = 4$ [4]	110M	3.98×10^{20}	$\leqslant 3.03$
Prism $b = 256$	654M	5×10^{19}	$\leqslant 3.25$
Prism $b = 64$	654M	5×10^{19}	$\leqslant 3.19$
Prism $b = 16$	654M	5×10^{19}	$\leqslant 3.10$
Prism $b = 4$	654M	5×10^{19}	$\leqslant 2.92$

Table 5.3: **Validation loss on OpenWebText**. Prism achieves better likelihoods using around 1/7th as much training compute as the discrete diffusion baselines, though it still falls short of the autoregressive model that's trained for 7x as long.

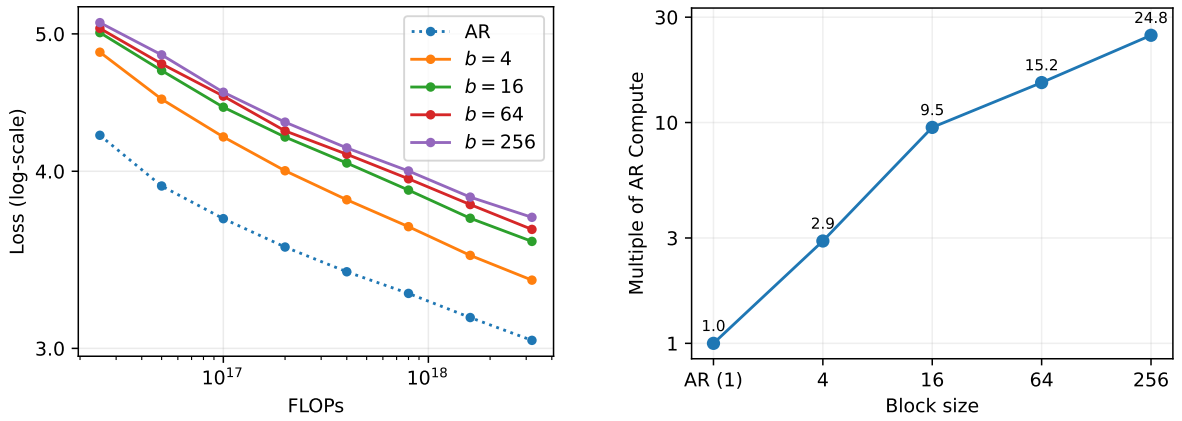
5.3 RESULTS

In this section, we test the effectiveness of the methods proposed in Section 5.2.

5.3.1 *Scaling Laws*

Prism, like almost all generative modeling algorithms, is consistent. With enough compute, data, and parameters, it will eventually model all conditional probabilities correctly and learn the true data distribution. Today, a more interesting question is how scalable Prism is. Can it efficiently improve as we increase the amount of training compute?

Kaplan et al. [91] showed that model log-likelihoods predictably follow a scaling law that accurately predicts model performance across many orders of magnitude of training compute. These scaling laws [81, 91] are crucial for understanding whether an approach will be practical at the large scales used in practice. Previous investigations into scaling laws for continuous diffusion language models [66] found that they require 64× as much training compute as autoregressive models, which is impractically large. In this section, we investigate whether Prism faces similar issues with scaling.



(a) Compute-optimal scaling laws

(b) Compute required to achieve a loss of 2.75.

Figure 5.4: Scaling law results for Prism on OpenWebText.

To measure a compute-optimal scaling law, we train models of varying sizes $\{N_1, \ldots, N_M\}$ for varying amounts of training FLOPs $\{C_1, \ldots, C_K\}$ on OpenWebText [61]. We consider a standard approximation of the compute as $C = 6ND$, where D is the number of training tokens and N excludes the embedding parameters. For a given budget C_i, we can find the optimal loss $L^*(C_i)$. We show this in Figure 5.4a, where each point is a separately trained model with different block size b that does best at a given budget. These scaling laws are roughly parallel on a log-log plot, which indicates that Prism requires a constant factor as much compute as an autoregressive model. If we fit the isoflop data using a power-law scaling law of form $L^*(C) = \alpha C^{\beta}$, we can extrapolate to predict how much compute is required to reach a target loss. Figure 5.4b shows that the required FLOPs (as a factor of autoregressive compute) increases as the block size b increases. This is expected, since a larger block size corresponds to the harder task of predicting more data orderings. However, this ratio is relatively reasonable for $b = 4$ and $b = 16$, and we may be willing to pay this in exchange for faster sampling (Section 5.3.2) and better reasoning (Section 5.3.3).

We can also fit scaling laws that determine the optimal model size $N^*(C)$ for a given compute budget. We find that the optimal ratio for Prism is roughly 19, which matches up with the "Chinchilla optimal" size from Hoffmann et al. [81]. Using these scaling laws, we train 654M parameter Prism models for 5×10^{19} FLOPs and compare against existing discrete diffusion models in Table 5.3. We find that Prism outperforms existing methods and comes close to autoregressive models.

5.3.2 *Accelerated sampling*

We test whether the sampling methods in Section 5.2.4 decrease latency without significantly hurting sample quality. Previous works have used a metric called generative perplexity [46, 67, 117, 162], which generates samples with a candidate model and evaluates their perplexity with a reference model. Unfortunately, this metric has several issues. It gives good scores to low-entropy output distributions, which is not necessarily indicative of a better generative model. It is also prone to sampling choices like temperature or precision which can unfairly affect results [211].

MEASURING DISCRETIZATION ERROR Instead, we directly measure the discretization error of multi-token prediction with respect to the validation loss. Recall that Prism's log-likelihood is bounded:

$$\mathbb{E}_{x\sim\mathcal{D}}[\log p_\theta(x)] \geqslant \sum_{t=1}^{T} \mathbb{E}_{x,\pi}\left[p_\theta(z_{\pi(t)} \mid z_{\pi(1:t-1)})\right] \tag{5.8}$$

$$= -\sum_{t=1}^{T} \mathcal{L}_t \tag{5.9}$$

where empirically $L_{t-1} > L_t$. When we sample more than one token at a time, this corresponds to a new factorization of $p(x)$, and we now have a new bound on the log-likelihood that reallocates more weight to the $\mathcal{L}_t$ that was used [82, 193]. In particular, we can write a sampling strategy as $\mathcal{S} = (n_1, \ldots, n_t, \ldots, n_T)$, where we sample n_t tokens in parallel with $t-1$ tokens in the context. Each n_t is a nonnegative integer with $\sum n_t = T$. Then our new discretized log-likelihood is:

$$\mathcal{L}_{\text{discretized}} = \sum_{t=1}^{T} n_t \mathcal{L}_t \tag{5.10}$$

We can sample a fixed number of tokens per forward pass, *e.g.*, we can sample 2 tokens per forward with $\mathcal{S} = (2, 0, 2, 0, \ldots)$. We can also solve a dynamic programming problem for the sampling strategy that uses a given number of forward passes and minimizes the discretized loss [82, 193].

RESULTS Figure 5.5 shows the optimal discretized loss for models with varying block size b. Interestingly, when doing naive multi-token prediction, small block sizes incur major error when predicting multiple tokens per step. This makes sense: we must indepedently predict tokens that are very close together (and are actually dependent on each other). Our largest block size of $b = 256$ incurs very little error, even when predicting up to 8-16 tokens per step.

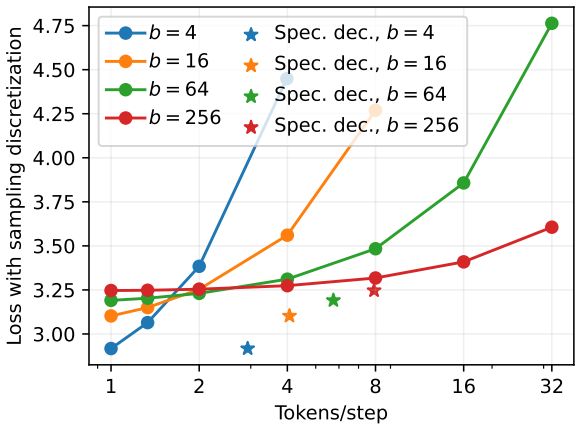


Figure 5.5: **Discretized loss vs the number of tokens decoded per sequential forward pass.** We trained 4 Prism models with 654M parameters each, with different block sizes b. We can directly sample k tokens independently on average from each model and evaluate the validation loss under this factorization of the sequence (solid line). This corresponds to treating Prism as a discrete diffusion model. Larger block sizes b enable sampling more tokens per step without incurring large discretization error. We can also treat Prism as an any-order autoregressive model and perform self-speculative decoding (stars). We see that the Pareto frontier of loss vs tokens/step for self-speculation is better than the frontier for naive multi-token prediction, though it is less flexible.

We also show how many tokens are sampled per step, on average, with our self-speculative decoding approach. Recall that self-speculative decoding incurs no discretization error, as it can reject samples where the tokens are inconsistent with each other. Figure 5.5 shows that this forms the entire Pareto frontier and is far more effective than the naive multi-token prediction strategy that absorbing diffusion models typically employ. Self-speculative decoding is a major advantage of Prism, as standard absorbing diffusion models cannot verify draft tokens in parallel and cannot use speculative decoding. Note that it does cost slightly more, since compute can be wasted verifying tokens after a rejection already occurs. There are also significant avenues for improvement here, as there is a large body of speculative decoding techniques for autoregressive models that can be directly applied here [34, 199]. However, one must be careful: Pannatier, Courdier, and Fleuret [137] propose running speculative decoding for any-order models over multiple permutations and picking the order with the longest acceptance, but this is not a valid rejection sampling strategy and distorts the sampling distribution.

Model	Decoding strategy	Sudoku (hard)	Path-star	Addition
Autoregressive	left to right	-	20.1	0.0
Prism	left to right	2.0	19.9	0.0
	right to left	1.3	100.0	100.0
	random	1.9	31.1	0.0
	confidence-based	71.7	100.0	100.0
	confidence-based (with head)	70.1	100.0	100.0

Table 5.4: **Improved reasoning abilities on algorithmic tasks**. Using confidence-based decoding to predict the next location to sample at drastically improves Prism's performance on all tasks.

5.3.3 *Reasoning*

In this section, we test whether our confidence-based method for dynamically determining decoding order (Section 5.2.3) is effective on algorithmic reasoning tasks.

BENCHMARKS We evaluate our models on three problem-solving benchmarks:

- Sudoku (hard) [136]: We train models on 50000 randomly generated Sudoku puzzles and evaluate their exact-solve accuracy on a hard test set with 17 provided clues. We do not evaluate autoregressive models on this task, as it is difficult to incorporate the clues into their context.
- Path-star graph [7]: This task corresponds to a graph with a single central node at the center with degree 5, and paths of length 5 that branch off the center. Given the adjacency list of this graph and a target node that lies at the end of one of these branches, the model must output an ordered list of intermediate nodes on the path to that target. This task is hard when predicted from left-to-right, because determining the first node on a branch requires multi-hop reasoning. Right-to-left prediction is much easier since it only requires one-step reads into the adjacency list, starting with the target node.
- Multi-digit addition: The model must predict the sum of two 20-digit numbers. The numbers and outputs are presented with the most significant digit on the left. This is difficult to solve from left-to-right because it requires accounting for multiple steps of potential carries.

RESULTS We show results in Table 5.4. Autoregressive models do poorly on both Path-star and addition. Instead of learning to do proper many-hop reasoning, they overfit to the training set. Prism, which trains over randomized orderings of the data, can fare much better. Using the confidence-based decoding method significantly improves performance beyond the default random ordering. Using a linear head to predict confidences for this decoding method is almost as effective, while greatly reducing sampling compute by eliminating the need to do a forward pass for every location.

5.4 DISCUSSION

Overall, we show a useful equivalence between absorbing discrete diffusion models and any-order autoregressive models. Prism utilizes this insight for more efficient training than alternative absorbing diffusion algorithms. Training on all orders also enables lower-latency sampling than autoregressive models and more efficient sampling than other absorbing diffusion models. Finally, any-order training also enables strong performance on reasoning tasks, especially when using our confidence-based decoding method to choose which location to sample at.

Prism does have several limitations. It costs more to pretrain than autoregressive models. While it can dynamically do fewer forward passes to reduce sampling latency, it cannot "think" for more than T steps and produce better samples. Handling the EOS token is hard, and the absorbing diffusion framework makes it difficult for the model to undo mistakes during sampling. We hope future work explores solutions to these problems.

6

CONCLUSION

Machine learning has vast potential to improve our lives. Yet, we must first figure out how to reliably improve generalization if we want robots to help us in our daily lives or automated scientists to discover new medicines.

In this thesis, we explored how to improve generalization by structuring learning problems to be *harder*. Chapter 2 proposed Internet Explorer, a method that autonomously queries the Internet for hard data that induces high self-supervised loss for our learner. Afterwards, we focused on how to extract more learning signal from a fixed amount of data. Chapters 3 and 4 revisit the paradigm of generative classifiers, which have the richer task of modeling $\log p(x \mid y)$. We develop techniques for using today's modern generative modeling algorithms, diffusion models and autoregressive models, as image and text classifiers. We show that generative classifiers are the first method to have qualitatively better scaling for its out-of-distribution accuracy ("effective robustness"). Generative classifiers often outperform discriminative classifiers in-distribution as well. Finally, Chapter 5 proposed a new sequence modeling algorithm called Prism. Prism is both an absorbing diffusion model and an any-order autoregressive model. Prism requires more training compute compared to autoregressive models, as it has the harder task of learning all orderings, but it has several advantages at sampling time. It can dynamically decide which order to sample in, drastically improving generalization on algorithmic reasoning tasks. It can also generate multiple tokens at once, so it can sample sequences with significantly lower latencies than a left-to-right autoregressive model can.

6.1 FRONTIERS

We still have a long way to go to match the level of generalization that we know is possible. Animals like horses can begin walking just an hour after birth; humans see far fewer images and hear far fewer words [17] than our large, pretrained vision-language models, yet generalize far more reliably. Several promising directions for future research include:

1. *Cross-modal generalization*: Most useful text data on the Internet is manually generated by humans, so we are rapidly running out of new tokens. However, we still have a vast amount of video data, which encodes so much temporal information about the world. Visual data is also often complementary to what is explicitly written in text [48, 80, 99], so it likely contains a huge amount of disjoint knowledge that can only be learned through pixels. Numerous works [113, 206] have tried training jointly on text, image, and video. However, we still haven't seen any improved text generalization from training on video or images. One key ingredient may simply be model size: even text-only models require tens of billions of parameters before seeing significant abstract (*e.g.*, cross-lingual) generalization [65]. Beyond scaling, I believe that fully utilizing this visual data would require significant autonomous data curation, as we did with Internet Explorer in Chapter 2. I did some preliminary web scraping, and I estimate that YouTube has at least 2.5 billion videos and a total of at least 43 trillion frames. Most of this data is likely useless, such as security camera videos or nature livestreams with almost no movement. Applying scaled-up versions of Internet Explorer or other data curation methods [1, 50, 102, 176] is necessarily to filter for hard, interesting, and learnable videos. Finally, we may need to use the right problem formulation and find good recipes for video tokenization, captioning, conditioning, and model architecture.

2. *Unifying generative classifiers with language models*: Language models are beginning to subsume all other tasks, due to their flexibility, large amount of pretraining data, and ease of prompting. We would really like to take advantage of the generalization benefits of generative classifiers within a language modeling framework. However, it's currently not clear how to efficiently do so. Translating the conditional likelihood $p_\theta(x \mid y)$ into its equivalent in the autoregressive LLM framework, we need to compute $p_\theta(x_{1:t} \mid x_{t+1})$ once per generated token. This appears infeasibly expensive, as it would naively require $O(TV)$ compute, where V is the size of the vocabulary. Perhaps parameterizing p_θ cleverly could reduce the compute required per token sampled, while maintaining the generalization benefits.

3. *Dynamic compute*: While Prism and other discrete diffusion models can use fewer forward passes in exchange for faster samples, they don't have an option to *increase* the number of forward passes past the sequence length T in exchange for higher quality samples. This ability could be useful if we encounter particularly difficult test scenarios that we need to "think harder" on. Current approaches mainly approach this by spending more tokens in an intermediate scratchpad [132, 194], but it's unclear how to spend more compute per token.

One approach is to use do a variable number of forward passes with a recurrent model before computing the logits [167]. In the limit, this resembles finding a fixed point, and deep equilibrium models may be a promising avenue to better generalization with more test-time compute [3, 8].

Part IV

APPENDIX

A

INTERNET EXPLORER: TARGETED REPRESENTATION LEARNING ON THE OPEN WEB

A.1 LEARNING FROM OTHER SOURCES OF DATA

Google Images is an exceptionally useful data source for Internet Explorer. It offers access to a large portion of the Internet's images, and it ranks images using weak supervision from the image caption, surrounding text, click rates, image features, incoming and outgoing hyperlinks, and other signals. This extra supervision is helpful and should be utilized. Nonetheless, we show that Internet Explorer is agnostic to the choice of text-to-image search engine and can still rapidly improve even when the data source is much noisier.

To test Internet Explorer in the most minimal setting, we build a custom search engine that finds images solely using their accompanying text—without using any pre-trained visual features whatsoever. We use the LAION-5B dataset [166], which consists of >5B noisy image-caption pairs. We filter the dataset to only include images of at least 512^2 pixels with English captions. This leaves us with about 600M text-image pairs. To find image results for a query, we find the 100 captions closest to the query in text representation space, then return the associated images. We

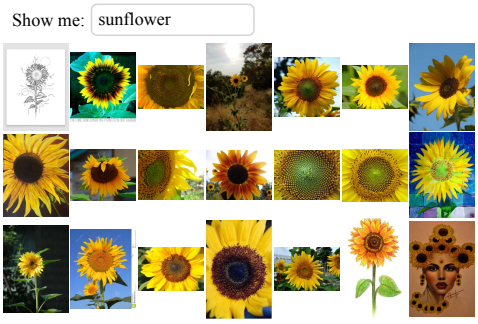


Figure A.1: **Our custom LAION-5B search engine.** We build a custom text-to-image search engine that finds images within the LAION-5B dataset by doing nearest neighbor search in text embedding space. This uses no image features whatsoever.

use a pre-trained text embedding model [151] to compute 384-dimensional text embeddings for each caption. Then, we use Faiss [88] to compute a fast, approximate nearest-neighbors lookup index. Querying our custom search engine finds 100 image results in less than a second. Figure A.1 shows that our search engine is reasonably accurate, even without using any image features.

We also test Flickr's photo search API as another text-to-image search engine, in addition to Google Images and LAION. Figure A.3 shows that each data source has its own tendencies. For the "spaghetti bolognese" query, Google Images is biased [30, 123] towards brightly-lit, photogenic images that typically come from food blogs. Flickr mainly consists of amateur home photos, so it returns a messier variety of images that perhaps better capture the real world. LAION images come from web crawling, without any ranking, so they additionally contain many graphics with text overlays. The same image can also frequently show up in the LAION results multiple times, as a result of being posted on multiple separate pages.

Figure A.2 and Table 2.2 (main paper) show that Internet Explorer still improves over time, even when the data comes from LAION or Flickr. Internet Explorer tends to perform better with Flickr than with LAION, which makes sense. Flickr indexes far more images, as our custom LAION search engine only uses 600M images, so it can return more of the useful photos that Internet Explorer queries for. Flickr is also slightly better at understanding descriptors, although both Flickr and LAION tend to be thrown off by specific or odd descriptors. Nevertheless, Internet Explorer significantly improves the starting model in less than a day of searching and training even with noisy search results and no hyperparameter tuning. Overall, these results prove that Internet Explorer can effectively utilize any window into the Internet's vast ocean of image data.

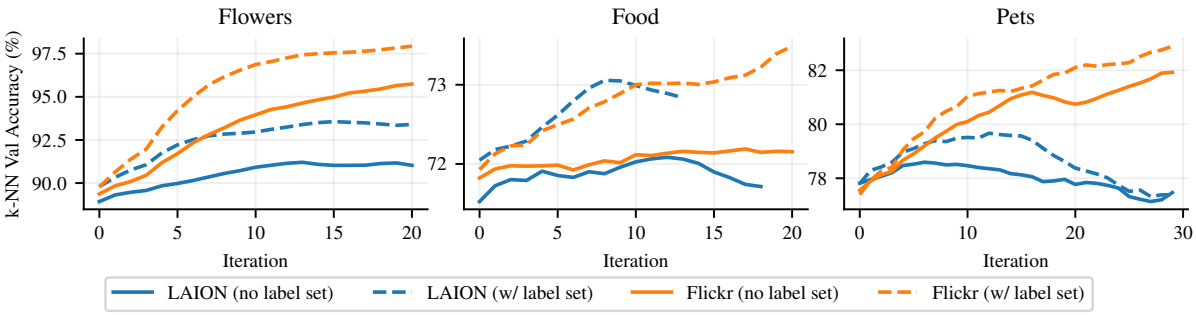


Figure A.2: **Learning from Flickr and LAION-5B.** Even with the noisy search results returned by Flickr and LAION, Internet Explorer still continuously improves performance.

Food101 dataset: “Spaghetti Bolognese”

Google Images: “Spaghetti Bolognese”

Flickr: “Spaghetti Bolognese”

LAION-5B: “Spaghetti Bolognese”

Figure A.3: **Comparison of different search engines.** We show images for the “spaghetti bolognese” class in the Food101 dataset, as well as 20 search results for “spaghetti bolognese” from Google Images, Flickr, and LAION5B. Google images are typically well-lit, aesthetic food blog pictures. In comparison, Flickr images are messier, darker, and capture a wider variety of real-world conditions. LAION-5B images lie somewhere in the middle, but contain text overlays much more frequently. Duplicate image results are also common.

	Birdsnap	Flowers	Food	Pets	VOC2007
Target test set size	1849	6142	25246	3663	4952
No exploration					
Target training set overlap	1 (0.05%)	5 (0.01%)	34 (0.13%)	21 (0.57%)	0 (0.00%)
Internet Explorer					
Ours++ (no label set)	28 (+1.46%)	11 (+0.01%)	35 (+0.00%)	26 (+0.14%)	1 (+0.02%)
Ours++ (with label set)	57 (+3.03%)	27 (+0.36%)	35 (+0.00%)	43 (+0.60%)	1 (+0.02%)

Table A.1: **Number of leaked test set images**. We use image hashing to compute the fraction of test images present in the set of images downloaded by Internet Explorer. Surprisingly, the training/validation sets of these datasets already leak a small fraction of the test sets—Pets is the most egregious, with 0.57% test leakage. For each dataset, we show the test set size, the number of leaked test images, and the percentage of the test set that this represents in blue. For each version of our method, we show the total number of leaked images that the model had access to, and the percentage increase this represents over the training set's leakage in blue. Leakage numbers for our methods include this train-test leakage, since our methods also train on the target dataset's training set. Internet Explorer only finds a tiny fraction of test set images online, and it only uses them for self-supervised training, so there is no *label leakage*. Internet Explorer's large increase in accuracy cannot be explained by test set leakage, so its performance gains must come through better feature learning and generalization.

A.2 ARE WE FINDING THE ENTIRE TEST SET ONLINE?

One may be concerned that Internet Explorer improves performance mainly by finding a significant portion of the test set images online. We address this concern by checking how much test data Internet Explorer has downloaded. We use difference hashing (dHash) [22] to compute hashes for the target dataset's training set, its test set, and the $\approx 10^6$ images that Internet Explorer has downloaded. We compare hashes to determine how many test images were leaked, and we report the number of collisions in Table A.1. Across all five datasets, Internet Explorer finds very few test images. On Birdsnap, Internet Explorer finds 56 additional test set images that were not leaked in the training set, which is roughly 3% of the test set. On the other datasets, the amount leaked ranges from 0.003% to 0.6% of the test set. Additionally, we only perform image-based self-supervised training on downloaded images, so it is much harder for our model to cheat with the leaked images. Overall, given that Internet Explorer outperforms its starting checkpoint by between 5 to 30 percentage points, we conclude that its performance cannot be explained by cheating.

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure A.4: **Top-10 most similar online images to Pets101**

In fact, we view it as a positive that Internet Explorer finds some test set images, because it serves as confirmation that it is learning to search for relevant images—and the most relevant images possible would be those from the dataset itself! But beyond test set images, Internet Explorer finds a lot of internet images that are very relevant to the dataset. We visualize the top-10 most similar downloaded images for 5 randomly selected test set images from multiple datasets in Figures A.4 to A.8. We use CLIP ViT-L/14 to compute the representations of the test set images, as well as the downloaded images. We then find the top-10 most similar online images given a test set image (from the downloaded images using Ours++ (with label set)). We see that Internet Explorer finds several images that are very similar but not identical to the test set images.

A.3 METHOD DETAILS

A.3.1 *WordNet Lemmas*

We draw our concepts from the WordNet hierarchy [124], which consists of 146,347 noun lemmas. For reference, here are 32 randomly sampled concepts:

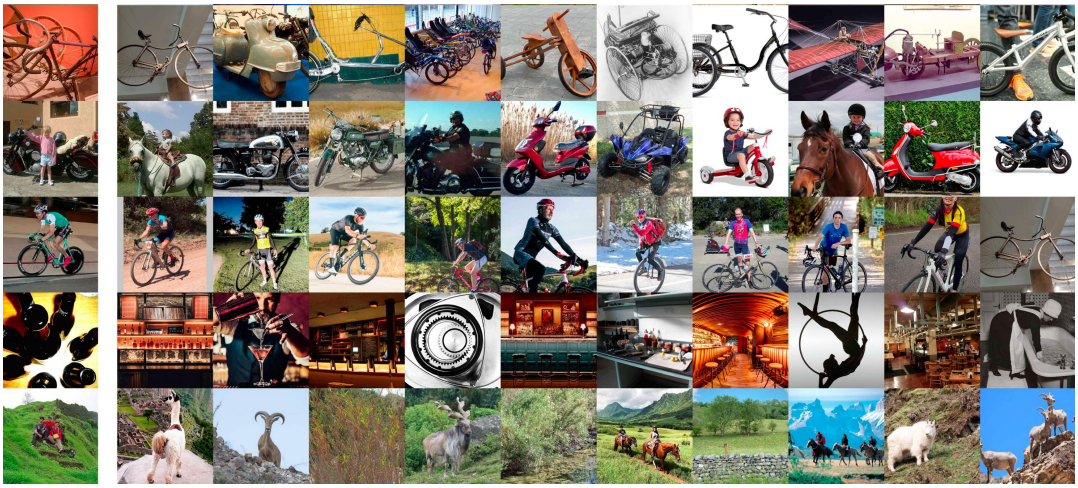
```
 "resolution", "lodgment", "phycobilin", "acidosis", "widening",
"human face", "family Crassulaceae", "sail", "Ipomoea imperialis",
"Davis", "prothrombin", "cease", "marsh clematis", "major power",
"chump change", "madcap", "junky", "pere david's deer", "make-up",
"genus Rumex", "gape", "Brachychiton populneus", "bell morel", "wain",
```

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure A.5: **Top-10 most similar online images to Food101**

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure A.6: **Top-10 most similar online images to Flowers102**



Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure A.7: **Top-10 most similar online images to PASCAL VOC2007**

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure A.8: **Top-10 most similar online images to IN100**

```
"friendly", "Principe", "bottle green", "glycerol trimargarate",
"water-shield", "San Joaquin River", "woodsman", "pin".
```

A.3.2 *GPT-J Descriptor Prompting*

We use GPT-J-6B [192], a free, open-source autoregressive language model, to generate useful descriptors for a given concept. We use the following prompt template:

```
"What are some words that describe the quality of ‘{concept}’?

The {concept} is frail.

The {concept} is red.

The {concept} is humongous.

The {concept} is tall.

The {concept} is"
```

We sample completions with a temperature of 0.9 and a max length of 100 tokens. We truncate the completion after the first comma, period, underscore, or newline character (including the special character). If the truncated completion is degenerate and contains a duplicate of the concept, we resample another completion. After successfully sampling a descriptor, we prepend it to the concept and use the resulting phrase as our search query.

For reference, here are 32 randomly sampled descriptors for "labrador retriever":

```
 "a good-looking dog", "very gentle", "a", "brown", "lovable", "a
strong runner", "a male or a female", "sturdy", "agile", "a strong",
"beautiful", "a male", "kind", "long-haired", "a male or a female",
"a good-looking dog", "gentle", "medium", "loyal", "very gentle",
"blue-eyed", "sturdy", "blue-eyed", "a retriever", "kind", "loyal",
"large", "brown", "good-natured", "gentle", "large", "small".
```

A.3.3 *Concept Vocabulary Size*

As stated in Section 2.1.2, our vocabulary comprises the 146,347 noun lemmas in the WordNet hierarchy. Thus, in all our experiments, Internet Explorer only searches for WordNet terms (plus the class names, if we have knowledge of the label set). We found that this worked quite well for these standard benchmarks. Note that expanding the vocabulary (e.g., adding technical terms relevant to a specific topic)

can easily be done by adding those terms to the list of possible concepts. One easy extension would be to add page titles and frequent unigrams and bigrams from Wikipedia, as was done to generate the CLIP training set [146]. Doing so would expand our vocabulary to roughly 500,000 total concepts.

A.3.4 *Query Model Details*

TEMPERATURE FOR CONCEPT DISTRIBUTION After estimating scores $r(c_i)$ for each concept c_i, we do a temperature-scaled softmax, followed by the tiering operation described in Section 2.6. We compute the temperature τ such that

$$\text{SMR} = \frac{\max_i r(c_i) - \min_i r(c_i)}{\tau} \tag{A.1}$$

where the "softmax range" $\text{SMR} \in \mathbb{R}$ is the desired gap between the largest and smallest scores after temperature scaling. After the softmax $p(c_i) \propto \exp(r(c_i)/\tau)$, the softmax range determines the likelihood ratio of most likely concept to least likely concept:

$$\frac{\max_i p(c_i)}{\min_i p(c_i)} = \frac{\max_i \exp(r(c_i)/\tau)}{\min_i \exp(r(c_i)/\tau)} \tag{A.2}$$

$$= \exp\left(\frac{\max_i r(c_i) - \min_i r(c_i)}{\tau}\right) \tag{A.3}$$

$$= \exp(\text{SMR}) \tag{A.4}$$

Thus, SMR is an easy way to specify the relative likelihood of the highest and lowest scoring concepts and achieve a desired exploration-exploitation balance.

LABEL SET-GUIDED VOCABULARY To reduce our search space in the label set-guided setting, in which we know the English names of the classes a priori, we generate a subset of the WordNet vocabulary that contains only the top-10% most semantically-relevant concepts to each target dataset. We use a pre-trained text embedding model [151] to generate 384-dimensional embeddings for each concept in WordNet, using the same template described in Section 2.5 of the main paper:

```
{lemma} ({hypernym}): {definition}.
```

To generate a similar embedding for concepts in target datasets, we use the summary from Wikipedia in place of the definition and the "category" of the target dataset (shown in Table A.2) in place of the hypernym:

```
{label} ({category}): {summary}.
```

Dataset	Category
Oxford Flowers102	Flower
Oxford IIIT Pets	Pet
Food101	Food
Birdsnap	Bird
VOC2007	Object

Table A.2: **Target Dataset "Category"**.

After generating the embeddings for each concept in the target dataset, we find the k-NN distance for each WordNet concept to the target dataset embeddings, where k is chosen to be 1/3 the size of the class label set. We then rank the concepts in WordNet by the distance and take the closest 10% of terms as our subset. This subset is used for all methods in the label set-guided setting, including the random exploration methods.

A.3.5 *Training Details*

In each iteration, we download roughly 25k candidate images, since we download up to 100 images for each of the 256 queries. Given this set $\mathcal{C}$ of candidate images, we sample PCR $\times$ $|\mathcal{C}|$ images from the union of the replay buffer $\mathcal{B}$ and the target dataset training images $\mathcal{D}$. PCR (past data to candidate data ratio) is a scalar value that determines how much old data vs new data to train on at every iteration. We set PCR $= 2$ for all experiments. We perform 10 epochs of training over the union of the new candidate data and the sampled replay buffer and target dataset images.

A.3.6 *Hyperparameters*

Table A.3 shows our hyperparameter values, which are shared across datasets. We perform minimal hyperparameter tuning and copy most of the values from the MoCo-v3 [33] ResNet-50 configuration. Our code has been released at `https://github.com/internet-explorer-ssl/internet-explorer`, which we hope will clarify any remaining implementation details and make it easy for the community to reproduce and build on our work.

Hyperparameter	Value
Architecture	Resnet-50 [72]
Optimizer	LARS [201]
Batch size	224
Learning rate	$0.8 \times \frac{224}{256}$
Learning rate schedule	constant
MoCo momentum	0.9985
RandomResizedCrop min crop area	0.2
Queries per iteration	256
Requested images per query	100
Min images per query	10
Softmax range (SMR)	3
PCR	2
Epochs per iteration	10

Table A.3: **Internet Explorer hyperparameters**.

A.3.7 *Image Licenses*

Internet Explorer uses images that were indexed by a web crawler (Google Images and LAION) or uploaded to Flickr. The images and their rights belong to their respective owners; we use, download, and train on them under fair use guidelines for research.

A.3.8 *Domain Dataset Descriptor Details*

When targeting a niche domain dataset—in which a practitioner almost surely has useful a priori knowledge to impart—it is simple to modify Internet Explorer to accelerate learning. Rather than using GPT to help provide variety to our queries for a concept, we can use leverage our practitioner's domain knowledge to help hone our search from the start.

This amounts to defining a list of "descriptors" that help return relevant results for arbitrary queries. For example, the below list of 16 descriptors was selected for the FMoW satellite dataset to help return satellite imagery when prepended to concepts (e. g., "tennis court") instead of their more canonical views. This list was hand-

selected through trial & error using a variety of randomly selected concepts. Note that this static list replaces the GPT-J generated descriptors for this dataset.

FMoW-WILDS Descriptors:

```
"a centered satellite photo of", "a satellite photo of", "a google
earth photo of", "satellite view of", "high resolution satellite",
"high resolution satellite imagery of", "aerial satellite", "aerial
satellite view", "aerial satellite view of", "satellite imagery,
centered photo of", "satellite imagery, photo of", "military highest
resolution satellite imagery of", "NASA imagery of", "geo high resolution
satellite", "land cover satellite image of", "european satellite
close up aerial image of", "super high resolution highest resolution
satellite imagery"
```

A.4 PROOF OF LEMMA 2.1.1

Here, we prove Lemma 2.1.1 from Section 2.1.6, which we repeat below:

Lemma 2.1.1. *Let T_{base} be the expected time to identify every relevant concept without the GPR, and T_{GPR} be the expected time when exploiting the additional knowledge from the GPR. Then, $T_{base} = nH_{c \cdot s}$, $T_{GPR} = \frac{nH_c}{s}$, and the speedup from GPR is $\frac{T_{base}}{T_{GPR}} \approx s \log s$.*

Proof. This problem is a variant of the coupon collector problem. Let's first compute T_{base} as the sum of expected times t_i to identify the next relevant concept.

$$T_{base} = \sum_{i=1}^{cs} t_i \tag{A.5}$$

$$= \sum_{i=1}^{cs} \frac{1}{p_i} \tag{A.6}$$

$$= \sum_{i=1}^{cs} \frac{n}{cs+1-i} \tag{A.7}$$

$$= n \sum_{i=1}^{cs} \frac{1}{cs+1-i} \tag{A.8}$$

$$= nH_{cs} \tag{A.9}$$

where H_{cs} is the csth harmonic number. Similarly, we can compute T_{GPR} as the sum of expected times t_i to identify the next relevant cluster.

$$T_{GPR} = \sum_{i=1}^{c} t_i \tag{A.10}$$

$$= \sum_{i=1}^{c} \frac{1}{p_i} \tag{A.11}$$

$$= \sum_{i=1}^{c} \frac{n}{s(c+1-i)} \tag{A.12}$$

$$= \frac{n}{s} \sum_{i=1}^{c} \frac{1}{c+1-i} \tag{A.13}$$

$$= \frac{nH_c}{s} \tag{A.14}$$

The speedup is then $\frac{T_{base}}{T_{GPR}} = s\frac{H_{cs}}{H_c} \approx s \log s$. □

We find that in practical settings (e.g., the Pets example analyzed in Figure 2.5), we can accurately predict how many samples are required to discover all useful concepts. If the vocabulary size is $n \approx 150{,}000$, the number of clusters is about $c = 2$ (one for cats and one for dogs), and the size of each cluster is about 150, then $T_{GPR} = 1500$, which roughly matches the 9 iterations $\times 256$ queries/iteration $= 1792$ queries it took to discover both cats and dogs in the Pets dataset.

A.5 PROGRESSION OF DOWNLOADED IMAGES

Just as Figure 2.4 in the main paper showed how Internet Explorer progressively discovers useful data when targeting the Pets dataset, Figures A.9 to A.12 show the progression of downloaded images when targeting Birdsnap, Flowers, Food, and VOC respectively. Note that this analysis is in the self-supervised setting, where Internet Explorer has no knowledge of the label set. Thus, it is quite surprising that Internet Explorer is able to identify relevant images in so few iterations.

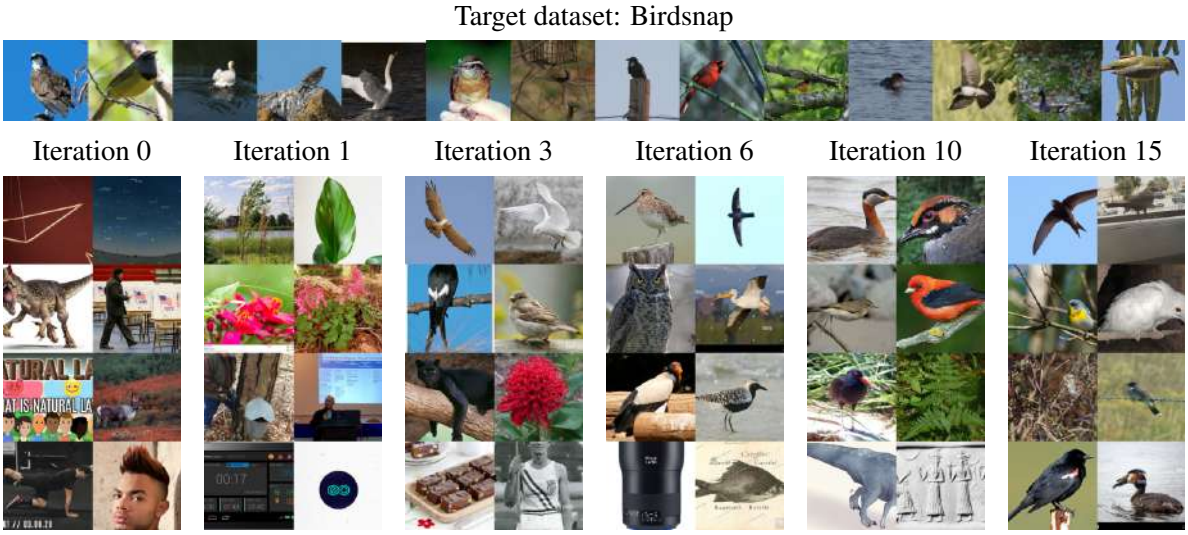


Figure A.9: **Progression of downloaded Birdsnap images.** This corresponds to Ours++ without using label set information.

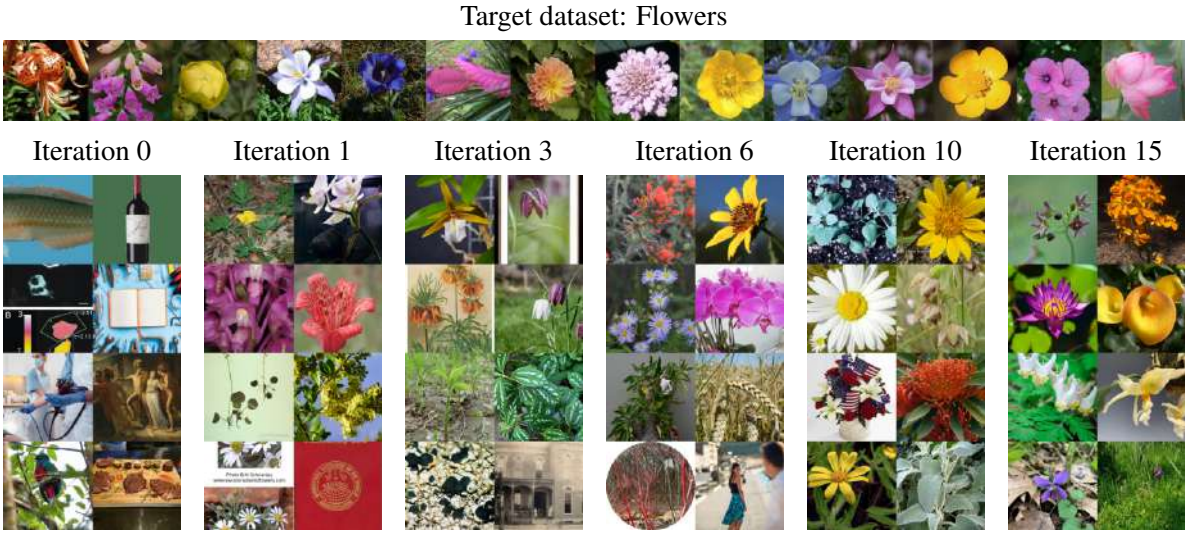


Figure A.10: **Progression of downloaded Flowers images.** This corresponds to Ours++ without using label set information.

Figure A.11: **Progression of downloaded Food images.** This corresponds to Ours++ without using label set information.

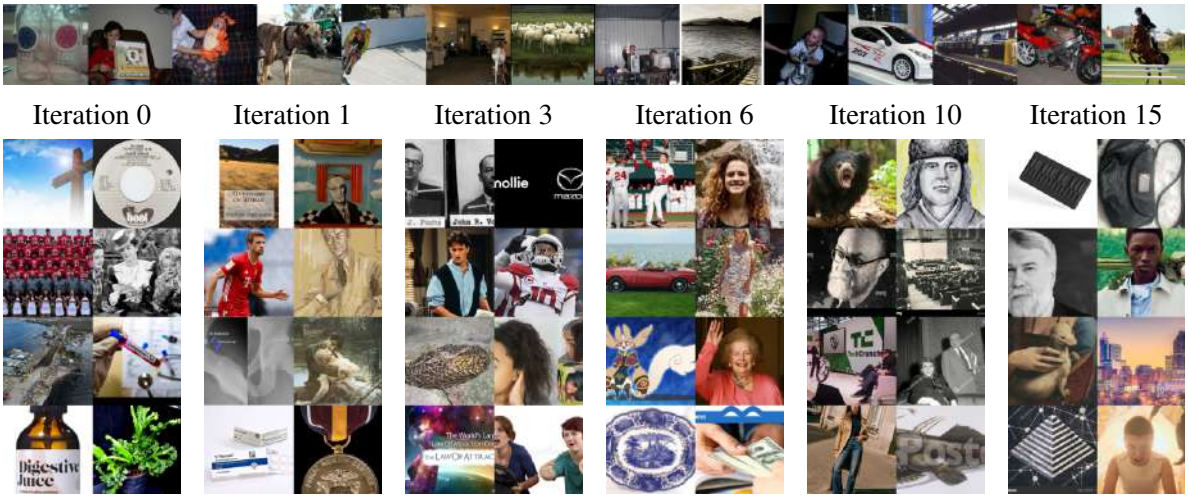


Figure A.12: **Progression of downloaded VOC2007 images.** This corresponds to Ours++ without using label set information.

B

YOUR DIFFUSION MODEL IS SECRETLY A ZERO-SHOT CLASSIFIER

B.1 EFFICIENT DIFFUSION CLASSIFIER ALGORITHM

Though Diffusion Classifier works straightforwardly with the procedure described in Algorithm 2, we are interested in speeding up inference as described in Section 3.2.2. Algorithm 3 shows the efficient Diffusion Classifier procedure that adaptively chooses which classes to continue evaluating. Table B.1 shows the evaluation strategy used for each zero-shot dataset. We hand-picked the strategies based on the number of classes in each dataset. Further gains in accuracy may be possible with more evaluations.

Dataset	Prompts kept per stage	Evaluations per stage	Avg. evaluations per class	Total evaluations
Food101	20 10 5 1	20 50 100 500	50.7	5120
CIFAR10	5 1	50 500	275	2750
Aircraft	20 10 5 1	20 50 100 500	51	5100
Pets	5 1	25 250	51	1890
Flowers102	20 10 5 1	20 50 100 500	50.4	5140
STL10	5 1	100 500	300	3000
ImageNet	500 50 10 1	50 100 500 1000	100	100000
ObjectNet	25 10 5 1	50 100 500 1000	118.6	13400

Table B.1: Adaptive evaluation strategy for each zero-shot dataset.

B.2 INFERENCE COSTS AND HYBRID CLASSIFICATION APPROACH

Table B.2 shows the inference time of Diffusion Classifier when using the efficient Diffusion Classifier algorithm (Algorithm 3). Classifying a single image takes anywhere between 18 seconds (Pets) to 1000 seconds (ImageNet). The issue with ImageNet is that Diffusion Classifier inference time still approximately scales linearly

Algorithm 3 `Diffusion Classifier (Adaptive)`

Input: test image $\mathbf{x}$, conditioning inputs $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^n$ (e.g., text embeddings or class indices), number of stages N_{stages}, list `KeepList` of number of $\mathbf{c}_i$ to keep after each stage, list `TrialList` of number of trials done by each stage
Initialize $\texttt{Errors}[\mathbf{c}_i] = \text{list}()$ for each $\mathbf{c}_i$
Initialize `PrevTrials = 0` // How many times we've tried each remaining element of $\mathcal{C}$ so far
for stage $i = 1, \ldots, N_{\text{stages}}$ **do**
  for trial $j = 1, \ldots, \texttt{TrialList}[i] - \texttt{PrevTrials}$ **do**
    Sample $t \sim [1, 1000]$
    Sample $\epsilon \sim \mathcal{N}(0, I)$
    $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1 - \bar{\alpha}_t}\epsilon$
    for conditioning $\mathbf{c}_k \in \mathcal{C}$ **do**
      $\texttt{Errors}[\mathbf{c}_k].\texttt{append}(\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_k)\|^2)$
    end for
  end for
  $\mathcal{C} \leftarrow \underset{\substack{\mathcal{S} \subset \mathcal{C}; \\ |\mathcal{S}| = \texttt{KeepList}[i]}}{\arg\min} \sum_{\mathbf{c}_k \in \mathcal{S}} \texttt{mean}(\texttt{Errors}[\mathbf{c}_k])$ // Keep top $\texttt{KeepList}[i]$ conditionings $\mathbf{c}_k$ with the lowest errors
  `PrevTrials = TrialList[i]`
end for
return $\underset{\mathbf{c}_i \in \mathcal{C}}{\arg\min}\ \texttt{mean}(\texttt{Errors}[\mathbf{c}_i])$

	Food101	CIFAR10	Aircraft	Oxford Pets	Flowers102	STL10	ImageNet
Diffusion Classifier	77.7	88.4	26.4	87.3	66.3	95.4	61.4
Time/img (s)	51	30	51	18	51	30	1000
Diffusion Classifier w/ discriminative pruning	78.7	88.4	26.8	86.4	67.0	95.4	62.6
Time/img (s)	35	30	35	18	35	30	150
Est. Time/img (s) at 128^2 res	2	2	2	1	2	2	9
CLIP ResNet-50	81.1	75.6	19.3	85.4	65.9	94.3	58.2

Table B.2: **Zero-shot accuracy and inference time with Stable Diffusion** 512×512**.** "Pruning" away unlikely classes with a weak discriminative classifier (e.g., CLIP ResNet-50) increases accuracy and reduces inference time. Additionally, reducing resolution to 128×128 would reduce inference time by roughly $16\times$. However, its impact on accuracy is difficult to estimate without retraining the Stable Diffusion model to expect lower resolutions. All times are estimated using a RTX 3090 GPU.

with the number of classes, even when using the adaptive strategy. One way to address this problem is to use a weak discriminative model to quickly "prune" away classes that are almost certainly incorrect. Table B.2 shows that using Diffusion Classifier to choose among the top 20 class predictions made by CLIP ResNet-50 for an image greatly reduces inference time, while even improving performance. This pruning procedure only requires the top-20 accuracy of the fast discriminative model to be high (close to 100%), so it works even when the top-1 accuracy of the ResNet-50 is low, like on Aircraft. We chose top-20 intuitively, without any hyperparameter search, and tuning the k for top-k pruning will trade off between inference time and accuracy. Note that no other results in this paper use the discriminative pruning procedure, to avoid conflating the capabilities of Diffusion Classifier with those of the weak discriminative model used to prune.

B.3 INFERENCE OBJECTIVE FUNCTION

	Food101	CIFAR10	Aircraft	Oxford Pets	Flowers102	STL10	ImageNet	ObjectNet
Squared ℓ_2	**77.7**	84.4	26.4	86.3	62.2	**95.4**	**61.4**	43.4
ℓ_1	73.8	**88.4**	22.1	**87.3**	**66.3**	**95.4**	59.6	36.8
Huber	**77.7**	84.6	**26.7**	86.6	62.6	**95.4**	60.9	**43.5**

Table B.3: **Diffusion Classifier zero-shot performance with different loss functions** $\mathcal{L}(\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}))$.

Resolution	Objective	ImageNet	ImageNetV2	ImageNet-A	ObjectNet
256^2	Squared ℓ_2	**77.5**	**64.6**	**20.0**	**32.1**
256^2	ℓ_1	74.9	60.5	9.7	24.7
512^2	Squared ℓ_2	**79.1**	**66.7**	**30.2**	**33.9**
512^2	ℓ_1	75.6	62.1	13.2	26.2

Table B.4: **Diffusion Classifier supervised performance with different loss functions** $\mathcal{L}(\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}))$.

While the theory in Section 3.1.1 justifies using $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|_2^2$ within the Diffusion Classifier inference objective, we surprisingly find that other loss functions can work better in some cases. Table B.3 shows that $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|_1$ (the ℓ_1 loss) instead of the squared ℓ_2 loss does better on roughly half of the datasets that we use to evaluate the Stable Diffusion-based zero-shot classifier. This is puzzling, since the ℓ_1 loss is neither theoretically justified nor appears in the Stable Diffusion training objective. We hope followup work can explain the empirical success of the ℓ_1 loss. Combining these two losses does not get the "best of both worlds." The Huber loss, which is the squared ℓ_2 loss for values less than 1 and is the ℓ_1 loss for values greater than 1, roughly achieves the same performance as the theoretically-justified squared ℓ_2 loss. We choose between squared ℓ_2 and ℓ_1 as a hyperparameter for Section 3.4.1. Table B.4 shows that ℓ_1 does not help with supervised classification (Section 3.4.3) using DiT-XL/2.

B.4 Interpretability via Image Generation

In contrast to discriminative classifiers, where it is difficult to understand what features the model has learned or why a model has made a certain decision, generative classifiers are easier to visualize. In this section, we examine how samples from the generative model can help us understand class-dependent features that the model has learned as well as failures in the model's understanding.

Experiment setup Given an input image, we first perform DDIM inversion [92, 175] (with 50 timesteps) using Stable Diffusion 2.0 and different captions as prompts: BLIP [107] generated caption, human-refined BLIP generated caption, "a photo of *{correct-class-name}*, a type of pet" and "a photo of *{incorrect-class-name}*, a type of pet.". Next, we leverage the inverted DDIM latent and the corresponding prompt to attempt to reconstruct the original image (using a deterministic diffusion scheduler

Figure B.1: **Analyzing Diffusion Classifier for Zero-Shot Classification:** We analyze the role of different text/captions (BLIP, Human-modified BLIP, correct class-name, incorrect class-name) for zero-shot classification using text-based diffusion models. To do so, we invert the input image using the corresponding caption and then reconstruct it using deterministic DDIM sampling. The image inverted and reconstructed using a human-modified BLIP caption aligns the most with the input image since this caption is the most descriptive. The images reconstructed using correct class names as prompts (column 4) align much better with the input image in terms of class-descriptive features of the underlying object than the images reconstructed using incorrect class names as prompts (columns 5 and 6). Row 3 (columns 4 and 5) demonstrates an example where the base Stable Diffusion does not understand the difference between the two cat breeds, Birman and Ragdoll, and hence cannot invert/sample them differently. As a result, our classifier also fails.

[175]). The underlying intuition behind this experiment is that the inverted image should look more similar to the original image when a correct and appropriate/descriptive prompt is used for DDIM inversion and sampling.

EXPERIMENTAL EVALUATION Figure B.1 shows the results of this experiment for the Oxford-IIIT Pets dataset. The image inverted using a human-modified BLIP caption (column 3) is the most similar to the original image (column 1). This aligns with our intuition as this caption is most descriptive of the input image. The human-modified caption only adds the correct class name (Bengal Cat, American Bull Dog, Birman Cat) ahead of the BLIP predicted "cat or dog" token for the foreground object and slightly enhances the description for the background. Comparing the BLIP-caption results (column 2) with the human-modified BLIP-caption results (column

3), we can see that by just using the class-name as the extra token, the diffusion model can inherit class-descriptive features. The Bengal cat has stripes, the American Bulldog has a wider chin, and the Birman cat has a black patch on its face in the reconstructed image.

Compared to the images generated using the human-generated caption as a prompt, the images reconstructed using only class names as prompts (columns 4,5,6) align less with the input image (column 1). This is expected, as class names by themselves are not dense descriptions of the input images. Comparing the results of column 4 (correct class names as prompt) with those of column 5,6 (incorrect class names as prompt), we can see that the foreground object has similar class-descriptive features (brown and black stripes in row 1 and black face patches in row 3) to the input image for the correct-prompt reconstructions. This highlights the fact that although using class names as approximate prompts will not lead to perfect denoising (Eq. 3.7), *for the global prediction task of classification, the correct class names should provide enough descriptive features for denoising, relative to the incorrect class names.*

Row 3 of Figure B.1 further highlights an example of a failure mode where Stable Diffusion generates very similar inverted images for correct Birman and incorrect Ragdoll text prompts. As a result, our model also incorrectly classifies the Birman cat as a Ragdoll. To fix this failure mode, we tried finetuning the Stable Diffusion model on a dataset of Ragdoll/Birman cats (175 images in total). Using this finetuned model, Diffusion Classifier accuracy on these two classes increases to 85%, from an initial zero-shot accuracy of 45%. In addition to minimizing the standard ϵ-prediction error $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_i)\|^2$, we found that adding a loss term to *increase* the error $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c}_j)\|^2$ for the wrong class $\mathbf{c}_j$ helped the model distinguish these commonly-confused classes.

B.5 How does Stable Diffusion version affect zero-shot accuracy?

We investigate how much the Stable Diffusion checkpoint version affects Diffusion Classifier's zero-shot classification accuracy. Table B.5 shows zero-shot accuracy for each Stable Diffusion release version so far. We use the same adaptive evaluation strategy (Algorithm 3) for each version. Accuracy improves with each new release for SD 1.x, as more training likely reduces underfitting on the training data. However, accuracy actually decreases when going from SD 2.0 to 2.1. The cause of this is not clear, especially without access to intermediate training checkpoints. One hypothesis is that further training on 512^2 resolution images causes the model to forget knowledge from its initial 256^2 resolution training set, which is closer to the distribution of these zero-shot benchmarks. SD 2.1 was finetuned using a more permissive NSFW

SD Version	Food101	CIFAR10	Aircraft	Oxford Pets	Flowers102	STL10	ImageNet	ObjectNet
1.1	60.3	83.4	20.1	78.8	43.1	92.6	51.7	38.1
1.2	75.7	85.9	26.3	85.4	54.4	94.4	57.3	39.4
1.3	77.5	**87.5**	27.8	87.2	54.5	**94.9**	**59.7**	40.9
1.4	77.8	86.0	28.6	87.4	54.2	94.8	59.2	41.2
1.5	**78.4**	85.5	**29.1**	**87.5**	**55.0**	94.5	59.6	**41.6**
2.0	77.7	**88.4**	**26.4**	**87.3**	**66.3**	**95.4**	**61.4**	**43.4**
2.1	**77.9**	87.1	24.3	86.2	59.4	95.3	58.4	38.3

Table B.5: **Effect of Stable Diffusion version on Diffusion Classifier zero-shot accuracy**. We bold the best version within SD 1.x and 2.x. For SD 1, accuracy tends to increase with more training. The main exception is on low-resolution datasets like CIFAR10 and STL10. SD 2 performance consistently decreases from SD 2.0 to SD 2.1 on almost every dataset.

threshold ($\geqslant 0.98$ instead of $\geqslant 0.1$), so another hypothesis is that this introduced a lot of human images that hurt performance on our object-centric benchmarks.

B.6 Additional implementation details

B.6.1 *Zero-shot classification using Diffusion Classifier*

Training data For our zero-shot Diffusion Classifier, we utilize Stable Diffusion 2.0 [154]. This model was trained on a subset of the LAION-5B dataset, filtered so that the training data is aesthetic and appropriately safe-for-work. LAION classifiers were used to remove samples that are too small (less than 512×512), potentially not-safe-for-work (punsafe $\geqslant 0.1$), or unaesthetic (aesthetic score $\leqslant 4.5$). These thresholds are conservative, since false negatives (NSFW or undesirable images left in the training set) are worse than removing extra images from a large starting dataset. As discussed in Section 3.4.1, these filtering criteria bias the distribution of Stable Diffusion training data and likely negatively affect Diffusion Classifier's performance on datasets whose images do not satisfy these criteria. SD 2.0 was trained for 550k steps at resolution 256×256 on this subset, followed by an additional 850k steps at resolution 512×512 on images that are at least that large. This checkpoint can be downloaded online through the diffusers repository at `stabilityai/stable-diffusion-2-0-base`.

INFERENCE DETAILS We use FP16 and Flash Attention [42] to improve inference speed. This enables efficient inference with a batch size of 32, which works across a variety of GPUs, from RTX 2080Ti to A6000. We found that adding these two tricks did not affect test accuracy compared to using FP32 without Flash Attention. Given a test image, we resize the shortest edge to 512 pixels using bicubic interpolation, take a 512×512 center crop, and normalize the pixel values to $[-1, 1]$. We then use the Stable Diffusion autoencoder to encode the $512 \times 512 \times 3$ RGB image into a $64 \times 64 \times 4$ latent. We finally classify the test image by applying the method described in Sections 3.1.1 and 3.2 to estimate ϵ-prediction error in this latent space.

B.6.2 *Compositional reasoning using Diffusion Classifier*

For our experiments on the Winoground benchmark [184], most details are the same as the zero-shot details described in Appendix B.6.1. We use Stable Diffusion 2.0, and we evaluate each image-caption pair with 1000 evenly spaced timesteps. We omit the adaptive inference strategy since there are only 4 image-caption pairs to evaluate for each Winoground example.

B.6.3 *ImageNet classification using Diffusion Classifier*

For this task, we use the recently proposed Diffusion Transformer (DiT) [141] as the backbone of our Diffusion Classifier. DiT was trained on ImageNet-1k, which contains about 1.28 million images from 1,000 classes. While it was originally trained to produce high-quality samples with strong FID scores, we repurpose the model and compare it against discriminative models with the same ImageNet-1k training data. We use the DiT-XL/2 model size at resolution 256^2 and 512^2. Notably, DiT achieves strong performance while using much weaker data augmentations than what discriminative models are usually trained with. During training time for the 256^2 checkpoint, the smaller edge of the input image is resized to 256 pixels. Then, a 256×256 center crop is taken, followed by a random horizontal flip, followed by embedding with the Stable Diffusion autoencoder. A similar process is done for the 512^2 model. At test time, we follow the same preprocessing pipeline, but omit the random horizontal flip. Classification performance could improve if stronger augmentations, like RandomResizedCrop or color jitter, are used during the diffusion model training process.

B.6.4 *Baselines for Zero-Shot Classification*

SYNTHETIC SD DATA: We provide the implementation details of the "Synthetic SD Data" baseline (row 1 of Table 3.1) for the task of zero-shot image classification. Our Diffusion Classifier approach builds on the intuition that a model capable of generating examples of desired classes should be able to directly discriminate between them. In contrast, this baseline takes the simple approach of using our generative model, Stable Diffusion, as intended to generate *synthetic training data* for a discriminative model. For a given dataset, we use pre-trained Stable Diffusion 2.0 with default settings to generate 10,000 synthetic 512×512 pixel images per class as follows: we use the English class name and randomly sample a template from those provided by the CLIP [146] authors to form the prompt for each generation. We then train a supervised ResNet-50 classifier using the synthetic data and the labels corresponding to the class name that was used to generate each image. We use batch size $= 256$, weight decay $= 1e-4$, learning rate $= 0.1$ with a cosine schedule, the AdamW optimizer, and use random resized crop & horizontal flip transforms. We create a validation set using the synthetic data by randomly selecting 10% of the images for each class; we use this for early stopping to prevent over-fitting. Finally, we report the accuracy on the target dataset's proper test set.

SD FEATURES: We provide the implementation details of the "SD Features" baseline (row 2 of Table 3.1) for the task of image classification. This baseline is inspired by Label-DDPM [10], a recent work on diffusion-based semantic segmentation. Unlike Label-DDPM, which leverages a category-specific diffusion model, we directly build on top of the open-sourced Stable Diffusion model (trained on the LAION dataset). We then approach the task of classification as follows: given the pre-trained Stable Diffusion model, we extract the intermediate U-Net features corresponding to the input image. These features are then passed through a ResNet-based classifier to predict logits for the potential classes. To extract the intermediate U-Net features, we add a noise equivalent to the 100th timestep noise to the input image and evaluate the corresponding noisy latent using the forward diffusion process. We then pass the noisy latent through the U-Net model, conditioned on timestep $t = 100$ and text conditioning ($\mathbf{c}$) as an empty string, and extract the features from the mid-layer of the U-Net at a resolution of $[8 \times 8 \times 1024]$. Next, we train a supervised classifier on top of these features. *Thus, this baseline is not zero-shot.* The architecture of our classifier is similar to ResNet-18, with small modifications to make it compatible with an input size of $[8 \times 8 \times 1024]$. Table B.6 defines these modifications. We set batch size $= 16$, learning rate $= 1e-4$, and use the AdamW optimizer. During training, we apply image augmentations typically used by discriminative classifiers (Random-

Arch	Conv1	Conv2	Conv3 x2	Conv4 x2	Conv5 x2
ResNet-18	7x7x64	3x3 max-pool	3x3x128	3x3x256	3x3x512
ResNet-18 (SD Features)	3x3x1280	-	3x3x1280	3x3x2560	3x3x2560

Table B.6: Comparison of SD Features' ResNet-18 classifier architecture with the original ResNet-18

ResizedCrop and horizontal flip). We do early stopping using the validation set to prevent overfitting.

B.7 TECHNIQUES THAT DID NOT HELP

Diffusion Classifier requires many samples to accurately estimate the ELBO. In addition to using the techniques in Section 3.1.1 and 3.2, we tried several other options for variance reduction. None of the following methods worked, however. We list negative results here for completeness, so others do not have to retry them.

CLASSIFIER-FREE GUIDANCE Classifier-free guidance [77] is a technique that improves the match between a prompt and generated image, at the cost of mode coverage. This is done by training a conditional $\epsilon_\theta(\mathbf{x}_t, \mathbf{c})$ and unconditional $\epsilon_\theta(\mathbf{x}_t)$ denoising network and combining their predictions at sampling time:

$$\tilde{\epsilon}(\mathbf{x}_t, \mathbf{c}) = (1+w)\epsilon_\theta(\mathbf{x}_t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}_t) \tag{B.1}$$

where w is a guidance weight that is typically in the range $[0, 10]$. Most diffusion models are trained to enable this trick by occasionally replacing the conditioning $\mathbf{c}$ with an empty token. Intuitively, classifier-free guidance "sharpens" $\log p_\theta(x \mid \mathbf{c})$ by encouraging the model to move away from regions that unconditionally have high probability. We test Diffusion Classifier to see if using the $\tilde{\epsilon}$ from classifier-free guidance can improve confidence and classification accuracy. Our new ϵ-prediction metric is now $\|\epsilon - (1+w)\epsilon_\theta(\mathbf{x}_t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}_t)\|^2$. However, Figure B.2 shows that $w = 0$ (i.e., no classifier-free guidance) performs best. We hypothesize that this is because Diffusion Classifier fails on uncertain examples, which classifier-free guidance affects unpredictably.

ERROR MAP CROPPING The ELBO $\mathbb{E}_{t,\epsilon}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})\|^2]$ depends on accurately estimating the added noise at every location of the $64 \times 64 \times 4$ image latent. We try to reduce the impact of edge pixels (which are less likely to contain the subject) by

computing $\mathbf{x}_t$ as normal, but only measuring the ELBO on a center crop of ϵ and $\epsilon_\theta(\mathbf{x}_t, \mathbf{c})$. We compute:

$$\|\epsilon_{[i:-i,i:-i]} - \epsilon_\theta(\mathbf{x}_t, \mathbf{c})_{[i:-i,i:-i]}\|^2 \tag{B.2}$$

where i is the number of latent "pixels" to remove from each edge. However, Figure B.3 shows that any amount of cropping reduces accuracy.

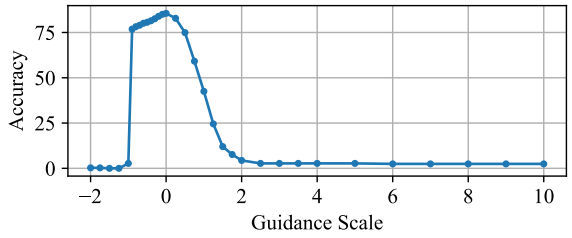


Figure B.2: Accuracy plot of classifier-free guidance on Pets.

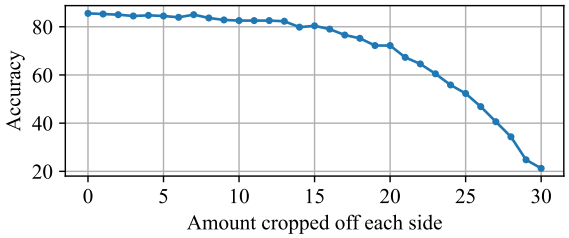


Figure B.3: Cropping ϵ and $\epsilon_\theta(\mathbf{x}_t, \mathbf{c})$ reduces accuracy on Pets.

IMPORTANCE SAMPLING Importance sampling is a common method for reducing the variance of a Monte Carlo estimate. Instead of sampling $\epsilon \sim \mathcal{N}(0, I)$, we sample ϵ from a narrower distribution. We first tried fixing $\epsilon = 0$, which is the mode of $\mathcal{N}(0, I)$, and only varying the timestep t. We also tried the truncation trick [20] which samples $\epsilon \sim \mathcal{N}(0, I)$ but continually resamples elements that fall outside the interval $[a, b]$. Finally, we tried sampling $\epsilon \sim \mathcal{N}(0, I)$ and rescaling them to the expected norm ($\epsilon \to \frac{\epsilon}{\|\epsilon\|_2} \mathbb{E}_{\epsilon'}[\|\epsilon'\|_2]$) so that there are no outliers. Table B.7 shows that none of these importance sampling strategies improve accuracy. This is likely because the noise ϵ sampled with these strategies are completely out-of-distribution for the noise prediction model. For computational reasons, we performed this experiment on a 10% subset of Pets.

Sampling distribution for ϵ	Pets accuracy
$\epsilon = 0$	41.3
TruncatedNormal, $[-1, 1]$	49.9
TruncatedNormal, $[-2.5, 2.5]$	81.5
Expected norm	86.9
$\epsilon \sim \mathcal{N}(0, I)$	87.5

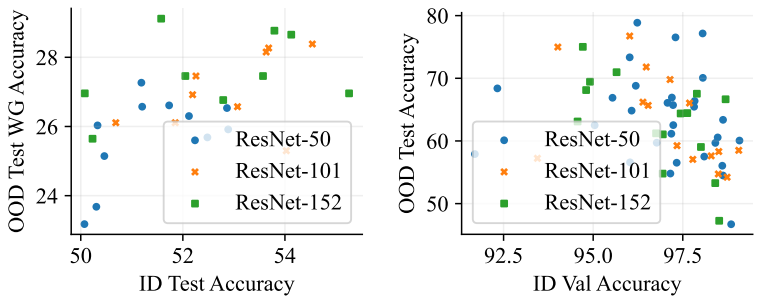
Table B.7: Every importance sampling strategy underperforms sampling the noise ϵ from a standard normal distribution.

C

GENERATIVE CLASSIFIERS AVOID SHORTCUT SOLUTIONS

C.1 ADDITIONAL ANALYSIS

C.1.1 *Additional Results on the Effect of Discriminative Model Size*



(a) FMoW (b) Camelyon17

Figure C.1: **Scaling up discriminative model size does not improve performance**. Each point with the same color is a model trained with different hyperparameters (learning rate and weight decay). Results on Waterbirds are shown in Figure 4.4.

We add additional results to our investigation into the role of discriminative model size. Previously, our analysis of CivilComments in Section 4.4.3 showed that matching the parameter count between the discriminative and generative classifiers did not account for the qualitative differences in their generalization behavior. Furthermore, Figure 4.4 showed that increasing model size on Waterbirds did not improve performance. Figure C.1 shows additional results. On FMoW, scaling only helps when going from ResNet-50 to ResNet-101; further scaling did not help. On Camelyon17, increasing model size had no effect on performance. Overall, we can confidently conclude that model size is not responsible for generative classifiers' improved robustness to distribution shift.

C.1.2 *Scaling Can Improve Generative Classifiers*

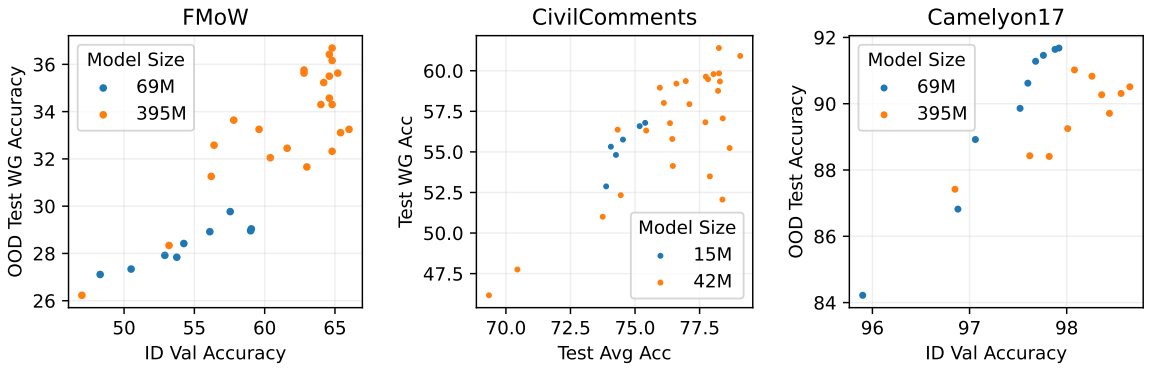


Figure C.2: **Effect of scaling up generative classifiers**. Increasing the number of parameters helps significantly on FMoW and CivilComments, but can sometimes hurt: OOD accuracy drops on Camelyon17 with a larger generative classifier.

Scaling model size has proved extremely effective for generative models in other settings [21, 81, 91]. This has typically been done in the "almost infinite data" regime, where only a few epochs are used, and overfitting is not an issue. Does scaling similarly help here for our generative classifiers?

We tried different model scales on three of our distribution shift benchmarks: FMoW, CivilComments, and Camelyon17. Figure C.2 shows the results of our investigation. On FMoW and CivilComments, scaling model size significantly improves performance both in- and out-of-distribution. However, on Camelyon17, a smaller model actually does significantly better out-of-distribution than a model that is 5.5 times as large. This indicates that overfitting can become an issue in this setting, where we have limited training data and must be careful about overfitting. Overall, we are excited by the fact that scaling generative classifiers *can be beneficial* in some settings, unlike discrimiminative classifiers, which consistently show poor use of extra model capacity (see Figure 4.4, Table 4.2, Figure C.1).

C.1.3 *Results on Additional Datasets*

We additionally run experiments on two highly-used subpopulation shift benchmarks from BREEDS [164]: Living-17 (with 17 animal classes) and Entity-30 (with 30 classes). As usual, we train our diffusion-based generative classifiers from scratch on each training set and evaluate them on the in-distribution and out-of-distribution test sets. We compare against discriminative baselines reported in the original BREEDS paper, which includes interventions such as stronger augmentations or adversarial

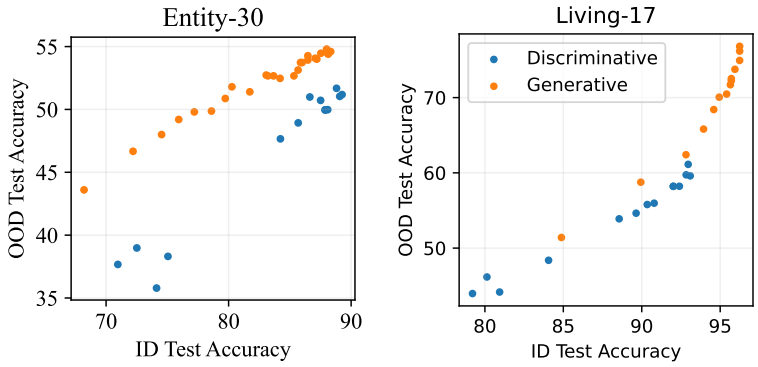


Figure C.3: **In-distribution vs out-of-distribution accuracy** for additional subpopulation shift datasets [164]. We again observe OOD scaling trends for generative classifiers. Each point for a discriminative model corresponds to a separate model with a different architecture, augmentation, or adversarial training method. Accuracies for the discriminative models are taken from Santurkar, Tsipras, and Madry [164].

training. Figure C.3 displays the same trends here as our main results. Both datasets display effective robustness (for a given ID accuracy, the OOD accuracy of the generative classifier is higher), though the effect is much stronger on Entity-30.

C.1.4 *Correlation between Generative and Discriminative Performance*

We take a careful look at how well generative capabilities like validation likelihood and sample quality correlate with classification performance. Figure C.4 shows how these three metrics evolve over the course of training for a diffusion-based generative classifier on CelebA.

We first find that the model does not need to generate good samples in order to have high classification accuracy. The first generation in Figure C.4 has significant visual artifacts, yet the generative classifier already achieves 90% class-balanced accuracy. This makes sense: for ground-truth class y^*, the classifier only needs $p_\theta(x \mid y^*) > p_\theta(x \mid y)$ for all other classes $y \neq y^*$, so $p_\theta(x \mid y^*)$ can be low as long as $p_\theta(x \mid y \neq y^*)$ is even lower. In fact, given a generative classifier $p_\theta(x \mid y)$, one can construct another generative classifier $\tilde{p}(x \mid y) = \lambda p_\theta(x \mid y) + (1-\lambda)p_{\text{other}}(x)$, which has the same accuracy as p_θ but generates samples that look increasingly like p_{other} as $\lambda \to 0^+$.

However, even though sample quality is not necessary for high accuracy, we do find that validation diffusion loss correlates well with class-balanced accuracy. As the loss decreases, class-balanced accuracy correspondingly increases. Figure C.5

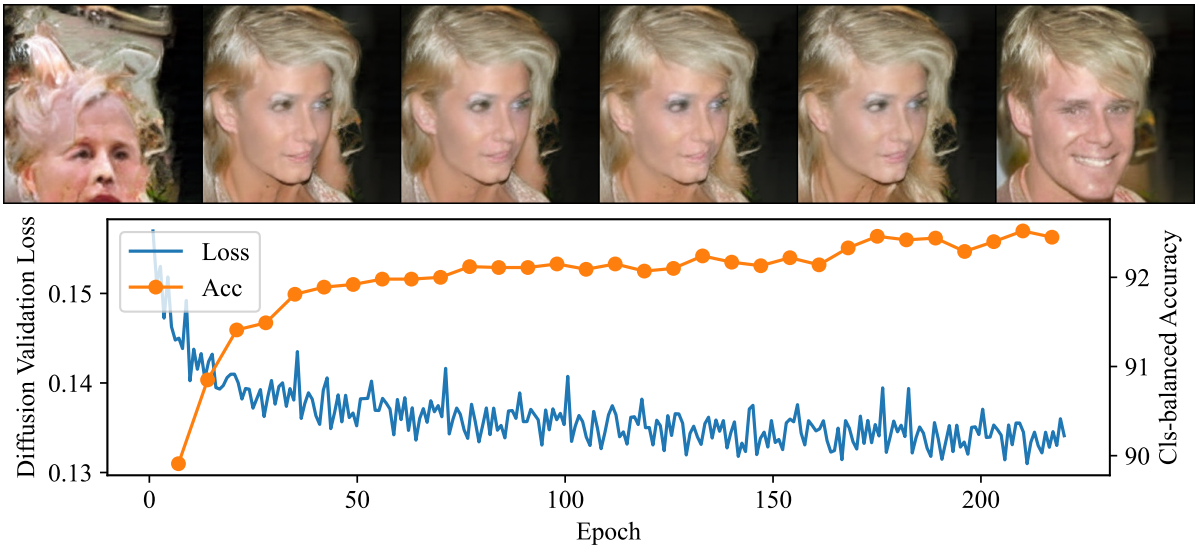


Figure C.4: **Correlation between accuracy and generative performance. Top**: class-conditional DDIM samples generated from the same noise using intermediate checkpoints. **Bottom**: diffusion validation loss and class-balanced accuracy on CelebA by training epoch. **Main findings**: First, high classification accuracy can be achieved even without good sample quality (see the first generation). Second, generative validation loss is highly correlated with classification accuracy. Third, as training progresses, the minority group (blond men) becomes more likely, indicating that the generative classifier correctly models less correlation between hair color (causal) and gender (shortcut).

shows how an increase in validation diffusion loss due to overfitting translates to a corresponding decrease in classification accuracy on Waterbirds.

Finally, Figure C.4 shows how we can check the samples to audit how the generative classifier models the spurious vs core features. The samples are generated deterministically with DDIM [174] from a fixed starting noise, so the sample from the last checkpoint shows that the model is increasing the probability of blond men (the minority group in CelebA). This means that the model is modeling less correlation between the hair color (causal for the blond vs not blond label) and the gender (the shortcut feature). This is one additional advantage of generative classifiers: generating samples is a built-in interpretability method [106]. Again, as we note above, generation of a specific feature is sufficient but not necessary to show that it is being used for classification.

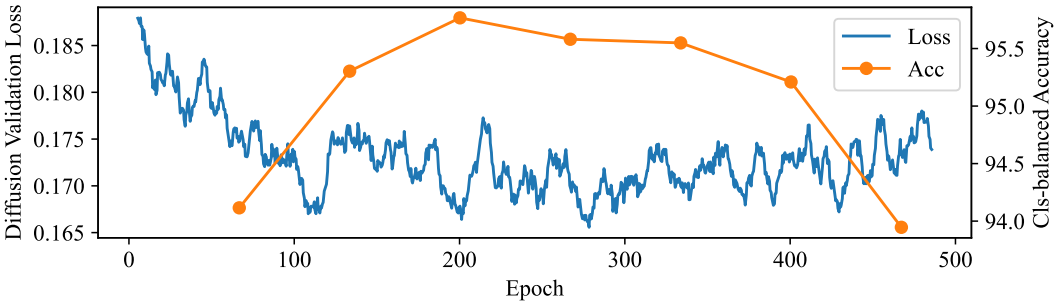


Figure C.5: Overfitting in diffusion loss on Waterbirds directly translates to overfitting in classification accuracy. We smooth the loss for better visual clarity.

Embedding model	Waterbirds		CelebA		Camelyon	
	ID	WG	ID	WG	ID	OOD
Pre-trained VAE [154]	**96.8**	**79.4**	91.2	69.4	98.3	90.8
PCA patch embeddings [31]	93.8	61.7	**91.3**	**71.1**	**98.7**	**93.8**

Table C.1: **Effect of image embedding model.** We compare different image encoders, which map the image from $256 \times 256 \times 3$ to $32 \times 32 \times 4$. For our main results, we use the pre-trained deep VAE released in the original LDM paper [154]. We compare it to a PCA-based patch embedding that tokenizes each $8 \times 8 \times 3$ patch independently and is trained separately on each dataset. We find that the pre-trained VAE is not consistently better, as it only does better on 1 of the 3 datasets that we tested the PCA encoder on.

C.1.5 *Effect of Image Embedding Model*

For our image results in the main paper, we trained latent diffusion models from scratch for each dataset. In order to be consistent with the generative modeling literature and keep the diffusion model training pipeline *completely unmodified*, we trained the diffusion models on the latent space of a pre-trained VAE [154]. This VAE compresses $256 \times 256 \times 3$ images into $32 \times 32 \times 4$ latent embeddings, which are cheaper to model. Perhaps our generative classifier is benefiting from an encoder that makes use of extra pre-training data? We test this hypothesis by trying to remove as much of the pre-trained encoding as possible. Following previous analysis work on diffusion models [31], we replace the VAE network with a simple PCA-based encoding of each image patch. Specifically, we turn each image into 32×32 total $8 \times 8 \times 3$ pixel patches, and use PCA to find the top 4 principal components of the patches. When encoding, we normalize by the corresponding singular values to ensure that the PCA embeddings have approximately the same variance in each dimension. Overall, we perform this process separately on each training dataset, which completely removes the effect of pre-training, and train a diffusion model for each dataset within the PCA latent space. Table C.1 compares this embedding model to the VAE and finds that it actually performs *better* on 2 of the 3 datasets. We conclude that the pre-trained encoder does not have a significant directional effect on our generative classifier results.

C.1.6 *Comparison with Pre-trained Discriminative Models*

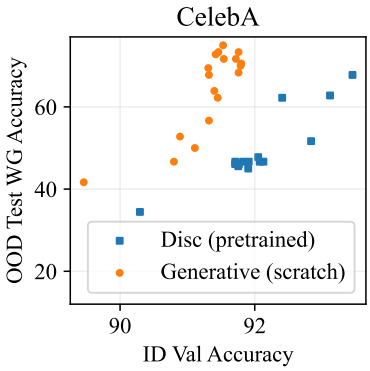


Figure C.6: Finetuning a pretrained discriminative model improves performance, but it still does not achieve the same "effective robustness" as our generative classifier.

All of our experiments so far train the classifier (whether discriminative or generative) from scratch. This is done to ensure a fair, apples-to-apples comparison

between methods. What happens if we use a pretrained discriminative model? In preliminary comparisons, we use a ResNet-50 pretrained with supervised learning on ImageNet-1k [101] and finetune it on CelebA. Figure C.6 shows the results of this *unfair comparison* between a pretrained discriminative model versus our generative classifier trained from scratch. We find that pretraining helps, but it does not significantly close the gap with the generative classifier. This is in spite of the fact that the discriminative model has seen an extra 1.2 million labeled training images, those labels have more bits (since there are 1000 classes instead of just two), and the pretraining classification task has minimal spurious correlations that are relevant to the downstream task.

C.1.7 *Additional Plots for Generalization Phase Diagrams*

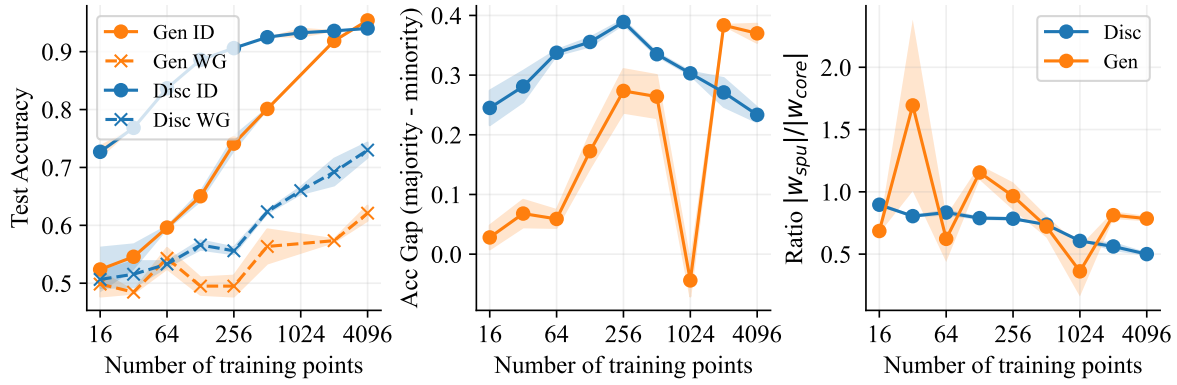


Figure C.7: Comparing logistic regression and LDA when the core feature variance has been increased from $\sigma = 0.15$ to $\sigma = 0.6$. The generative approach's accuracy drops much more in this setting.

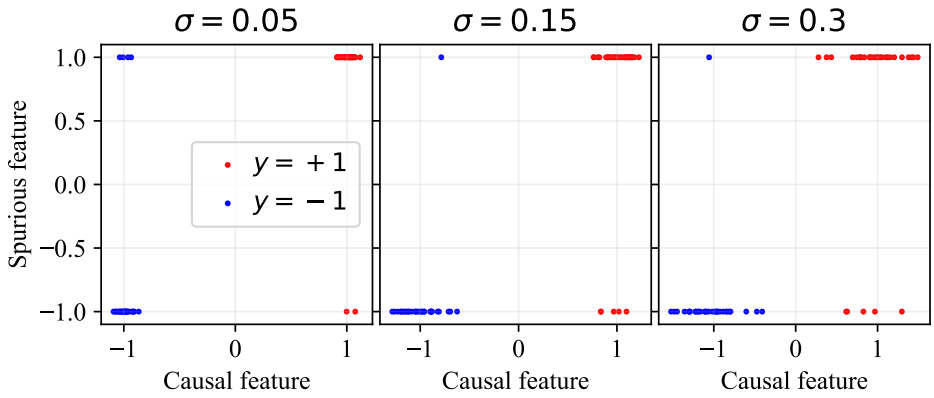


Figure C.8: Effect of varying the standard deviation σ of the core feature. $d - 2$ noise dimensions not shown. These correspond to the σ shown in Figure 4.7.

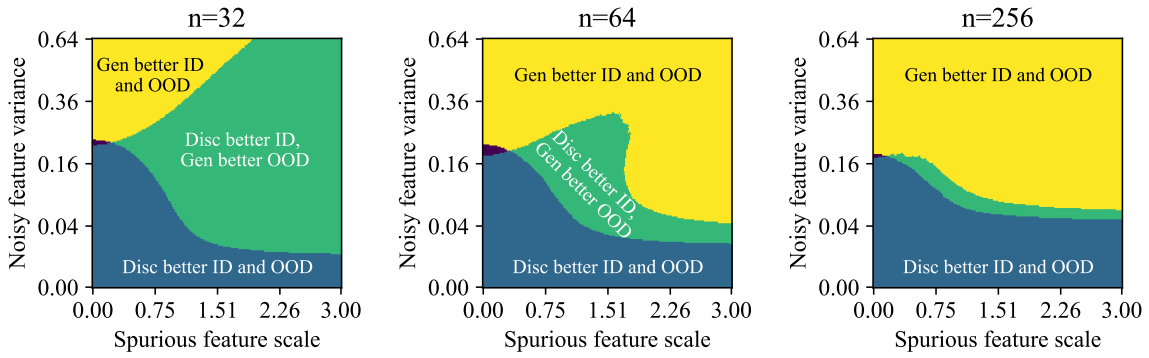


Figure C.9: Each plot corresponds to a different number n of training examples.

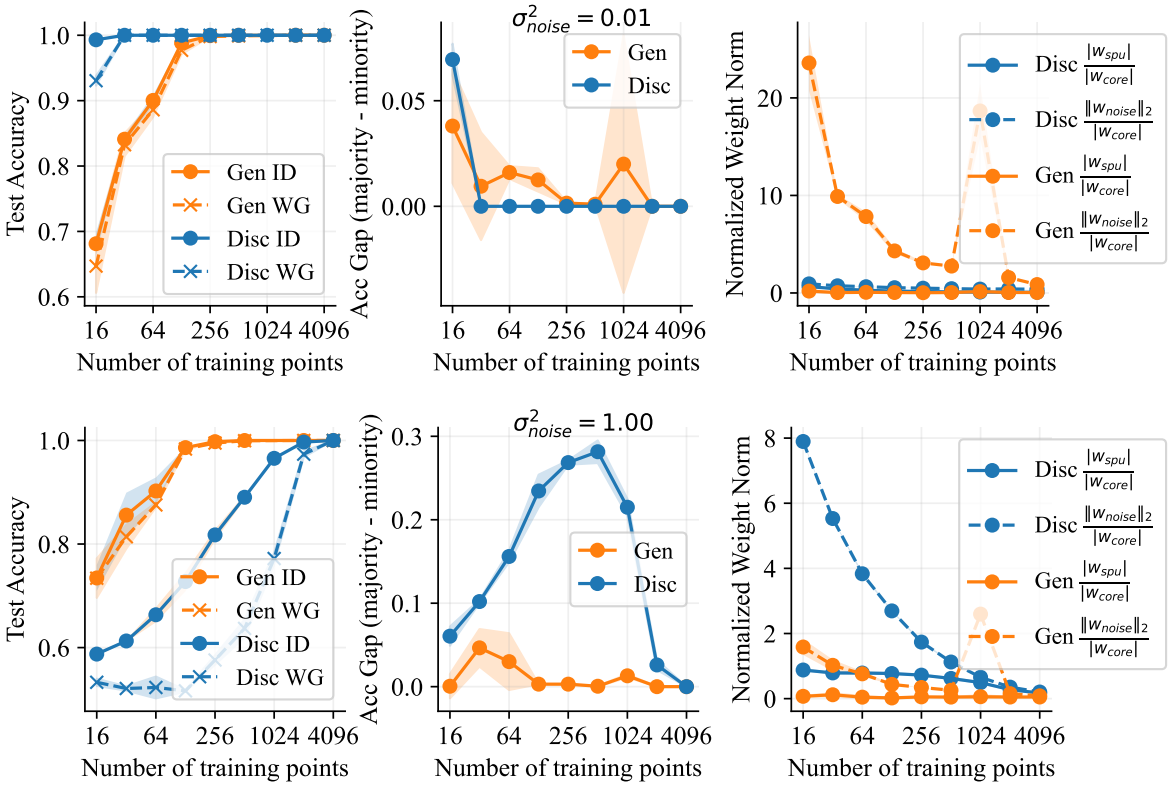


Figure C.10: Effect of σ_{noise} on the generalization of SVM vs LDA. Larger σ_{noise} makes it easier for SVM to overfit, since it uses the high-norm noise features to increase its margin. Lower σ_{noise} makes it harder to overfit, since the noise features are too small to significantly increase the margin.

C.2 EXPERIMENTAL DETAILS

Algorithm 4 `Generative Classifier`

Input: Training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
Training model $p_\theta(x \mid y)$**:**
    Minimize generative loss $\mathbb{E}_{(x,y)\sim\mathcal{D}}[-\log p_\theta(x \mid y)]$
Classification of test input x**:**
for class $\mathbf{y}_i \in \mathcal{Y}$ **do**
  Compute $p_\theta(x|y_i)$
end for
Return $\arg\max_{y_i} p_\theta(x \mid y_i)p(y_i)$

C.2.1 *Image-based Experiments*

C.2.1.1 *Diffusion-based Generative Classifier*

We train diffusion models from scratch in a lower-dimensional latent space [154]. We use the default 395M parameter class-conditional UNet architecture and train it from scratch with AdamW [116] with a constant base learning rate of 1e-6 and no weight decay or dropout. We did not tune diffusion model hyperparameters and simply used the default settings for conditional image generation. Again, we emphasize: *we achieved SOTA accuracies under distribution shift, using the default hyperparameters from image generation.*

Each diffusion model requires about 3 A6000 days to train. For inference on Waterbirds, CelebA, and Camelyon, we sample 100 noises ϵ and use them with each of the two classes. For FMoW, we use the adaptive strategy from Diffusion Classifier [106] that uses 100 samples per class, then does an additional 400 samples for the top 5 remaining classes.

C.2.1.2 *Discriminative Baselines*

We use the official training codebase released by Koh et al. [95] to train our discriminative baselines. For image-based benchmarks, we train 3 model scales (ResNet-50, ResNet-101, and ResNet-152) and sweep over 4 learning rates and 4 weight decay parameters. We use standard augmentations: normalization, random horizontal flip, and RandomResizedCrop.

C.2.2 *Autoregressive Generative Classifier*

For training, we pad shorter sequences to a length of 512 and only compute loss for non-padded tokens. We use a Llama-style architecture [186] and train 15M and 42M parameter models from scratch. We train for up to 200k iterations, which can take 2 A6000 days. We use a repository without default hyperparameters, so we sweep over learning rate, weight decay, and dropout based on their effect on the data log-likelihood. The resulting family of models is then shown in Figure 4.2.

C.2.2.1 *Discriminative Baselines*

For CivilComments, we use a randomly initialized encoder-only transformer with the same architecture as DistilBert [163]. We train for 100 epochs and sweep over dropout rate, learning rate, and weight decay.

BIBLIOGRAPHY

[1] Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. "Semdedup: Data-efficient learning at web-scale through semantic deduplication." In: *arXiv preprint arXiv:2303.09540* (2023).

[2] Ehab A AlBadawy, Ashirbani Saha, and Maciej A Mazurowski. "Deep learning for segmentation of brain tumors: Impact of cross-institutional training and testing." In: *Medical physics* 45.3 (2018), pp. 1150–1158.

[3] Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter, and Roger B Grosse. "Path independent equilibrium models can better exploit test-time computation." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 7796–7809.

[4] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. "Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models." In: *arXiv preprint arXiv:2503.09573* (2025).

[5] Peter Auer. "Using confidence bounds for exploitation-exploration trade-offs." In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.

[6] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. "Structured denoising diffusion models in discrete state-spaces." In: *Advances in neural information processing systems* 34 (2021), pp. 17981–17993.

[7] Gregor Bachmann and Vaishnavh Nagarajan. "The pitfalls of next-token prediction." In: *arXiv preprint arXiv:2403.06963* (2024).

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep equilibrium models." In: *Advances in neural information processing systems* 32 (2019).

[9] Hangbo Bao, Li Dong, and Furu Wei. "Beit: Bert pre-training of image transformers." In: *arXiv preprint arXiv:2106.08254* (2021).

[10] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. "Label-Efficient Semantic Segmentation with Diffusion Models." In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=SlxSY2UZQT.

[11] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua B. Tenenbaum, and Boris Katz. "ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models." In: *Neural Information Processing Systems*. 2019.

[12] Adrien Bardes, Jean Ponce, and Yann LeCun. "Vicreg: Variance-invariance-covariance regularization for self-supervised learning." In: *arXiv preprint arXiv:2105.04906* (2021).

[13] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. "Efficient training of language models to fill in the middle." In: *arXiv preprint arXiv:2207.14255* (2022).

[14] Sara Beery, Grant Van Horn, and Pietro Perona. "Recognition in terra incognita." In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 456–473.

[15] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. "Birdsnap: Large-scale fine-grained visual categorization of birds." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2011–2018.

[16] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians." In: *Journal of the American Statistical Association* (2017).

[17] Roger Bohn and James E Short. "Info capacity| measuring consumer information." In: *International Journal of Communication* 6 (2012), p. 21.

[18] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101–mining discriminative components with random forests." In: *European conference on computer vision*. Springer. 2014, pp. 446–461.

[19] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests." In: *European Conference on Computer Vision*. 2014.

[20] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis." In: *arXiv preprint arXiv:1809.11096* (2018).

[21] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[22] Johannes Buchner. *imagehash (fork)*. https://github.com/JohannesBuchner/imagehash. 2021.

[23] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. "A continuous time framework for discrete denoising models." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 28266–28279.

[24] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. "Toward an architecture for never-ending language learning." In: *Twenty-Fourth AAAI conference on artificial intelligence*. 2010.

[25] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging properties in self-supervised vision transformers." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660.

[26] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. "Maskgit: Masked generative image transformer." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11315–11325.

[27] Huanran Chen, Yinpeng Dong, Shitong Shao, Zhongkai Hao, Xiao Yang, Hang Su, and Jun Zhu. "Your diffusion model is secretly a certifiably robust classifier." In: *arXiv preprint arXiv:2402.02316* (2024).

[28] Huanran Chen, Yinpeng Dong, Zhengyi Wang, Xiao Yang, Chengqi Duan, Hang Su, and Jun Zhu. "Robust classification via a single diffusion model." In: *arXiv preprint arXiv:2305.15241* (2023).

[29] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In: *preprint arXiv:2002.05709* (2020).

[30] Xinlei Chen and Abhinav Gupta. "Webly supervised learning of convolutional networks." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1431–1439.

[31] Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. "Deconstructing denoising diffusion models for self-supervised learning." In: *arXiv preprint arXiv:2401.14404* (2024).

[32] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. "Neil: Extracting visual knowledge from web data." In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1409–1416.

[33] Xinlei Chen, Saining Xie, and Kaiming He. "An empirical study of training self-supervised vision transformers." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9640–9649.

[34] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. "Sequoia: Scalable, robust, and hardware-aware speculative decoding." In: *arXiv preprint arXiv:2402.12374* (2024).

[35] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. "Reproducible scaling laws for contrastive language-image learning." In: *arXiv preprint arXiv:2212.07143* (2022).

[36] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. "Functional Map of the World." In: *CVPR*. 2018.

[37] Kevin Clark and Priyank Jaini. "Text-to-Image Diffusion Models are Zero Shot Classifiers." In: *Advances in Neural Information Processing Systems* 36 (2023).

[38] Joe Clinton. *Google Images Download (fork)*. https://github.com/Joeclinton1/google-images-download. 2020.

[39] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 215–223.

[40] David Cohn, Les Atlas, and Richard Ladner. "Improving generalization with active learning." In: *Machine learning* 15 (1994), pp. 201–221.

[41] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. "Randaugment: Practical automated data augmentation with a reduced search space." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.

[42] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness." In: *Advances in Neural Information Processing Systems*. 2022.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *preprint arXiv:1810.04805* (2018).

[45] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis." In: *NeurIPS* (2021).

[46] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. "Continuous diffusion for categorical data." In: *arXiv preprint arXiv:2211.15089* (2022).

[47] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. "Measuring and mitigating unintended bias in text classification." In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 2018, pp. 67–73.

[48] Pierre Dognin, Igor Melnyk, Youssef Mroueh, Inkit Padhi, Mattia Rigotti, Jarret Ross, Yair Schiff, Richard A Young, and Brian Belgodere. "Image captioning as an assistive technology: Lessons learned from vizwiz 2020 challenge." In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 437–459.

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *preprint arXiv:2010.11929* (2020).

[50] Talfan Evans, Shreya Pathak, Hamza Merzic, Jonathan Schwarz, Ryutaro Tanno, and Olivier J Henaff. "Bad students make great teachers: Active learning accelerates large-scale visual understanding." In: *arXiv preprint arXiv:2312.05328* (2023).

[51] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." In: *IJCV* (2010).

[52] Alex Fang, Gabriel Ilharco, Mitchell Wortsman, Yuhao Wan, Vaishaal Shankar, Achal Dave, and Ludwig Schmidt. "Data determines distributional robustness in contrastive language image pre-training (clip)." In: *International Conference on Machine Learning*. PMLR. 2022, pp. 6216–6234.

[53] Vitaly Feldman and Chiyuan Zhang. "What neural networks memorize and why: Discovering the long tail via influence estimation." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2881–2891.

[54] Robert Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. "Learning object categories from google's image search." In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1816–1823.

[55] Ronald A Fisher. "The use of multiple measurements in taxonomic problems." In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[56] Weifeng Ge. "Deep metric learning with hierarchical triplet loss." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 269–285.

[57] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. "Shortcut learning in deep neural networks." In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[58] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." In: *International conference on learning representations*. 2018.

[59] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. "Made: Masked autoencoder for distribution estimation." In: *International conference on machine learning*. PMLR. 2015, pp. 881–889.

[60] Shantanu Ghosh, Ke Yu, Forough Arabshahi, and Kayhan Batmanghelich. "Route, Interpret, Repeat: Blurring the line between post hoc explainability and interpretable models." In: *arXiv preprint arXiv:2307.05350* (2023).

[61] Aaron Gokaslan and Vanya Cohen. "OpenWebTextCorpus https://skylion007. github. io." In: *OpenWebTextCorpus.</b* (2019).

[62] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. "Diffuseq: Sequence to sequence text generation with diffusion models." In: *arXiv preprint arXiv:2210.08933* (2022).

[63] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. "Your classifier is secretly an energy based model and you should treat it like one." In: *International Conference on Learning Representations*. 2020.

[64] Jean-Bastien Grill et al. "Bootstrap your own latent: A new approach to self-supervised learning." In: *NeurIPS*. 2020.

[65] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. "Studying large language model generalization with influence functions." In: *arXiv preprint arXiv:2308.03296* (2023).

[66] Ishaan Gulrajani and Tatsunori B Hashimoto. "Likelihood-based diffusion language models." In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 16693–16715.

[67] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. "Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control." In: *arXiv preprint arXiv:2210.17432* (2022).

[68] Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. "Smart mining for deep metric learning." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2821–2829.

[69] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked Autoencoders Are Scalable Vision Learners." In: *arXiv:2111.06377* (2021).

[70] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked autoencoders are scalable vision learners." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[71] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In: *CVPR*. 2020.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *CVPR*. 2016.

[73] Dan Hendrycks and Thomas Dietterich. "Benchmarking neural network robustness to common corruptions and perturbations." In: *arXiv preprint arXiv:1903.12261* (2019).

[74] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. "Natural Adversarial Examples." In: *CVPR* (2021).

[75] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. "Natural adversarial examples." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 15262–15271.

[76] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[77] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance." In: *arXiv preprint arXiv:2207.12598* (2022).

[78] Jonathan Ho et al. *Imagen Video: High Definition Video Generation with Diffusion Models*. 2022.

[79] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[80] Micah Hodosh, Peter Young, and Julia Hockenmaier. "Framing image description as a ranking task: Data, models and evaluation metrics." In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 853–899.

[81] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. "Training compute-optimal large language models." In: *arXiv preprint arXiv:2203.15556* (2022).

[82] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. "Autoregressive diffusion models." In: *arXiv preprint arXiv:2110.02037* (2021).

[83] Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. "Simple data balancing achieves competitive worst-group-accuracy." In: *Conference on Causal Learning and Reasoning*. PMLR. 2022, pp. 336–351.

[84] Gabriel Ilharco, Mitchell Wortsman, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, et al. "OpenCLIP." In: *Zenodo* 4 (2021), p. 5.

[85] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. "Datamodels: Predicting predictions from training data." In: *arXiv preprint arXiv:2202.00622* (2022).

[86] Priyank Jaini, Kevin Clark, and Robert Geirhos. "Intriguing properties of generative classifiers." In: *arXiv preprint arXiv:2309.16779* (2023).

[87] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. "Improving contrastive learning on imbalanced data via open-world sampling." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5997–6009.

[88] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs." In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 535–547.

[89] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold." In: *Nature* 596.7873 (2021), pp. 583–589.

[90] Amita Kamath, Christopher Clark, Tanmay Gupta, Eric Kolve, Derek Hoiem, and Aniruddha Kembhavi. "Webly Supervised Concept Expansion for General Purpose Vision Models." In: *arXiv preprint arXiv:2202.02317* (2022).

[91] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." In: *arXiv preprint arXiv:2001.08361* (2020).

[92] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. "DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 2426–2435.

[93] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. "Last layer re-training is sufficient for robustness to spurious correlations." In: *arXiv preprint arXiv:2204.02937* (2022).

[94] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions." In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894.

[95] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. "Wilds: A benchmark of in-the-wild distribution shifts." In: *International conference on machine learning*. PMLR. 2021, pp. 5637–5664.

[96] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. "Big transfer (BiT): General visual representation learning." In: *ECCV*. 2020.

[97] Daisuke Komura and Shumpei Ishikawa. "Machine learning methods for histopathological image analysis." In: *Computational and structural biotechnology journal* 16 (2018), pp. 34–42.

[98] Simon Kornblith, Jonathon Shlens, and Quoc V Le. "Do better imagenet models transfer better?" In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2661–2671.

[99] Elisa Kreiss, Fei Fang, Noah D Goodman, and Christopher Potts. "Concadia: Towards image-based text generation with a purpose." In: *arXiv preprint arXiv:2104.08376* (2021).

[100] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)." In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[101] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012).

[102] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. "Deduplicating training data makes language models better." In: *arXiv preprint arXiv:2107.06499* (2021).

[103] Yoonho Lee, Huaxiu Yao, and Chelsea Finn. *Diversify and Disambiguate: Learning From Underspecified Data*. 2023. arXiv: 2202.03418 [cs.LG].

[104] Yaniv Leviathan, Matan Kalman, and Yossi Matias. "Fast inference from transformers via speculative decoding." In: *International Conference on Machine Learning*. PMLR. 2023, pp. 19274–19286.

[105] Alexander C Li, Alexei A Efros, and Deepak Pathak. "Understanding Collapse in Non-Contrastive Siamese Representation Learning." In: *European Conference on Computer Vision*. Springer. 2022, pp. 490–505.

[106] Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. "Your Diffusion Model is Secretly a Zero-Shot Classifier." In: *arXiv preprint arXiv:2303.16203* (2023).

[107] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation." In: *International Conference on Machine Learning*. PMLR. 2022, pp. 12888–12900.

[108] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. "Diffusion-lm improves controllable text generation." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4328–4343.

[109] Yuanzhi Li, Colin Wei, and Tengyu Ma. "Towards explaining the regularization effect of initial large learning rate in training neural networks." In: *Advances in neural information processing systems* 32 (2019).

[110] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. "Competition-level code generation with alphacode." In: *Science* 378.6624 (2022), pp. 1092–1097.

[111] Zhiheng Li, Ivan Evtimov, Albert Gordo, Caner Hazirbas, Tal Hassner, Cristian Canton Ferrer, Chenliang Xu, and Mark Ibrahim. "A Whac-A-Mole Dilemma: Shortcuts Come in Multiples Where Mitigating One Amplifies Others." In: *arXiv preprint arXiv:2212.04825* (2022).

[112] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. "Just train twice: Improving group robustness without training group information." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6781–6792.

[113] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. "World model on million-length video and language with blockwise ringattention." In: *arXiv preprint arXiv:2402.08268* (2024).

[114] Hao Liu, Matei Zaharia, and Pieter Abbeel. "Ring attention with blockwise transformers for near-infinite context." In: *arXiv preprint arXiv:2310.01889* (2023).

[115] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep learning face attributes in the wild." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3730–3738.

[116] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization." In: *arXiv preprint arXiv:1711.05101* (2017).

[117] Aaron Lou, Chenlin Meng, and Stefano Ermon. "Discrete diffusion modeling by estimating the ratios of the data distribution." In: *arXiv preprint arXiv:2310.16834* (2023).

[118] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. "Exploring the limits of weakly supervised pretraining." In: *ECCV*. 2018.

[119] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. *Fine-Grained Visual Classification of Aircraft*. Tech. rep. 2013. arXiv: 1306.5151 [cs-cv].

[120] R Thomas McCoy, Ellie Pavlick, and Tal Linzen. "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference." In: *arXiv preprint arXiv:1902.01007* (2019).

[121] Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. "Inverse scaling: When bigger isn't better." In: *arXiv preprint arXiv:2306.09479* (2023).

[122] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. "Concrete score matching: Generalized score matching for discrete data." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 34532–34545.

[123] Elad Mezuman and Yair Weiss. "Learning about canonical views from internet image collections." In: *Advances in neural information processing systems* 25 (2012).

[124] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[125] John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. "Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization." In: *International conference on machine learning*. PMLR. 2021, pp. 7721–7735.

[126] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. "Never-ending learning." In: *Communications of the ACM* 61.5 (2018), pp. 103–115.

[127] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. "When does label smoothing help?" In: *Advances in neural information processing systems* 32 (2019).

[128] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. "Understanding the failure modes of out-of-distribution generalization." In: *arXiv preprint arXiv:2010.15775* (2020).

[129] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. "Learning from failure: De-biasing classifier from biased classifier." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20673–20684.

[130] Andrew Y. Ng and Michael I. Jordan. "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes." In: MIT Press, 2001.

[131] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes." In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. 2008.

[132] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. "Show your work: Scratchpads for intermediate computation with language models." In: (2021).

[133] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep metric learning via lifted structured feature embedding." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4004–4012.

[134] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." In: *preprint arXiv:1807.03748* (2018).

[135] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. "Dinov2: Learning robust visual features without supervision." In: *arXiv preprint arXiv:2304.07193* (2023).

[136] Rasmus Palm, Ulrich Paquet, and Ole Winther. "Recurrent relational networks." In: *Advances in neural information processing systems* 31 (2018).

[137] Arnaud Pannatier, Evann Courdier, and François Fleuret. "σ-GPTs: A New Approach to Autoregressive Models." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2024, pp. 143–159.

[138] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. "Cats and Dogs." In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

[139] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. "Cats and dogs." In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3498–3505.

[140] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. "Deep learning on a data diet: Finding important examples early in training." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20596–20607.

[141] William Peebles and Saining Xie. "Scalable Diffusion Models with Transformers." In: *arXiv preprint arXiv:2212.09748* (2022).

[142] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. "Gradient starvation: A learning proclivity in neural networks." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1256–1272.

[143] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. "Dreamfusion: Text-to-3d using 2d diffusion." In: *arXiv preprint arXiv:2209.14988* (2022).

[144] Mihir Prabhudesai, Tsung-Wei Ke, Alexander Cong Li, Deepak Pathak, and Katerina Fragkiadaki. "Diffusion-TTA: Test-time Adaptation of Discriminative Models via Generative Feedback." In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[145] Aahlad Manas Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. "Don't blame dataset shift! shortcut learning due to gradients and cross entropy." In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 71874–71910.

[146] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[147] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." In: ().

[148] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: (2020).

[149] Inioluwa Deborah Raji and Roel Dobbe. "Concrete problems in AI safety, revisited." In: *arXiv preprint arXiv:2401.10899* (2023).

[150] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. "Do ImageNet Classifiers Generalize to ImageNet?" In: *International Conference on Machine Learning*. 2019.

[151] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." In: *arXiv preprint arXiv:1908.10084* (2019).

[152] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier." In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[153] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. "Contrastive learning with hard negative samples." In: *arXiv preprint arXiv:2010.04592* (2020).

[154] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[155] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. "The elephant in the room." In: *arXiv preprint arXiv:1808.03305* (2018).

[156] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. "Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization." In: *arXiv preprint arXiv:2202.06856* (2022).

[157] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation." In: *arXiv preprint arXiv:2208.12242* (2022).

[158] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. *Learning internal representations by error propagation*. 1985.

[159] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization." In: *arXiv preprint arXiv:1911.08731* (2019).

[160] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. "An investigation of why overparameterization exacerbates spurious correlations." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8346–8356.

[161] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: https://openreview.net/forum?id=08Yk-n5l2Al.

[162] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. "Simple and effective masked diffusion language models." In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 130136–130184.

[163] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: *arXiv preprint arXiv:1910.01108* (2019).

[164] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. "Breeds: Benchmarks for subpopulation shift." In: *arXiv preprint arXiv:2008.04859* (2020).

[165] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[166] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. "LAION-5B: An open large-scale dataset for training next generation image-text models." In: *arXiv preprint arXiv:2210.08402* (2022).

[167] Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. "Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 6695–6706.

[168] Amrith Setlur, Don Dennis, Benjamin Eysenbach, Aditi Raghunathan, Chelsea Finn, Virginia Smith, and Sergey Levine. "Bitrate-constrained DRO: Beyond worst case robustness to unknown group shifts." In: *arXiv preprint arXiv:2302.02931* (2023).

[169] Burr Settles. "Active learning literature survey." In: (2009).

[170] Claude E Shannon. "A mathematical theory of communication." In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[171] Ruoqi Shen, Sébastien Bubeck, and Suriya Gunasekar. "Data augmentation as feature manipulation." In: *International conference on machine learning*. PMLR. 2022, pp. 19773–19808.

[172] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." In: *arXiv preprint arXiv:1712.01815* (2017).

[173] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics." In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[174] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." In: *arXiv preprint arXiv:2010.02502* (2020).

[175] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models." In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=St1giarCHLP.

[176] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. "Beyond neural scaling laws: beating power law scaling via data pruning." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19523–19536.

[177] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. "The implicit bias of gradient descent on separable data." In: *Journal of Machine Learning Research* 19.70 (2018), pp. 1–57.

[178] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[179] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. "Roformer: Enhanced transformer with rotary position embedding." In: *Neurocomputing* 568 (2024), p. 127063.

[180] Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting." In: *ACM Sigart Bulletin* 2.4 (1991), pp. 160–163.

[181] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. "Measuring Robustness to Natural Distribution Shifts in Image Classification." In: (2020). URL: https://arxiv.org/abs/2007.00644.

[182] David Tellez, Geert Litjens, Péter Bándi, Wouter Bulten, John-Melle Bokhorst, Francesco Ciompi, and Jeroen Van Der Laak. "Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology." In: *Medical image analysis* 58 (2019), p. 101544.

[183] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. "YFCC100M: The new data in multimedia research." In: *arXiv preprint arXiv:1503.01817* (2015).

[184] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. "Winoground: Probing vision and language models for visio-linguistic compositionality." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5238–5248.

[185] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive multiview coding." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer. 2020, pp. 776–794.

[186] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models." In: *arXiv preprint arXiv:2302.13971* (2023).

[187] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. "Neural autoregressive distribution estimation." In: *Journal of Machine Learning Research* 17.205 (2016), pp. 1–37.

[188] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[189] Hardik Vasa. *Google Images Download*. https://github.com/hardikvasa/google-images-download. 2015.

[190] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *NeurIPS*. 2017.

[191] Mitko Veta, Paul J Van Diest, Mehdi Jiwa, Shaimaa Al-Janabi, and Josien PW Pluim. "Mitosis counting in breast cancer: Object-level interobserver agreement and comparison to an automatic method." In: *PloS one* 11.8 (2016), e0161286.

[192] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[193] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. "Learning to efficiently sample from diffusion probabilistic models." In: *arXiv preprint arXiv:2106.03802* (2021).

[194] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models." In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[195] Christopher Williams and Carl Rasmussen. "Gaussian processes for regression." In: *Advances in neural information processing systems* 8 (1995).

[196] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. "Sampling matters in deep embedding learning." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2840–2848.

[197] Shirley Wu, Mert Yuksekgonul, Linjun Zhang, and James Zou. "Discover and cure: Concept-aware mitigation of spurious correlation." In: *International Conference on Machine Learning*. PMLR. 2023, pp. 37765–37786.

[198] Yutaro Yamada, Yingtian Tang, and Ilker Yildirim. "When are Lemons Purple? The Concept Association Bias of CLIP." In: *arXiv preprint arXiv:2212.12043* (2022).

[199] Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. "Multi-candidate speculative decoding." In: *arXiv preprint arXiv:2401.06706* (2024).

[200] Yuzhe Yang, Haoran Zhang, Dina Katabi, and Marzyeh Ghassemi. "Change is hard: A closer look at subpopulation shift." In: *arXiv preprint arXiv:2302.12254* (2023).

[201] Yang You, Igor Gitman, and Boris Ginsburg. "Large Batch Training of Convolutional Networks." In: *preprint arXiv:1708.03888* (2017).

[202] Alan Yuille and Daniel Kersten. "Vision as Bayesian inference: analysis by synthesis?" In: *Trends in cognitive sciences* 10.7 (2006), pp. 301–308.

[203] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features." In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6023–6032.

[204] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction." In: *arXiv preprint arXiv:2103.03230* (2021).

[205] John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric Karl Oermann. "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study." In: *PLoS medicine* 15.11 (2018), e1002683.

[206] Hang Zhang, Xin Li, and Lidong Bing. "Video-llama: An instruction-tuned audio-visual language model for video understanding." In: *arXiv preprint arXiv:2306.02858* (2023).

[207] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. "mixup: Beyond empirical risk minimization." In: *arXiv preprint arXiv:1710.09412* (2017).

[208] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. "Draft & verify: Lossless large language model acceleration via self-speculative decoding." In: *arXiv preprint arXiv:2309.08168* (2023).

[209] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. "Men also like shopping: Reducing gender bias amplification using corpus-level constraints." In: *arXiv preprint arXiv:1707.09457* (2017).

[210] Chenyu Zheng, Guoqiang Wu, Fan Bao, Yue Cao, Chongxuan Li, and Jun Zhu. "Revisiting discriminative vs. generative classifiers: Theory and implications." In: *International Conference on Machine Learning*. PMLR. 2023, pp. 42420–42477.

[211] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. "Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling." In: *arXiv preprint arXiv:2409.02908* (2024).

[212] Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. "What algorithms can transformers learn? a study in length generalization." In: *arXiv preprint arXiv:2310.16028* (2023).

[213] Roland S Zimmermann, Lukas Schott, Yang Song, Benjamin A Dunn, and David A Klindt. "Score-based generative classifiers." In: *arXiv preprint arXiv:2110.00473* (2021).

[214] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022.