Online Representation Learning on the Open Web

Ellis L. Brown, II

CMU-CS-23-1XX
May 2023

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee

Deepak Pathak	*Carnegie Mellon University*, Chair
Deva Ramanan	*Carnegie Mellon University*
Alexei A. Efros	*University of California, Berkeley*

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Copyright © 2023 Ellis L. Brown, II

May 4, 2023
DRAFT

Keywords: machine learning, deep learning, computer vision, representation learning, self-supervised learning, active learning, online learning, internet

To my parents, John and Amy, for their unwavering support and love,
and to Mara for putting up with all of the madness.

Abstract

Modern vision models typically rely on fine-tuning general-purpose models pre-trained on large, static datasets. These general-purpose models only capture the knowledge within their pre-training datasets, which are tiny, out-of-date snapshots of the Internet—where *billions* of images are uploaded daily.

We suggest in this thesis an alternate approach: rather than hoping our static datasets transfer to our desired tasks after large-scale pre-training, we propose dynamically utilizing the Internet to quickly train a small-scale model that does extremely well on the task at hand. Our approach, called **Internet Explorer**, explores the web in a self-supervised manner to progressively find relevant examples that improve performance on a desired target dataset. It cycles between searching for images on the Internet with text queries, self-supervised training on downloaded images, determining which images were useful, and prioritizing what to search for next.

We evaluate **Internet Explorer** across several datasets and show that it outperforms or matches CLIP oracle performance by using just a single GPU desktop to actively query the Internet for 30–40 hours.

The source code for this thesis document is available in open source form at:
https://github.com/ellisbrown/cmu-masters-thesis

Acknowledgments

TODO

May 4, 2023
DRAFT

Contents

May 4, 2023
DRAFT

List of Figures

May 4, 2023
DRAFT

List of Tables

May 4, 2023
DRAFT

List of Algorithms

May 4, 2023
DRAFT

May 4, 2023
DRAFT

Chapter 1

Introduction

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 [Den+09; He+16] and now would probably be CLIP [Rad+21]. 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 because our world is continually changing. Although the size of the pretraining datasets has grown from 1.2M [Den+09] to 400M [Sch+21] 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 thesis, we rethink the idea of a ***generic*** large-scale pretrained model and propose an alternate paradigm of training a rather small-scale but up-to-date model geared towards the ***specific*** downstream task of interest. To train such a model, we go beyond static datasets and *treat the Internet itself as a dynamic, open-ended dataset.* Unlike conventional datasets, which are expensive to increase 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.

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

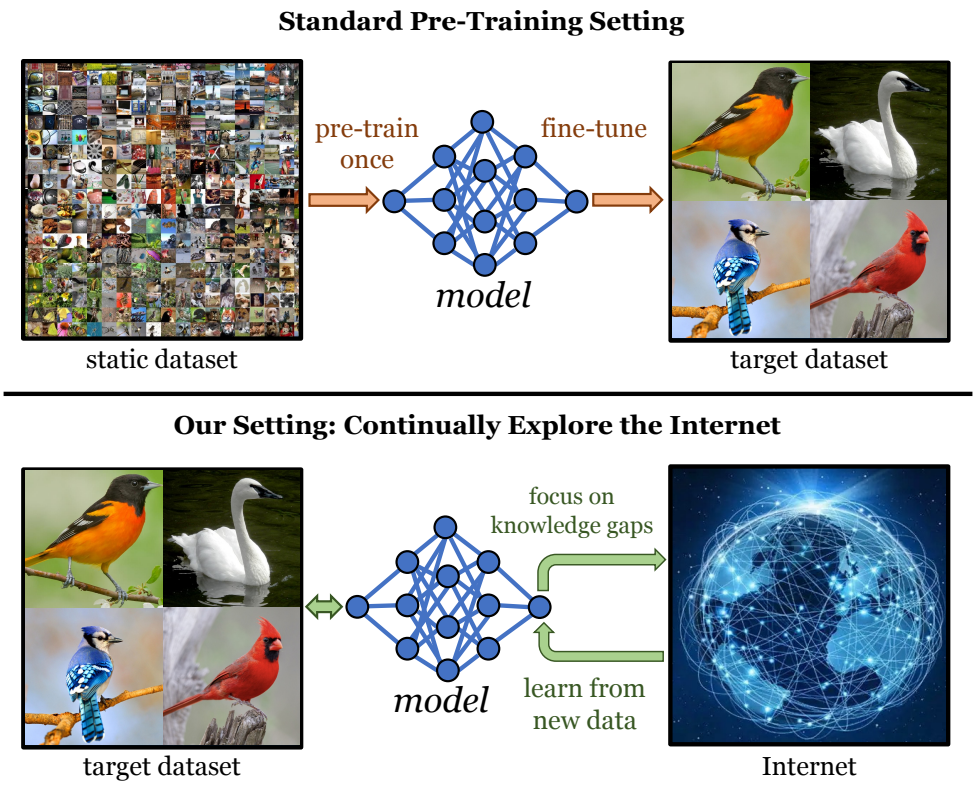

Figure 1.1: Given unlabeled data for a target task, our approach, Internet Explorer, searches the Internet to progressively find more and more relevant training data via self-supervised exploration.

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 searches the Internet using standard search engines to find relevant visual data that improve feature quality on a target dataset (see Figure 1.1). 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 2.1). We also bootstrap Internet Explorer using existing pre-trained models such as MoCov3 [He+20] and obtain a significant boost on the target datasets.

Our setting is different from active learning [Set09], 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 [Car+10] 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 [CSG13] builds on the dictionary developed by NELL to search visual data to 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. Recent work [Jia+21] follows a similar setting but searches a static dataset and not the Internet.

We evaluate Internet Explorer across 5 datasets, including 4 fine-grained datasets and PASCAL VOC. We search for relevant images using Google; however, the method is compatible with any text-based search engine or even a static dataset (see Section 3.2.4). We compare against several strong baselines, including CLIP, on downstream tasks. Note that CLIP acts as an oracle for our approach because it has likely already seen all or more queries that Internet Explorer makes. In most scenarios, Internet Explorer either outperforms or matches CLIP oracle using only a single 3090 GPU desktop machine that runs for 30–40 hours, makes over 10K progressively improving queries, and downloads over 1M relevant Internet images for each target dataset.

Chapter 2

Method

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, as well as others where the labels are not semantic (e.g., depth prediction or robotics). An overview of the Internet Explorer method is depicted in Figure 2.1 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, returns 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 [Vas15; Cli20]. We also try other search engines in Section 3.2.4.

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.

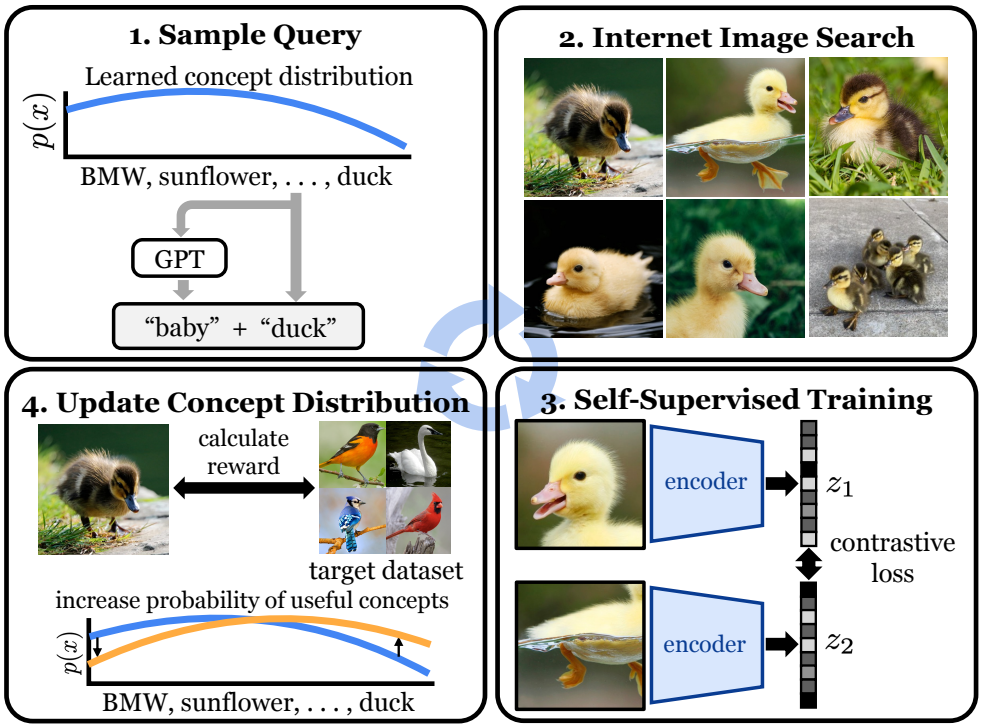

Figure 2.1: **Overview of Internet Explorer.** Our goal is to efficiently search the Internet for images that improve our performance on a target dataset. In each iteration, we first generate text queries by combining a concept sampled from a learned distribution with a GPT-generated descriptor (§2.1.2, 2.1.7). Next, we query search engines with the resulting phrase and download the top 100 image results (§2.1.1, 3.2.4). 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).

We draw our concepts from the WordNet hierarchy [Mil95], which consists of 146,347 noun lemmas. Not all of these lemmas are visual, but the vocabulary still covers an incredible range of topics (see examples in Section 5.1.1). 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 [WK21] with examples of descriptor-concept pairs (details and examples in Section 5.1.2). Finally, as shown in Step 1 of Figure 2.1, we simply concatenate the concept and descriptor. If our concept is "duck" and the GPT-generated descriptor is "baby," our search engine query will be "baby duck".

May 4, 2023
DRAFT

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 $\mathcal{V} = \{c_i\}_{i=1}^C$, where C is # concepts
concepts/itr M
query results/search Q
GPT-based concept → 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 $d_i \leftarrow \texttt{GPTDesc}(c_i)$ (§5.1.2)
    Image search $\{I_j^i\}_{j=1}^Q \leftarrow \texttt{SE}(d_i + c_i, Q)$ (§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: $\mathcal{B} \leftarrow \mathcal{B} \cup \texttt{Top50\%}(\mathcal{B}_{\text{new}}, r)$
  Predict all concept rewards $\mathbf{r}_{\text{concept}}$ from $\{r_{c_i}\}$ (§2.1.5)
  Update concept dist $p \leftarrow \texttt{CalcProb}(\mathbf{r}_{\text{concept}})$ (§2.1.7)
end for

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 [He+20; Che+20], non-contrastive [Gri+20; Zbo+21; BPL21; Car+21], masking-based [BDW21; He+22], or multimodal [Rad+21] approaches. For speed and stability reasons, we use the MoCo-v3 algorithm [CXH21], 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 [OLV18]:

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

where 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 [He+16] 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 [LEP22]. In each iteration of our method, we use MoCo-v3 to fine-tune 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 [KL17; FZ20; PGD21; Ily+22], 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" [Rob+20; SKP15; Oh +16; Har+17; Wu+17; Ge18] 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:

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

where $S_{\cos}$ is the cosine similarity. A previous metric for identifying relevant data [Jia+21] 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 3.2.6, 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.

2.1.5 Estimating Reward for Unseen Concepts

Concept Embeddings

Since our vocabulary contains hundreds of thousands of concepts, it would be inefficient to test whether *each* possible query yields relevant images. Luckily, we can estimate the quality of queries by using the observed rewards of the queries searched so far. Humans can do this effortlessly due to our understanding of what each concept means. To us, it is obvious that if querying "duck" yielded useful images for this dataset and "BMW" did not, then we should search for more images of animals and not cars. We want to give our method the same understanding of concept meaning, so we embed our 146,347 WordNet concepts into a 384-dimensional space using a pre-trained sentence similarity model [RG19]. We provide relevant context about each concept to the text embedding model by using additional information from WordNet in the following template:

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

For example, the text that is embedded for the concept "Chihuahua" is:

```
Chihuahua (toy dog): an old breed of tiny short-haired dog with protruding
eyes from Mexico held to antedate Aztec civilization.
```

Predicting Rewards

Now that we have a representation of each concept, we can estimate the reward for untried concepts by using the rewards of concepts that we have already searched for. We use Gaussian process regression (GPR) [WR95] 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. The covariance of any two variables $r(\mathbf{e}_i)$ and $r(\mathbf{e}_j)$ is determined by the kernel $k(\mathbf{e}_i, \mathbf{e}_j)$, which we set as the default RBF kernel $k(\mathbf{e}_i, \mathbf{e}_j) = \exp(\frac{-\|\mathbf{e}_i - \mathbf{e}_j\|_2}{2})$. Given the observed rewards for concepts $R_{obs} = \{r(\mathbf{e}_i)\}$, GPR calculates the posterior distribution over the rewards for an unobserved concept $\mathbf{e}'$, $P(r(\mathbf{e}') \mid \{r(\mathbf{e}_i)\} = R_{obs})$. Given that the joint distribution $P(\{r(\mathbf{e}_i)\}, r(\mathbf{e}'))$ is Gaussian, the posterior is also Gaussian with mean $\mu(\mathbf{e}')$ and variance $\sigma(\mathbf{e}')^2$. The locality provided by the RBF kernel enables reasonable reward predictions, and having a distribution over rewards instead of a point estimate allows us to explore potentially good concepts. We encourage exploration by setting the score of unobserved concepts to $\mu(\mathbf{e}_i) + \sigma(\mathbf{e}_i)$.

May 4, 2023
DRAFT

A practical issue is that GPR does not scale well to large numbers of points, and fitting the model after each iteration becomes prohibitively slow. To address this, we use GPR for the first 10 iterations (256 queries $\times$ 100 images $\times$ 10 iterations $\approx$ 256,000 points), and then switch to a simple ridge regression model that predicts solely the mean rewards $\mu(\mathbf{e}_i)$—and thus we eliminate the exploration term $\sigma(\mathbf{e}_i)$ from the reward prediction. This is a good tradeoff because we have a lot of data by this point, and we don't need to explore as much anymore. The number of iterations at which we switch to ridge regression is a hyperparameter that can be tuned to balance exploration and exploitation + speed. We found that 10 iterations to work well in practice.

2.1.6 Provable speedup in relevant query identification

We now show that our method can provably speed up the time to identify all relevant concepts in a dataset. Assume that our vocabulary of n concepts contains $c \cdot s \ll n$ relevant concepts, which are partitioned into c disjoint clusters of size s. We want to discover every relevant concept by sampling concepts uniformly at random (with replacement) to test. Assume that sampling a concept conclusively tells us whether it is relevant. Furthermore, assume that we could optionally use a Gaussian process regression model which, if we've sampled a relevant concept, tells us that all the concepts in its cluster are also relevant.

Lemma 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 $\frac{T_{base}}{T_{GPR}} \approx s \log s$.*

We find that in practical settings (e.g., the Pets example analyzed in Fig. 3.3), 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 $9 \times 256 = 1792$ queries it took to discover both cats and dogs in the Pets 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 [Sut91]. 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

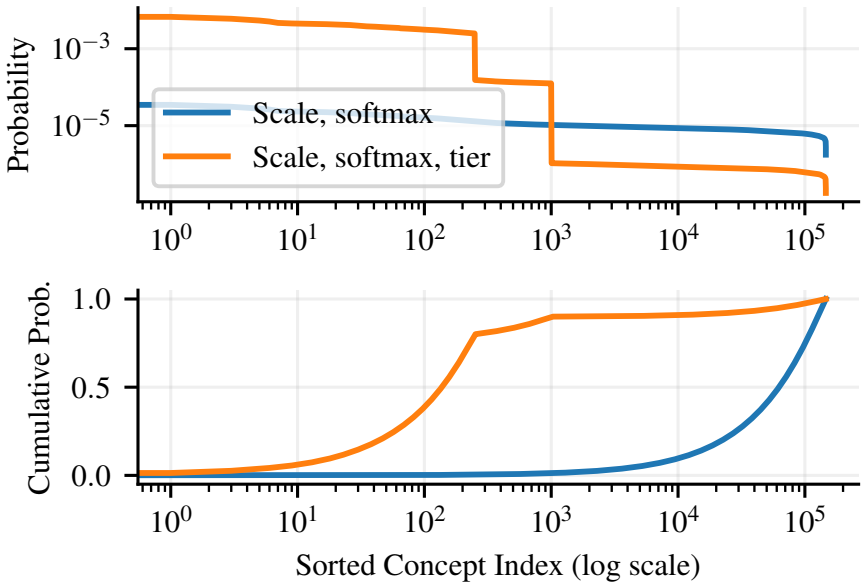

Figure 2.2: **Learned concept sampling distribution.** Given estimated scores for each of the 146,347 concepts, we need to choose how often to sample each one in order to balance exploration and exploitation. **Top:** we scale our scores to a desired temperature, then take the softmax to obtain a distribution 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 1000 concepts are only sampled a tiny fraction of the time without tiering.

$\Delta_0, \ldots, \Delta_{n-1}$ such that $\sum_i \Delta_i = 1$, tiering computes a new distribution:

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

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 2.2 shows that tiering the scaled softmax distribution samples frequently enough from the top concepts while a vanilla scaled softmax distribution does not. Note that the untiered distribution would sample the top 1000 concepts only $\approx 0.1\%$ of the time, while the tiered distribution samples them 90% of the time.

May 4, 2023
DRAFT

Chapter 3

Experimental Setting and Results

This section provides an overview of the experimental setting used to evaluate Internet Explorer, as well as detailed results and analysis of our experiments in these settings.

3.1 Experimental Setting

3.1.1 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.

The lack of label knowledge makes this setting challenging, but also widely applicable. Our method is forced to explore the space of concepts to discover what is useful for the target dataset, without any prior knowledge of what might be useful.

3.1.2 Label Set-guided Exploration

In practice, we may sometimes know the set of labels for our task even if we do not have image-label pairs (*i.e.*, the English names of the classes). For example, we may know that our data contains images of "Golden Retrievers," "Maine Coon," *etc.*, even if we do not have any images labeled with these names. Knowing the label set greatly accelerates learning on the Internet, because it acts as a strong prior on what could be useful for our target dataset and allows us to focus our exploration on the subset of the vocabulary that is most likely to 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. Restricting our vocabulary to a semantically relevant subset also strengthens our baselines by ensuring that they only search for potentially useful concepts.

May 4, 2023
DRAFT

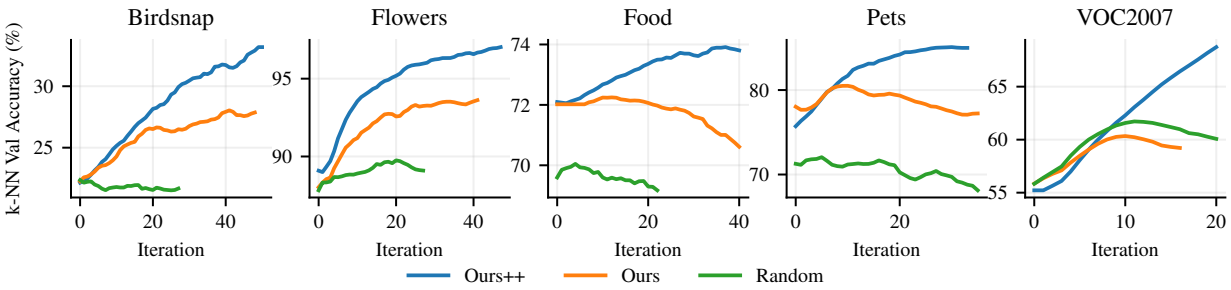

Figure 3.1: **Learning curves in self-supervised setting.** k-NN validation accuracy improves across iterations on each target dataset. Without using any labels, Internet Explorer identifies and focuses on relevant concepts for each target dataset. This allows it to find more useful data than the baseline that searches for random concepts. Adding GPT-generated descriptors (Ours++) further improves performance by enabling Internet Explorer to generate diverse views of useful concepts.

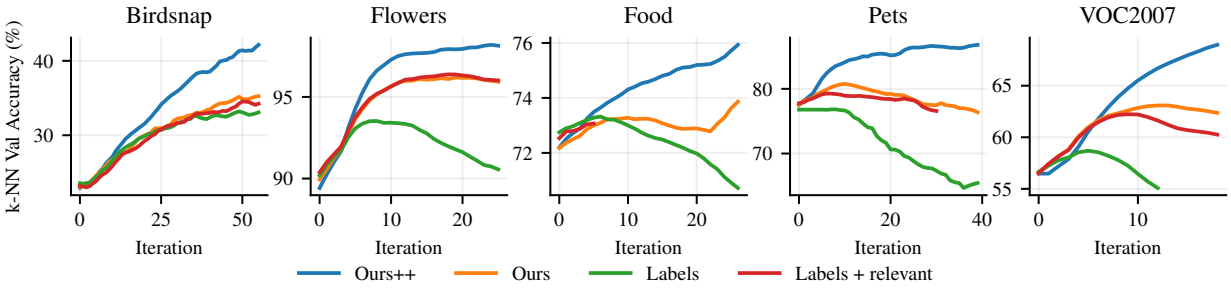

Figure 3.2: **Learning curves in label set-guided setting.** Using knowledge of the label set improves the performance of all methods. Internet Explorer (Ours) outperforms the baselines on all datasets, and adding GPT-generated descriptors (Ours++) further improves performance.

We compare 4 methods in this setting:

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.

3.1.3 Datasets

We evaluate Internet Explorer on 4 popular small-scale fine-grained classification datasets: Birdsnap [Ber+14], Flowers-102 [NZ08], Food101 [BGG14], and Oxford-IIT Pets [Par+12]. We also evaluate on Pascal VOC 2007 (Classification) [Eve+10]—a coarse-grained multi-label classification task consisting of only 2,040 training examples, making it ideal for

May 4, 2023
DRAFT

Model	Birdsnap	Flowers	Food	Pets	VOC2007	Images	GPU hrs.
Fixed dataset, lang. supervision							
CLIP ResNet-50 (**oracle**)	57.1	96.0	**86.4**	88.4	**86.7**	400×10^6	4,000
Fixed dataset, self-supervised							
MoCo-v3 (ImageNet pre-train)	26.8	83.2	70.5	79.6	–	1.2×10^6	72
MoCo-v3 (ImageNet + target)	39.9	94.6	78.3	85.3	58.0†	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)	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)	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)	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)	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)	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)	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)	2.2×10^6	84 + 40

Table 3.1: **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, we do not do ImageNet pre-training because ImageNet is too close to VOC2007.

testing whether Internet Explorer can efficiently find relevant useful data. We do not target large-scale datasets like ImageNet [Den+09] because they already contain over a million human-curated Internet images.

3.1.4 Evaluation Metrics

We compare the representation quality of our models using two metrics: k-nearest neighbors (k-NN) accuracy and linear probe accuracy. To measure k-NN accuracy, we use our ResNet-50 to encode each dataset's training and test sets. Then, for each test example, we find its k nearest neighbors in representation space and use the mode of its neighbors' labels as the prediction. We use k-NN accuracy to plot learning curves of model performance after every iteration, since it is easy to quickly compute.

To compute the linear probe accuracy, we first select the best-performing model over all iterations on the validation set according to the k-NN accuracy. Then, we learn a linear head on top of these learned representations by minimizing the cross-entropy loss. We tune the weight decay parameter in the logscale range $(10^{-6}, 10^6)$, as is done in [Rad+21], using `scikit-learn` [Ped+11] with Brent's method [Bre73].

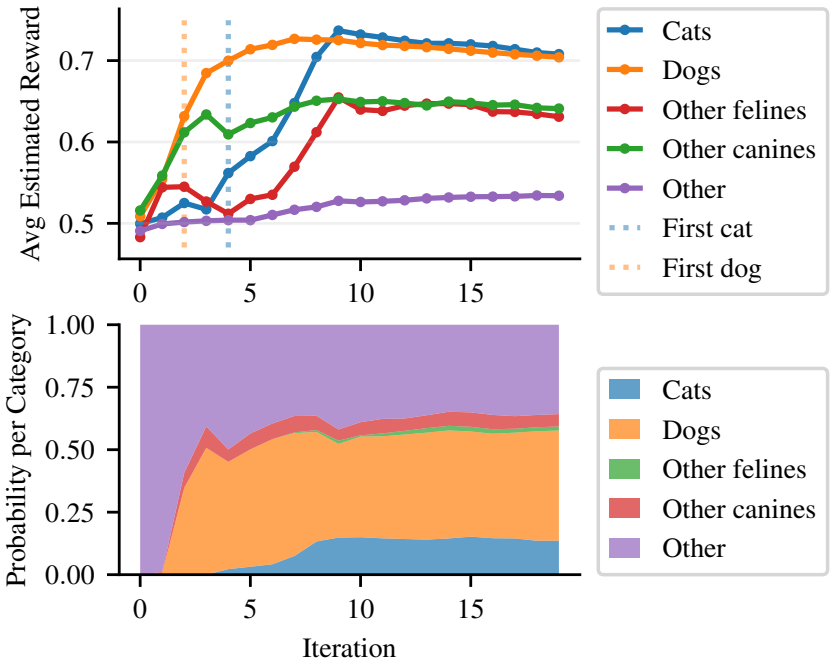

Figure 3.3: **Self-supervised concept discovery on Pets dataset.** When targeting the Pets dataset, self-supervised Internet Explorer quickly estimates high reward for concepts from the cat category (82 concepts) and dog category (246 concepts). It is also able to identify felines that are not cats (e.g., tigers) and canines that are not dogs (e.g., wolves), 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.

3.2 Results and Analysis

3.2.1 Self-supervised Results

Figure 3.1 shows how 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 [MW12; CG15]. Table 3.1 shows that our method significantly improves on the starting MoCo-v3 (ImageNet + target) checkpoint and can outperform a CLIP [Rad+21] model of the same size while using much less compute and data. This is impressive as CLIP can be thought of as 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.

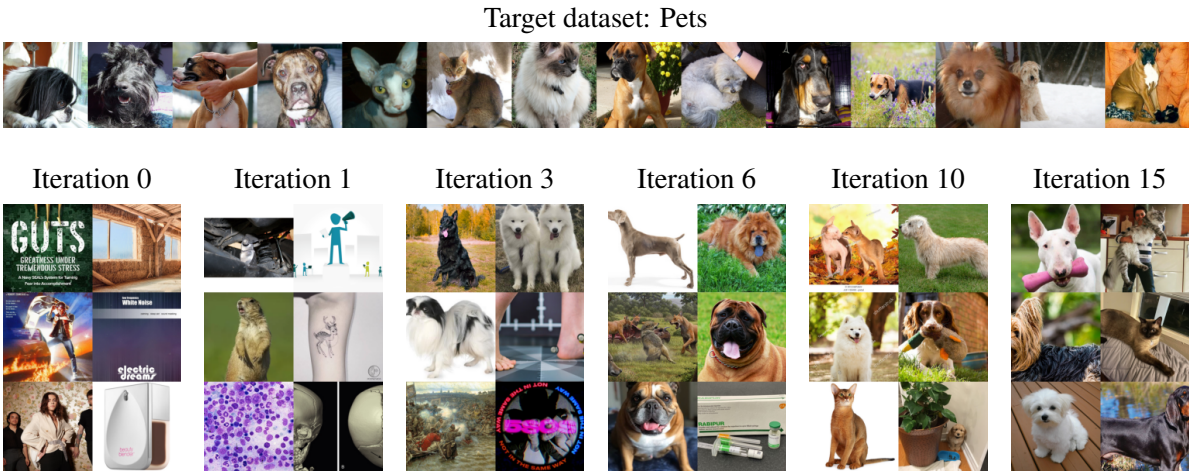

Figure 3.4: **Progression of downloaded images across training. Top:** samples of Oxford-IIIT Pets images. **Bottom:** samples of images queried by Internet Explorer across iterations. As the method learns, it makes queries that are progressively more relevant to the target dataset.

3.2.2 Self-supervised Exploration Behavior

Figure 3.3 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 3.3 (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 3.4 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. Iteration #0 contains mostly useless data, like graphics or screenshots, but Pets-relevant images already make up most of the downloads by iteration #3.

3.2.3 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 3/5 datasets and approaches its performance in the other 2, using just 2.5% of the time and 0.5% the data—as can be seen in Table 3.1. The fact that Ours++ can match/outperform CLIP is especially impressive, since CLIP has access to up to 20k Bing image search results for each WordNet lemma (in addition to other queries). k-NN accuracy learning curves for the label set-guided setting are shown in Figure 3.2.

3.2.4 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.

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 3.2: **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.

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 [Sch+22], which consists of 5.85 billion noisy image-caption pairs. We filter the dataset to only include samples with English captions and images with at least 512^2 pixels. 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 use a pre-trained text embedding model [RG19] to compute 384-dimensional text embeddings for each caption. Then, we use Faiss [JDJ19] to compute a fast, approximate nearest-neighbors lookup index. Querying our custom search engine finds 100 image results in less than a second. Figure 3.5 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 3.7 shows that each data source has its own tendencies. For the "spaghetti bolognese" query, Google Images is biased [MW12; CG15]

May 4, 2023
DRAFT

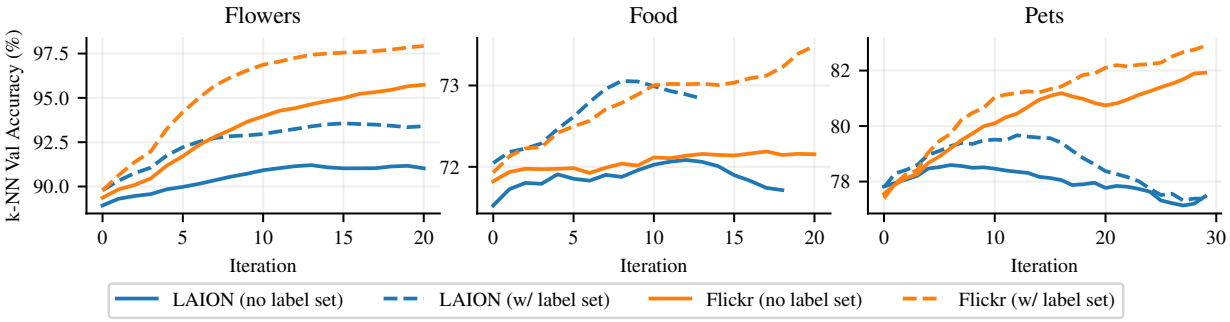

Figure 3.6: **Learning from Flickr and LAION-5B.** Even with the noisy search results returned by Flickr and LAION, Internet Explorer still continuously improves performance.

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 3.6 and Table 3.2 show 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. Overall, these results are proof that Internet Explorer can effectively utilize any window into the Internet's vast ocean of image data.

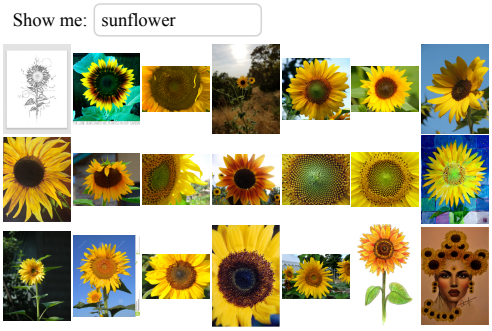

Figure 3.5: **Our custom LAION-5B search engine.** We build a custom text-to-image search engine that finds images within the LAION-5B dataset by doing nearest neighbor search in text embedding space. This uses no image features whatsoever.

3.2.5 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) [Buc21] 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 3.3. 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

	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 3.3: **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% of test images leaked. 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 dataset'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*. Overall, Internet Explorer's increase in accuracy cannot be explained by test set leakage, so it must be improving performance through better feature learning and generalization.

test set. On the other datasets, the amount leaked ranges from 0.003% to 0.6% of the test set. Additionally, we only perform 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.

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 images for 5 randomly selected test set images from the Flowers, Food, and Pets datasets in Figure 3.8. 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.

3.2.6 Effect of image reward type

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 3.4 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 5.5 in the appendix. 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 text because these images 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.

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

Table 3.4: **Ablation on type of image reward.** MoCo loss does not identify relevant concepts, and $k = 1$ similarity is too noisy to identify useful concepts.

May 4, 2023
DRAFT

Food101 dataset: “Spaghetti Bolognese”

Google Images: “Spaghetti Bolognese”

Flickr: “Spaghetti Bolognese”

LAION-5B: “Spaghetti Bolognese”

Figure 3.7: **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.

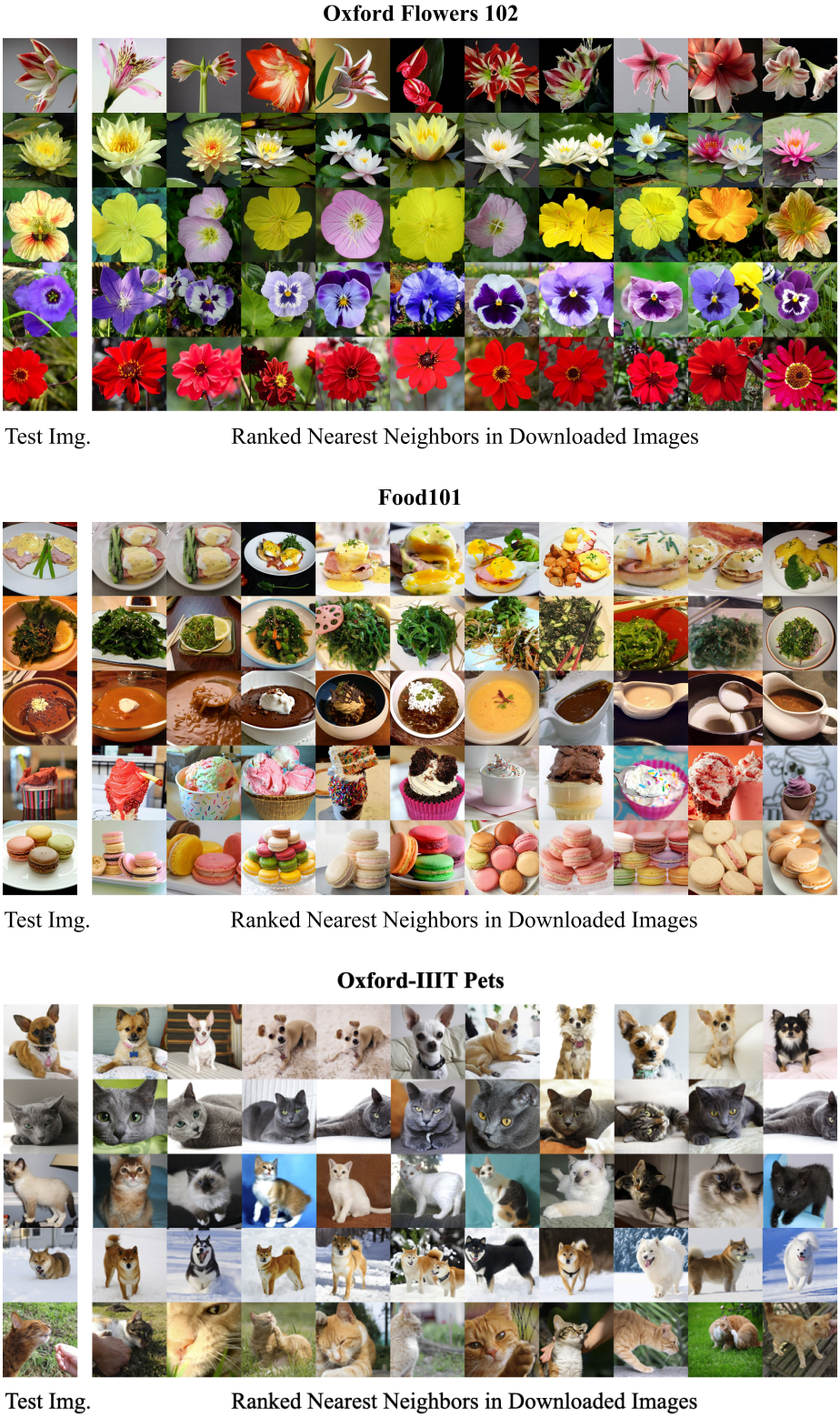

Figure 3.8: **Top-10 most similar online images**. The left column shows randomly chosen test set images from each dataset, and the right block shows the 10 most similar images in the downloaded data for each test image, ranked left to right.

May 4, 2023
DRAFT

Chapter 4

Conclusions and Future Directions

We show that interactively exploring the Internet is an efficient source of highly relevant training data—if one knows how to search for it. In just 30–40 hours of training on a single GPU, Internet Explorer either significantly outperforms or closely matches the performance of compute-heavy *oracle* models like CLIP [Rad+21] trained on static datasets, as well as strong baselines that search the Internet in an undirected manner.

May 4, 2023
DRAFT

May 4, 2023
DRAFT

Chapter 5

Appendix

5.1 Method Details

5.1.1 WordNet Lemmas

We draw our concepts from the WordNet hierarchy [Mil95], 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", "friendly", "Principe", "bottle green", "glycerol trimargarate", "water-shield", "San Joaquin River", "woodsman", "pin".

5.1.2 GPT-J Descriptor Prompting

We use GPT-J-6B [WK21], 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".
```

5.1.3 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 [Rad+21]. Doing so would expand our vocabulary to roughly 500,000 total concepts.

5.1.4 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

$$\text{SMR} = \frac{\max_i r(c_i) - \min_i r(c_i)}{\tau} \tag{5.1}$$

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:

$$\frac{\max_i p(c_i)}{\min_i p(c_i)} = \frac{\max_i \exp(r(c_i)/\tau)}{\min_i \exp(r(c_i)/\tau)} \tag{5.2}$$

$$= \exp\left(\frac{\max_i r(c_i) - \min_i r(c_i)}{\tau}\right) \tag{5.3}$$

$$= \exp(\text{SMR}) \tag{5.4}$$

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.

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

Table 5.1: **Target Dataset "Category"**.

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 [RG19] 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 5.1) in place of the hypernym:

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

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.

5.1.5 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.

5.1.6 Hyperparameters

Table 5.2 shows our hyperparameter values, which are shared across datasets. We perform minimal hyperparameter tuning and copy most of the values from the MoCo-v3 [CXH21] ResNet-50 configuration. We will also release our code upon acceptance, which we hope will clarify any remaining implementation details and make it easy for the community to reproduce and build on our work.

May 4, 2023
DRAFT

Hyperparameter	Value
Architecture	Resnet-50 [He+16]
Optimizer	LARS [YGG17]
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 5.2: **Internet Explorer hyperparameters**.

5.1.7 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.

5.2 Proof of Lemma 1

Here, we prove Lemma 1 from Section 2.1.6, which we repeat below:

Assume that our vocabulary of n concepts contains $c \cdot s \ll n$ relevant concepts, which are partitioned into c disjoint clusters of size s. We want to discover every relevant concept by sampling concepts uniformly at random (with replacement) to test. Assume that sampling a concept conclusively tells us whether it is relevant. Furthermore, assume that we could optionally use a Gaussian process regression model which, if we've sampled a relevant concept, tells us that all the concepts in its cluster are also relevant.

Lemma 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.

$$T_{base} = \sum_{i=1}^{cs} t_i \tag{5.5}$$
$$= \sum_{i=1}^{cs} \frac{1}{p_i} \tag{5.6}$$
$$= \sum_{i=1}^{cs} \frac{n}{cs+1-i} \tag{5.7}$$
$$= n \sum_{i=1}^{cs} \frac{1}{cs+1-i} \tag{5.8}$$
$$= nH_{cs} \tag{5.9}$$

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.

$$T_{GPR} = \sum_{i=1}^{c} t_i \tag{5.10}$$
$$= \sum_{i=1}^{c} \frac{1}{p_i} \tag{5.11}$$
$$= \sum_{i=1}^{c} \frac{n}{s(c+1-i)} \tag{5.12}$$
$$= \frac{n}{s} \sum_{i=1}^{c} \frac{1}{c+1-i} \tag{5.13}$$
$$= \frac{nH_c}{s} \tag{5.14}$$

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

5.3 Progression of downloaded images

Just as Fig. 4 in the main paper showed how Internet Explorer progressively discovers useful data when targeting the Pets dataset, Figure 5.1, Figure 5.2, Figure 5.3, and Figure 5.4 show the progression of downloaded images when targeting Birdsnap, Flowers, Food, and VOC respectively. Note that this analysis is in the self-supervised setting, without any knowledge of the label set.

5.4 Additional Figures

May 4, 2023
DRAFT

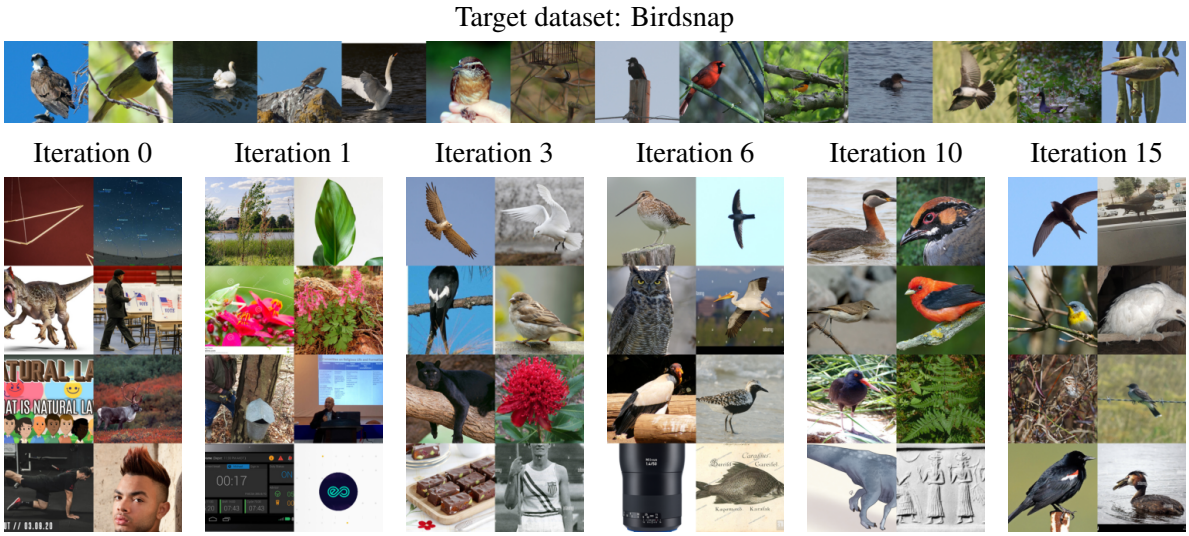

Figure 5.1: **Progression of downloaded Birdsnap images.** This corresponds to Ours++ without using label set information.

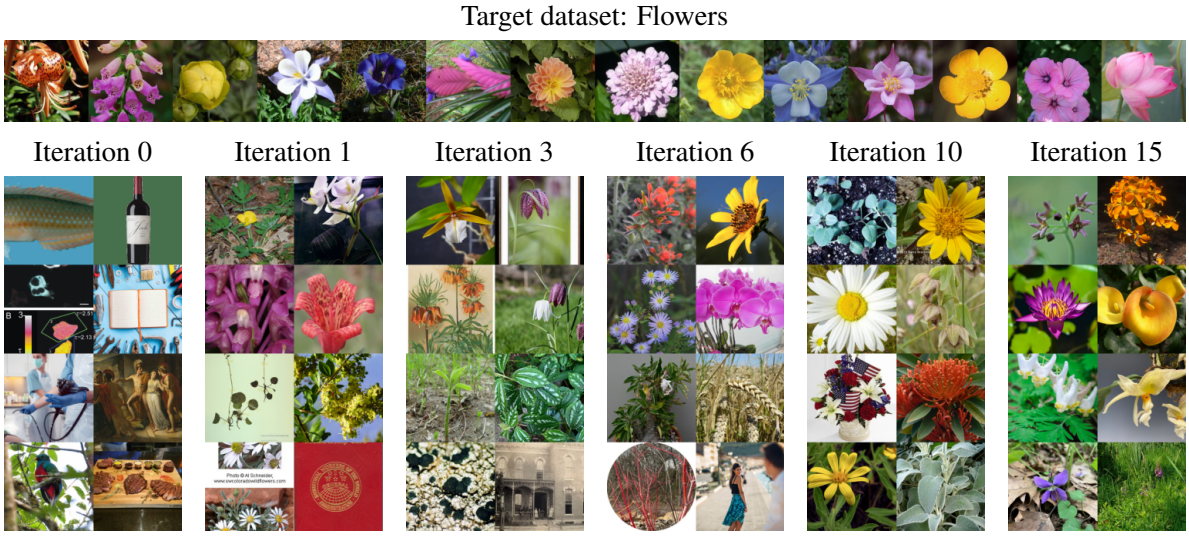

Figure 5.2: **Progression of downloaded Flowers images.** This corresponds to Ours++ without using label set information.

Figure 5.3: **Progression of downloaded Food images.** This corresponds to Ours++ without using label set information.

Figure 5.4: **Progression of downloaded VOC2007 images.** This corresponds to Ours++ without using label set information.

Figure 5.5: **Top images preferred by different rewards.** We show the top 5 downloaded images ranked by 3 possible image rewards on the Food dataset. 15-NN (ours) prefers a variety of food images, whereas MoCo prefers noisy images out of the training distribution. 1-NN is thrown off by outliers in the Food dataset and thus prefers black images, text, and zebras.

May 4, 2023
DRAFT

Bibliography

This bibliography contains 49 references.

[BDW21] Hangbo Bao, Li Dong, and Furu Wei. "Beit: Bert pre-training of image transformers". In: *arXiv preprint arXiv:2106.08254* (2021).

[Ber+14] 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.

[BGG14] 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.

[BPL21] Adrien Bardes, Jean Ponce, and Yann LeCun. "Vicreg: Variance-invariance-covariance regularization for self-supervised learning". In: *arXiv preprint arXiv:2105.04906* (2021).

[Bre73] Richard P. Brent. *Algorithms for Minimization without Derivatives*. 1st. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

[Buc21] Johannes Buchner. *imagehash (fork)*. https://github.com/JohannesBuchner/imagehash. 2021.

[Car+10] 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.

[Car+21] 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.

[CG15] 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.

[Che+20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". In: *preprint arXiv:2002.05709* (2020).

[Cli20] Joe Clinton. *Google Images Download (fork)*. https://github.com/Joeclinton1/google-images-download. 2020.

[CSG13] 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.

[CXH21] 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.

[Den+09] 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.

[Eve+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge". In: *IJCV* (2010).

[FZ20] 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.

[Ge18] Weifeng Ge. "Deep metric learning with hierarchical triplet loss". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 269–285.

[Gri+20] Jean-Bastien Grill et al. "Bootstrap your own latent: A new approach to self-supervised learning". In: *NeurIPS*. 2020.

[Har+17] 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.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR*. 2016.

[He+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning". In: *CVPR*. 2020.

[He+22] 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.

[Ily+22] 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).

[JDJ19] 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.

[Jia+21] 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.

[KL17] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894.

[LEP22] 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.

[Mil95] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[MW12] Elad Mezuman and Yair Weiss. "Learning about canonical views from internet image collections". In: *Advances in neural information processing systems* 25 (2012).

[NZ08] 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.

[Oh +16] 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.

[OLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *preprint arXiv:1807.03748* (2018).

[Par+12] 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.

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PGD21] 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.

[Rad+21] 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.

[RG19] Nils Reimers and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084* (2019).

[Rob+20] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. "Contrastive learning with hard negative samples". In: *arXiv preprint arXiv:2010.04592* (2020).

[Sch+21] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs". In: *arXiv preprint arXiv:2111.02114* (2021).

[Sch+22] 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).

[Set09] Burr Settles. "Active learning literature survey". In: (2009).

[SKP15] 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.

[Sut91] Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM Sigart Bulletin* 2.4 (1991), pp. 160–163.

[Vas15] Hardik Vasa. *Google Images Download*. https://github.com/hardikvasa/google-images-download. 2015.

[WK21] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[WR95] Christopher Williams and Carl Rasmussen. "Gaussian processes for regression". In: *Advances in neural information processing systems* 8 (1995).

[Wu+17] 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.

[YGG17] Yang You, Igor Gitman, and Boris Ginsburg. "Large Batch Training of Convolutional Networks". In: *preprint arXiv:1708.03888* (2017).

[Zbo+21] 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).