Internet Explorer: Targeted Representation Learning on the Open Web

Alexander C. Li [*1] **Ellis Brown** [*1] **Alexei A. Efros** [2] **Deepak Pathak** [1]

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 These general-purpose models only capture the knowledge within their pre-training datasets, which are tiny, out-of-date snapshots of the Internet—where billions of images are uploaded each day. We suggest 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. Results, visualizations, videos, and code on our website: internet-explorer-ssl.github.io/

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 (Deng et al., 2009; He et al., 2016) and now would probably be CLIP (Radford et al., 2021). 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 (Deng et al., 2009) to 5B (Schuhmann et al., 2022) 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.

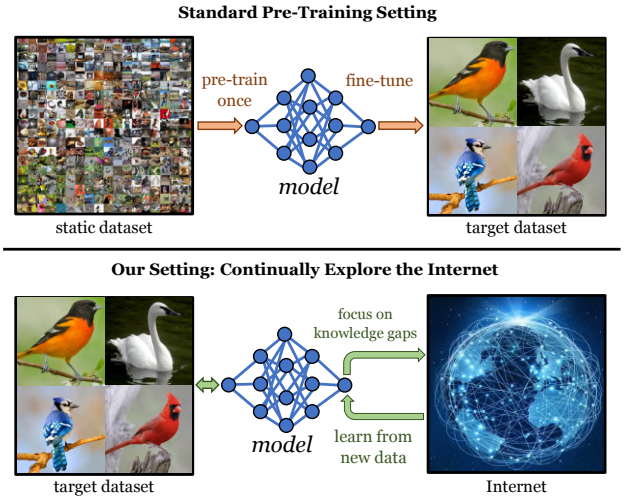


Figure 1. Given unlabeled data for a target task, our approach, Internet Explorer, searches the Internet to progressively find more and more relevant training data via self-supervised exploration.

In this paper, 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

[*]Equal contribution [1]Carnegie Mellon University [2]University of California, Berkeley. Correspondence to: Alexander Li <alexanderli@cmu.edu>, Ellis Brown <ellisbrown@cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

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.

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 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). We also bootstrap Internet Explorer using existing pre-trained models such as MoCo-v3 (He et al., 2020) and obtain a significant boost on the target datasets.

Our setting is different from active learning (Settles, 2009), 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 (Carlson et al., 2010) 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 (Chen et al., 2013) 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. Recent work (Jiang et al., 2021) follows a similar setting but searches a static dataset and not the Internet.

We evaluate Internet Explorer across 7 datasets, including 4 fine-grained datasets, PASCAL VOC, ImageNet-100, and FMoW-WILDS. 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 4.5). 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 the 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.

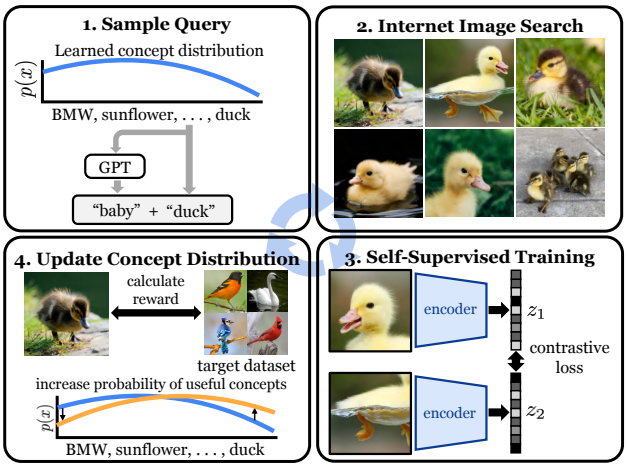


Figure 2. **Overview of Internet Explorer.** Our goal is to efficiently search the Internet for images that improve our performance on a target dataset. In each iteration, we first generate text queries by combining a concept sampled from a learned distribution with a GPT-generated descriptor (§2.2, §2.7). Next, we query search engines with the resulting phrase and download the top 100 image results (§2.1, 4.5). We add these images to the set of previously downloaded images and perform self-supervised training on the combined dataset (§2.3). 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 (§2.4, §2.5).

2. 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 and described in Algorithm 1.

2.1. 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 (Vasa, 2015; Clinton, 2020). We also try other search engines in Section 4.5.

2.2. 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 (Miller, 1995), 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 Appendix C.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 2.4 for details on estimating rewards and Section 2.7 for the distribution). Given a sampled concept, we can generate a descriptor by prompting a GPT-J language model (Wang & Komatsuzaki, 2021) with examples of descriptor-concept pairs (details in Appendix C.2). Finally, as shown in Step 1 of Figure 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."

2.3. 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 et al., 2020; Chen et al., 2020), non-contrastive (Grill et al., 2020; Zbontar et al., 2021; Bardes et al., 2021; Caron et al., 2021), masking-based (Bao et al., 2021; He et al., 2022), or multimodal (Radford et al., 2021) approaches. For speed and stability reasons, we use the MoCo-v3 algorithm (Chen et al., 2021), 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 (Oord et al., 2018):

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)} \quad (1)$$

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 (He et al., 2016) 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 (Li et al., 2022). 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.

2.4. 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 (Koh & Liang, 2017; Feldman & Zhang, 2020; Paul et al., 2021; Ilyas et al., 2022), 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. 1 is larger when the negative examples $\{k^-\}$ are closer to q. These "hard negatives" (Robinson et al., 2020; Schroff et al., 2015; Oh Song et al., 2016; Harwood et al., 2017; Wu et al., 2017; Ge, 2018) 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

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 $c_i \sim p(\mathcal{V})$ (§2.2)
    Sample descriptor $d_i \leftarrow \texttt{GPTDesc}(c_i)$ (§C.2)
    Image search $\{I_j^i\}_{j=1}^Q \leftarrow \texttt{SE}(d_i + c_i, Q)$ (§2.1)
    Calc. reward $r_{c_i} \leftarrow \frac{1}{Q}\sum_{j=1}^Q r(f, \mathcal{D}, I_j^i)$ (§2.4)
  end for
  $\mathcal{B}_{\text{new}} = \{I_j^1\}_{j=1}^Q \cup \cdots \cup \{I_j^M\}_{j=1}^Q$
  SSL training: $\mathbb{A}(f, \mathcal{D} \cup \mathcal{B} \cup \mathcal{B}_{\text{new}})$ (§2.3)
  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 $\{r_{c_i}\}$ (§2.5)
  Update concept dist $p \leftarrow \texttt{CalcProb}(\mathbf{r}_{\text{concept}})$ (§2.7)
end for

reward is calculated:

$$r(f_k, \mathcal{D}, y) = \max_{\substack{I\subset\{1,\ldots,N\};\\ |I|=k}} \frac{1}{k}\sum_{i\in I} S_{\cos}(f_k(x_i), f_k(y)) \quad (2)$$

where $S_{\cos}$ is the cosine similarity. A previous metric for identifying relevant data (Jiang et al., 2021) 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 4.6, 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.

2.5. 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 (Reimers & Gurevych, 2019). We provide relevant context about concepts to the text embedding model using the following template:

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

For example,

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

We use Gaussian process regression (GPR) (Williams & Rasmussen, 1995) 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 $\mathbf{e}'$, $P(r(\mathbf{e}')|\{r(\mathbf{e}_i)\} = R_{obs})$. 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 $\mu(\mathbf{e}_i) + \sigma(\mathbf{e}_i)$.

2.6. 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

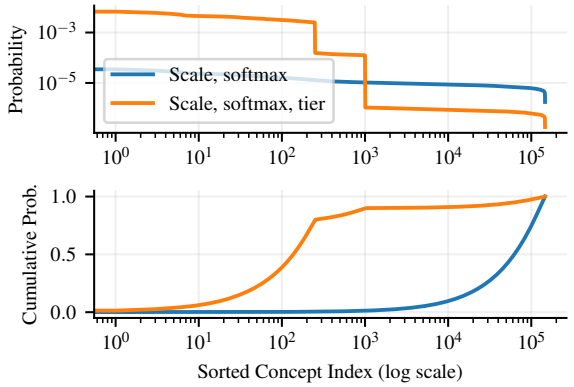


Figure 3. **Learned concept sampling distribution.** Given estimated scores for each of the $146,347$ concepts, we need to choose how often to sample each one in order to balance exploration and exploitation. **Top:** we scale our scores to a desired temperature, then take the softmax to obtain a distribution 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 1000 concepts are only sampled a tiny fraction of the time without tiering.

us that all concepts in its cluster are also relevant. Then, Lemma 2.1 shows that the Gaussian process drastically reduces the time required to identify all relevant concepts.

Lemma 2.1. *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 $\frac{T_{base}}{T_{GPR}} \approx s \log s$.*

The proof is in Appendix D. For our vocabulary and target datasets, $s \approx 100$. This shows that a predictive model like GPR is crucial for quickly identifying all useful concepts.

2.7. 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 (Sutton, 1991). 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:

$$p_i^{\text{tier}} = \Delta_j \frac{p_i}{\sum_{k=T_j}^{T_{j+1}} p_k} \quad \text{for } j \text{ s.t. } T_j \leq i < T_{j+1} \qquad (3)$$

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 3 shows that tiering the scaled softmax distribution samples frequently enough from the top concepts while a vanilla scaled softmax distribution does not.

3. Experimental Setting

3.1. 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.

3.2. 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.

3.3. Datasets and Metrics

We evaluate Internet Explorer on 4 popular small-scale fine-grained classification datasets: Birdsnap (Berg et al.,

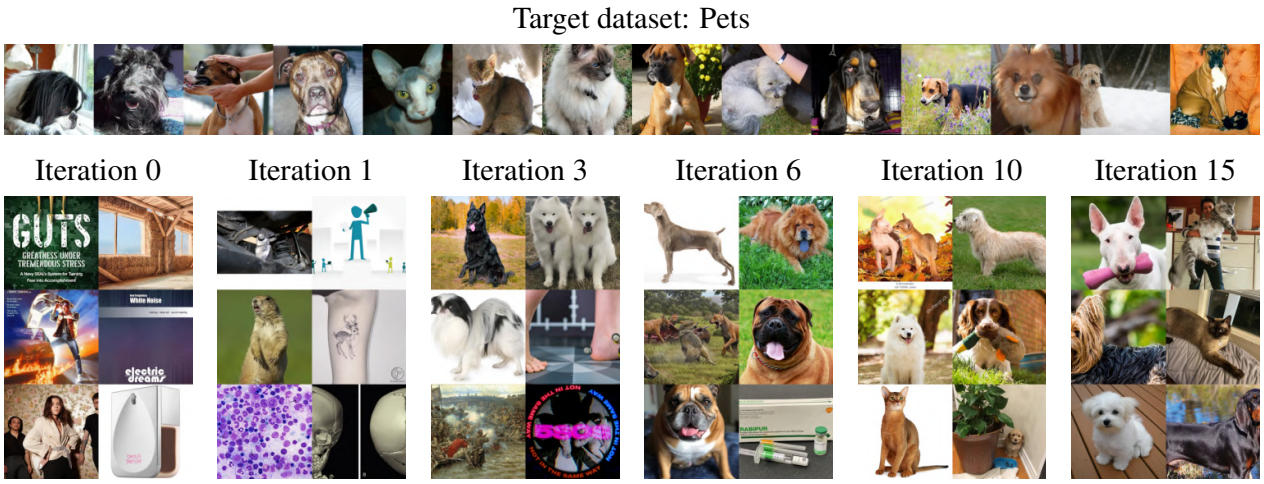


Figure 4. **Progression of downloaded images across training. Top:** samples of Oxford-IIIT Pets images. **Bottom:** samples of images queried by Internet Explorer across iterations. As it learns, it makes queries that are progressively more relevant to the target dataset.

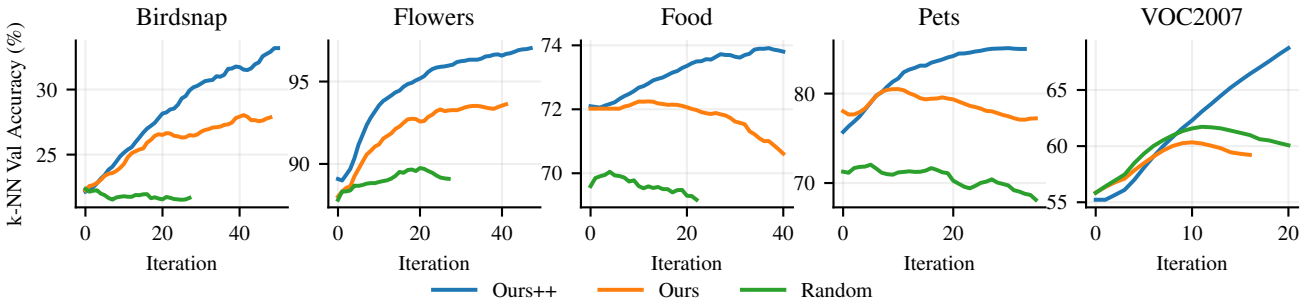


Figure 5. **Learning curves in self-supervised setting.** We show how k-NN validation accuracy improves across iterations on each target dataset. Without using any labels, Internet Explorer identifies and focuses on relevant concepts for each target dataset. This allows it to find more useful data than 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.

2014), Flowers-102 (Nilsback & Zisserman, 2008), Food101 (Bossard et al., 2014), and Oxford-IIT Pets (Parkhi et al., 2012). These small datasets consist of 2,040 to 75,750 training examples, making them ideal for testing whether Internet Explorer can efficiently find relevant useful data. We also evaluate on PASCAL VOC 2007 (Cls) (Everingham et al., 2010), a coarse-grained multi-label classification task, and ImageNet-100 (Tian et al., 2020). Finally, we try FMoW (Christie et al., 2018), 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.

4. Results and Analysis

4.1. Self-supervised Results

Figure 5 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 (Mezuman & Weiss, 2012; Chen & Gupta, 2015). Table 1 shows that our method significantly improves on the starting MoCo-v3 (ImageNet + target) checkpoint and can outperform a CLIP (Radford et al., 2021) model of the same size while using much less compute and 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.

4.2. Self-supervised Exploration Behavior

Figure 6 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

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)	86.7 (+2.0)	**54.5** (+**2.0**)	2.2×10^6	84 + 40

Table 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 (see Appendix C.8 for more details).

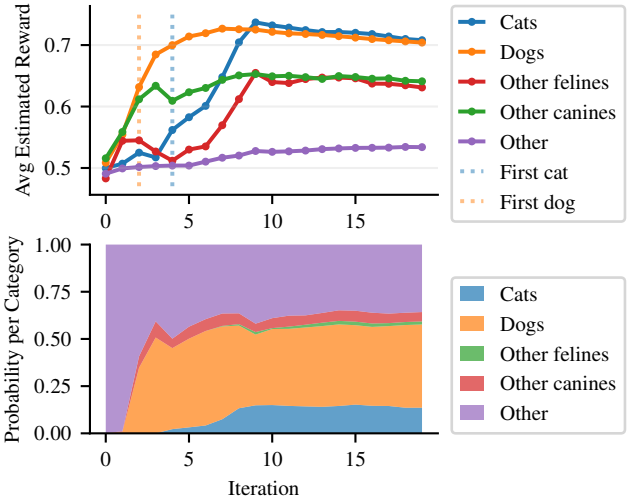


Figure 6. **Self-supervised concept discovery on Pets dataset.** When targeting the Pets dataset, self-supervised Internet Explorer quickly estimates high reward for concepts from the cat category (82 concepts) and dog category (246 concepts). It is also able to identify felines that are not cats (e.g., 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.

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 6 (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 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 $\#3$. Appendix E shows that Internet Explorer identifies useful images shockingly quickly across every dataset, without any knowledge of their label sets.

4.3. 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.

4.4. Domain dataset results

To test if Internet Explorer is effective when the target dataset contains very specific domain knowledge, we applied it to FMoW-WILDS (Christie et al., 2018)—a popular satellite imaging domain dataset—by hand-designing a dozen search prompts that help induce satellite image results (details in Appendix C.8). Even though the WordNet

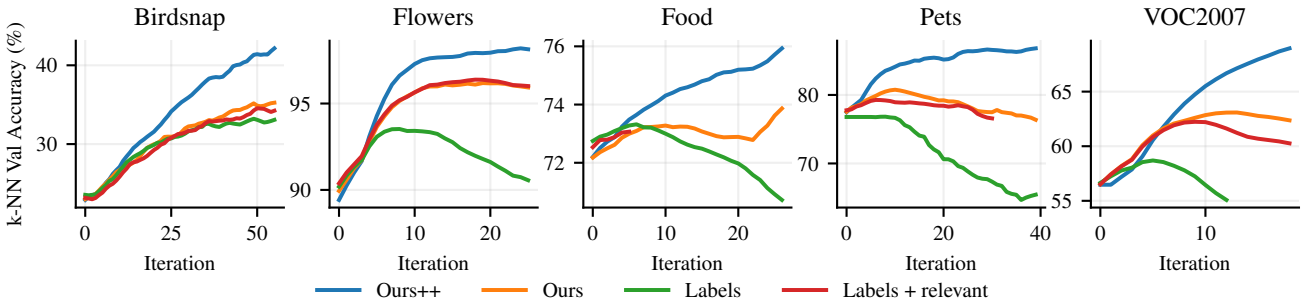


Figure 7. **Learning curves in label set-guided setting.** Using knowledge of the label set improves the performance of all methods.

vocabulary is not particularly suited for this dataset, Internet Explorer still improves the LP accuracy by 2 percentage points (see Table 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.

4.5. 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 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 (Schuhmann et al., 2022). 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. We discuss more details in Appendix A. Table 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.

4.6. 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 2.3), we also try using $k = 1$ or the MoCo contrastive loss as the reward. Table 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 8. 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.

Figure 8. **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.

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

Table 3. **Ablation on type of image reward.** MoCo loss does not identify relevant concepts, and 1-NN is sensitive to outlier images.

4.7. 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 relies on using strong visual representations from pretrained models in order to identify similar images. This defeats the

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. **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.

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 because it makes use of additional supervision, in the form of the caption and surrounding text, that makes it easier to consistently index new images. Image-to-image search is additionally quite expensive, as it relies on paid APIs that can cost thousands of dollars per experiment.

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 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.

5. Related Work

Many papers use self-supervised or weakly-supervised learning on large-scale static datasets collected from the Internet, such as YFCC-100M (Thomee et al., 2015), Instagram-1B (Mahajan et al., 2018), or LAION-5B (Schuhmann et al., 2022). However, these are usually impractically expensive since they attempt to train on all of the data, not just the subset relevant for a target dataset. Another line of work continuously interacts with the Internet to find useful data, instead of using fixed-size scrapings. NELL (Carlson et al., 2010; Mitchell et al., 2018) extracts text from web pages to form beliefs, and NEIL (Chen et al., 2013) 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 proceeds slowly. Kamath et al. (2022) improves a visual question-answering model using a set of predetermined Bing queries. In contrast to these works, Internet Explorer uses targeted exploration on the Internet to find data for self-supervised training.

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

Table 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.

6. Conclusion

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 (Radford et al., 2021) trained on static datasets, as well as strong baselines that search the Internet in an undirected manner.

Acknowledgements We thank Russell Mendonca for helpful discussions and Shivam Duggal, Mihir Prabhudesai, Sheng-Yu Wang, Jason Y. Zhang, and Rishi Veerapaneni for paper feedback. AL is supported by the NSF GRFP, grants DGE1745016 and DGE2140739. This work is supported by NSF IIS-2024594 and ONR MURI N00014-22-1-2773.

References

Bao, H., Dong, L., and Wei, F. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Berg, T., Liu, J., Woo Lee, S., Alexander, M. L., Jacobs, D. W., and Belhumeur, P. N. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2011–2018, 2014.

Bossard, L., Guillaumin, M., and Gool, L. V. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Buchner, J. imagehash (fork). https://github.com/JohannesBuchner/imagehash, 2021.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*, 2010.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *preprint arXiv:2002.05709*, 2020.

Chen, X. and Gupta, A. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1431–1439, 2015.

Chen, X., Shrivastava, A., and Gupta, A. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE international conference on computer vision*, pp. 1409–1416, 2013.

Chen, X., Xie, S., and He, K. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021.

Christie, G., Fendley, N., Wilson, J., and Mukherjee, R. Functional map of the world. In *CVPR*, 2018.

Clinton, J. Google images download (fork). https://github.com/Joeclinton1/google-images-download, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

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

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Harwood, B., Kumar BG, V., Carneiro, G., Reid, I., and Drummond, T. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2829, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

Jiang, Z., Chen, T., Chen, T., and Wang, Z. Improving contrastive learning on imbalanced data via open-world sampling. *Advances in Neural Information Processing Systems*, 34:5997–6009, 2021.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7 (3):535–547, 2019.

Kamath, A., Clark, C., Gupta, T., Kolve, E., Hoiem, D., and Kembhavi, A. Webly supervised concept expansion for general purpose vision models. *arXiv preprint arXiv:2202.02317*, 2022.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Li, A. C., Efros, A. A., and Pathak, D. Understanding collapse in non-contrastive siamese representation learning. In *European Conference on Computer Vision*, pp. 490–505. Springer, 2022.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

Mezuman, E. and Weiss, Y. Learning about canonical views from internet image collections. *Advances in neural information processing systems*, 25, 2012.

Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *preprint arXiv:1807.03748*, 2018.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Paul, M., Ganguli, S., and Dziugaite, G. K. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34: 20596–20607, 2021.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Robinson, J., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Settles, B. Active learning literature survey. 2009.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794. Springer, 2020.

Vasa, H. Google images download. https://github.com/hardikvasa/google-images-download, 2015.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Williams, C. and Rasmussen, C. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *preprint arXiv:1708.03888*, 2017.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Appendix

A. 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.

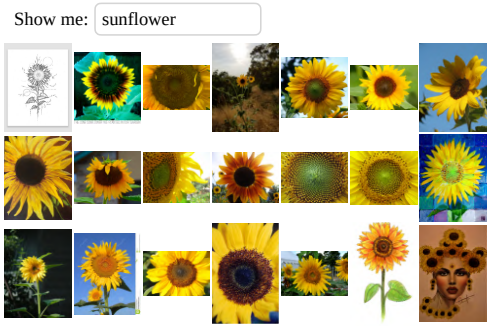


Figure 9. **Our custom LAION-5B search engine.** We build a custom text-to-image search engine that finds images within the LAION-5B dataset by doing nearest neighbor search in text embedding space. This uses no image features whatsoever.

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 (Schuhmann et al., 2022), 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 use a pre-trained text embedding model (Reimers & Gurevych, 2019) to compute 384-dimensional text embeddings for each caption. Then, we use Faiss (Johnson et al., 2019) to compute a fast, approximate nearest-neighbors lookup index. Querying our custom search engine finds 100 image results in less than a second. Figure 9 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 11 shows that each data source has its own tendencies. For the "spaghetti bolognese" query, Google Images is biased (Mezuman & Weiss, 2012; Chen & Gupta, 2015) 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 10 and Table 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.

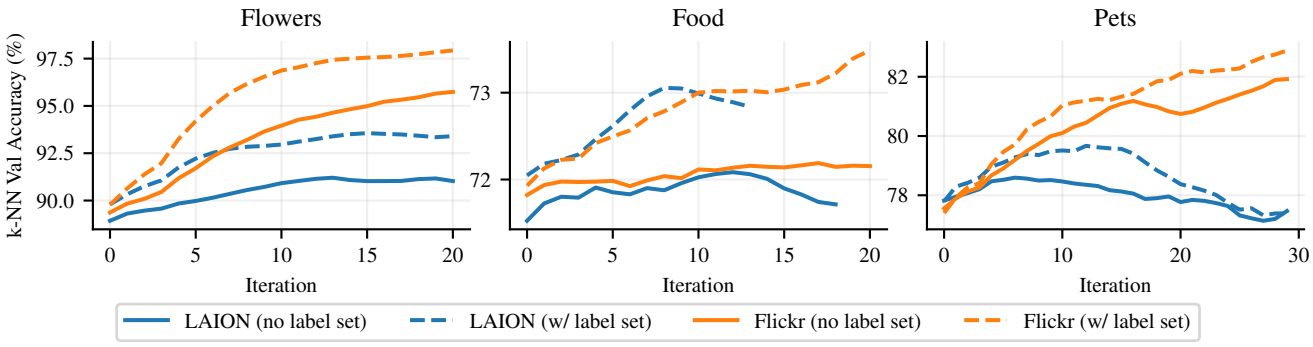


Figure 10. **Learning from Flickr and LAION-5B.** Even with the noisy search results returned by Flickr and LAION, Internet Explorer still continuously improves performance.

Food101 dataset: “Spaghetti Bolognese”

Google Images: “Spaghetti Bolognese”

Flickr: “Spaghetti Bolognese”

LAION-5B: “Spaghetti Bolognese”

Figure 11. **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 5. **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.

B. 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) (Buchner, 2021) 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 5. 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.

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 12 to 16. 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.

C. Method Details

C.1. WordNet Lemmas

We draw our concepts from the WordNet hierarchy (Miller, 1995), 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".
```

Test Img. Ranked Nearest Neighbors in Downloaded Images

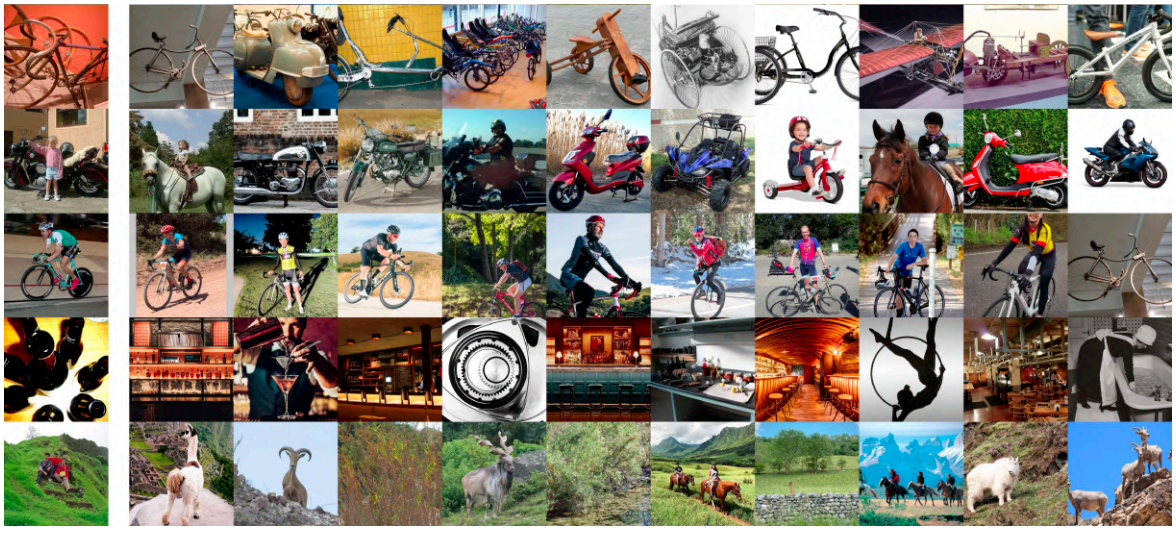
Figure 12. **Top-10 most similar online images to Pets101**

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure 13. **Top-10 most similar online images to Food101**

Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure 14. **Top-10 most similar online images to Flowers102**



Test Img. Ranked Nearest Neighbors in Downloaded Images

Figure 15. **Top-10 most similar online images to PASCAL VOC2007**

Figure 16. **Top-10 most similar online images to IN100**

C.2. GPT-J Descriptor Prompting

We use GPT-J-6B (Wang & Komatsuzaki, 2021), 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".
```

C.3. Concept Vocabulary Size

As stated in Section 2.2, 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

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

Table 6. **Target Dataset "Category"**.

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 (Radford et al., 2021). Doing so would expand our vocabulary to roughly $500{,}000$ total concepts.

C.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

$$\text{SMR} = \frac{\max_i r(c_i) - \min_i r(c_i)}{\tau} \tag{4}$$

where the "softmax range" 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:

$$\frac{\max_i p(c_i)}{\min_i p(c_i)} = \frac{\max_i \exp(r(c_i)/\tau)}{\min_i \exp(r(c_i)/\tau)} \tag{5}$$

$$= \exp\left(\frac{\max_i r(c_i) - \min_i r(c_i)}{\tau}\right) \tag{6}$$

$$= \exp(\text{SMR}) \tag{7}$$

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 (Reimers & Gurevych, 2019) 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 6) 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.

C.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

Hyperparameter	Value
Architecture	Resnet-50 (He et al., 2016)
Optimizer	LARS (You et al., 2017)
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 7. **Internet Explorer hyperparameters**.

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.

C.6. Hyperparameters

Table 7 shows our hyperparameter values, which are shared across datasets. We perform minimal hyperparameter tuning and copy most of the values from the MoCo-v3 (Chen et al., 2021) 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.

C.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.

C.8. FMoW-WILDS Training Details

Each query to a search engine is randomly prepended with one of the below "descriptors." This list was selected by taking some random concepts and trying (via trial & error) search queries that seemed most likely to return satellite-view results. The below list is the result of this trial-and-error process. Note that GPT-J is not used to generate descriptors for this dataset.

FMoW 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",
```

D. Proof of Lemma 2.1

Here, we prove Lemma 2.1 from Section 2.6, which we repeat below:

Lemma 2.1. *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.

$$T_{base} = \sum_{i=1}^{cs} t_i \tag{8}$$

$$= \sum_{i=1}^{cs} \frac{1}{p_i} \tag{9}$$

$$= \sum_{i=1}^{cs} \frac{n}{cs+1-i} \tag{10}$$

$$= n \sum_{i=1}^{cs} \frac{1}{cs+1-i} \tag{11}$$

$$= nH_{cs} \tag{12}$$

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.

$$T_{GPR} = \sum_{i=1}^{c} t_i \tag{13}$$

$$= \sum_{i=1}^{c} \frac{1}{p_i} \tag{14}$$

$$= \sum_{i=1}^{c} \frac{n}{s(c+1-i)} \tag{15}$$

$$= \frac{n}{s} \sum_{i=1}^{c} \frac{1}{c+1-i} \tag{16}$$

$$= \frac{nH_c}{s} \tag{17}$$

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 6), 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.

E. Progression of downloaded images

Just as Figure 4 in the main paper showed how Internet Explorer progressively discovers useful data when targeting the Pets dataset, Figures 17 to 20 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.

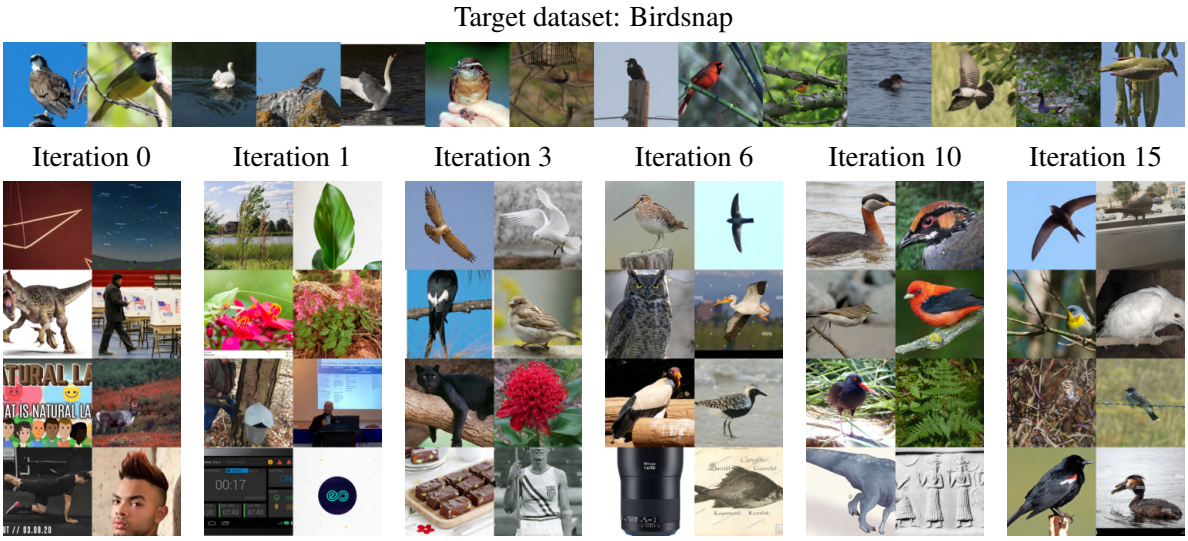


Figure 17. **Progression of downloaded Birdsnap images.** This corresponds to Ours++ without using label set information.

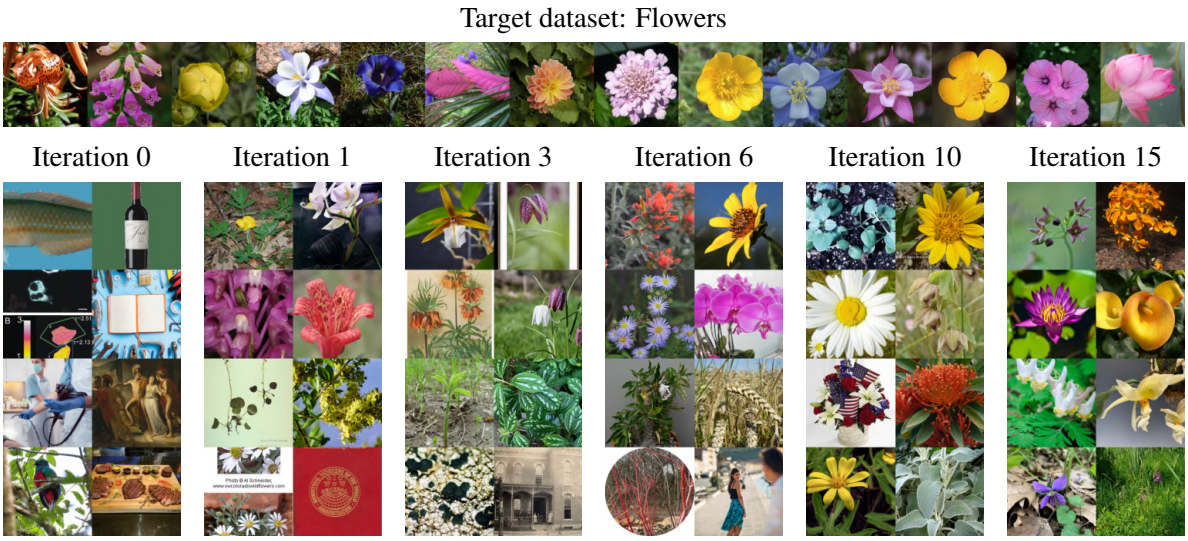


Figure 18. **Progression of downloaded Flowers images.** This corresponds to Ours++ without using label set information.

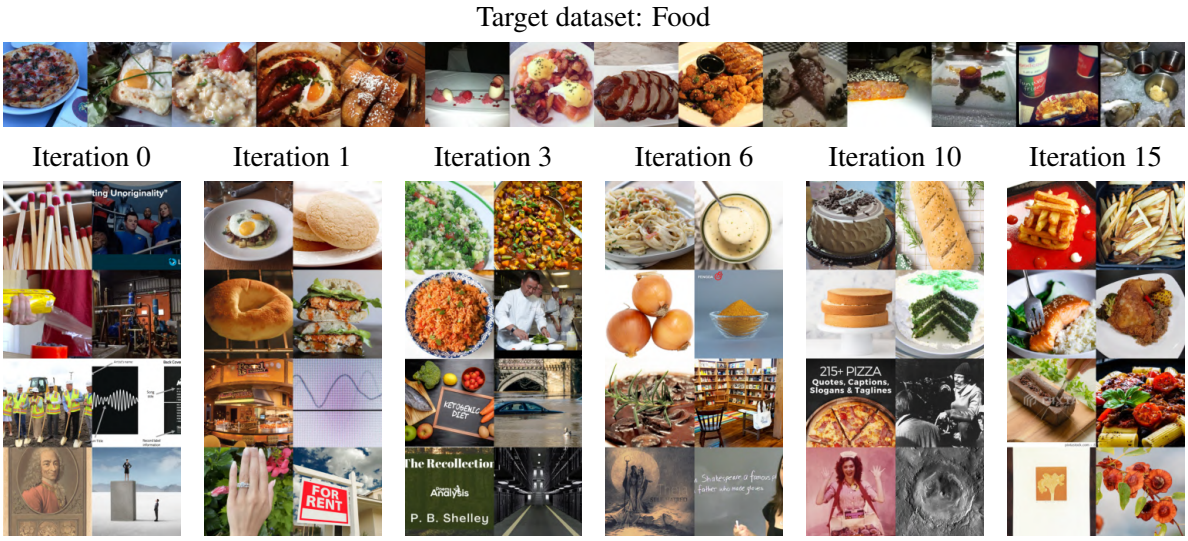


Figure 19. **Progression of downloaded Food images.** This corresponds to Ours++ without using label set information.

Figure 20. **Progression of downloaded VOC2007 images.** This corresponds to Ours++ without using label set information.