Internet Explorer: Representation Learning on the Open Web

Alexander C. Li[1*], Ellis Brown[1*], Alexei A. Efros[2], Deepak Pathak[1]
[1]Carnegie Mellon University, [2]University of California, Berkeley

Abstract

Vision models today are almost always pre-trained on large, static datasets and then adapted to downstream tasks. Despite containing hundreds of millions of images, their pre-training datasets are easily dwarfed by the scale of the Internet, where billions of images are uploaded each day. In this paper, we propose treating the Internet as a dynamic, open-ended dataset. Given a small, unlabeled, target dataset, our approach named Internet Explorer explores the web in a self-supervised manner to progressively find relevant examples via text queries that improve performance on the 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 closely matches CLIP fine-tuning despite using a single GPU desktop actively querying the Internet for 30-40 hours.

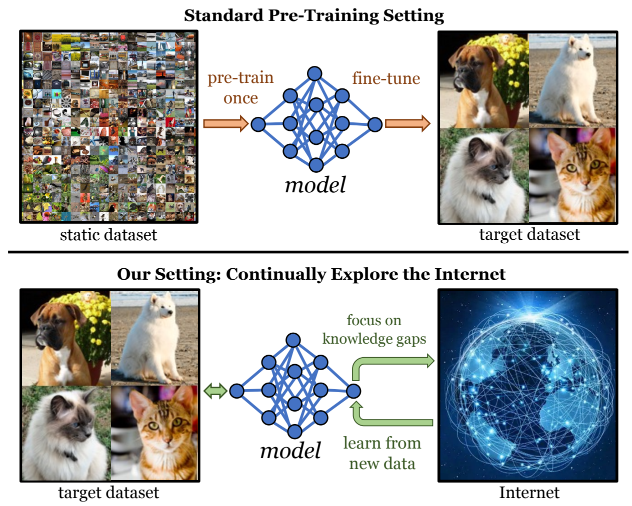

Figure 1. Given an unlabeled data for a target task, our approach, Internet Explorer, searches the Internet to progressively find out more and more relevant training data via self-supervised training.

1. Introduction

Suppose you have a small dataset and need to train a model for some task, say classification. How would you go about doing it? A pipeline that has become standard today is to download the latest and greatest pre-trained deep network and fine-tune it on your own small dataset. This pre-trained model used to be ImageNet-based [17, 25] and now would probably be CLIP [45]. Although the size of the datasets these models are pre-trained on has grown from 1.2M to 400M images, what has not changed at all is their nature: these datasets are curated, and, more importantly, *static*. Although a few hundred million images represent a staggering quantity of visual data, they are minuscule compared to the entire Internet where billions of photos are uploaded every day, continuously capturing an incredible diversity of real-world objects and scenes. So the "big data" in machine learning is easily dwarfed by the data generated collectively by the world. Furthermore, the bigger we make our static pre-training datasets, the more compute burden they entail: e.g., CLIP is trained on 256 GPUs for 12 days. This begs the question: are static datasets, as big as they are, ever going to truly scale to capture the richness and dynamic nature of the data available on the Internet?

In this paper, we suggest thinking beyond static datasets and *treating 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 stays up to date. Its continuously evolving nature also means we cannot hope to ever download it or train a model on all of it. But thinking pragmatically, do we even need to do so? Perhaps not.

We argue that the Internet can be treated as a special kind of dataset – one that just exists out there, to be queried as needed to quickly train a customized model. However, the issue is that the Internet is too big, and finding relevant images that help improve performance on a target dataset is a challenging endeavour. This is analogous to reinforcement learning in robotics, where even if 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 should the agent take so that it quickly finds

*Authors contributed equally to this work.

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 improves feature quality on a target dataset (see Fig. 1). The actions are text queries and the observation is the images obtained by querying.

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 were useful, and prioritizing what to search for next (see Fig. 2). Our setting is different from active learning [51], 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. However, we also show results in semi-supervised settings when the label set of the target dataset (not individual labels) are known.

Some prior works have also discussed ways to leverage the Internet as an additional source of supervision. For instance NELL [8] proposed an automatic way to read the internet from web pages to retrieve new concepts and relationships which are curated by a human in the loop once in a while. NEIL [14] builds on the dictionary developed by NELL to search visual data to develop visual relationships. Both of these are semi-supervised methods to gather general 'common-sense' knowledge off 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 [29] follows a similar setting, but they search on a static dataset and not on the Internet.

We evaluate Internet Explorer across 5 datasets including 4 fine-grained datasets and PASCAL VOC. For simplicity, the search engine used is Google, but the method itself can work by searching on just image tags/captions as well. We compare against several strong baselines including CLIP fine-tuning on downstream tasks. In most scenarios, Internet Explorer either outperforms or matches CLIP 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.

2. Internet Explorer: An Online Agent

We focus on the problem of quickly improving performance on an arbitrary task with a corresponding image dataset. We make as few assumptions as possible and assume that we have only unlabeled training data from the target domain, without any labels or information about the dataset. We can apply self-supervised methods directly to the target dataset, but performance quickly saturates—especially if the target dataset is small. We thus prioritize selectively collecting the data that is expected to improve

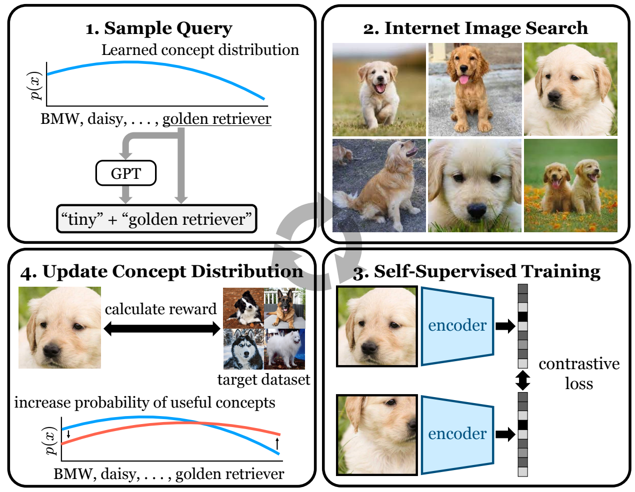

Figure 2. **Overview of Internet Explorer.** Our goal is to efficiently search the Internet for images that improve our performance on a target dataset. In each iteration, we generate text queries by combining a concept sampled from a learned distribution with a GPT-generated descriptor. We query Google Images with the resulting phrase and download the top 100 image results. We add these images to the set of previously downloaded images and perform self-supervised learning on the combined dataset. Finally, we evaluate the relevance of the new images and increase the likelihood of the query and other related queries if the new images were similar to the target dataset.

the current model's performance on the target task.

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 rank 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). For this reason, 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 [16, 54].

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 [37], which consists of $146,347$ noun lemma names. Not all of these lemmas are visual, but the vocabulary still covers an incredible range of topics. For reference, here are 6 randomly sampled concepts: `'sleep talking'`, `'beach wagon'`, `'Balearic Islands'`, `'borosilicate'`, `'genus Loranthus'`, `'humpback whale'`.

We generate descriptors for each concept by prompting a GPT-J language model [55] with examples of descriptor-concept pairs (details in the supplementary). For reference, here are 7 randomly sampled descriptors for "labrador retriever": `'friendly'`, `'short'`, `'long-legged'`, `'big'`, `'fast'`, `'blue-eyed'`, `'handsome'`.

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. We experimented with using SimSiam [12] as our base SSL algorithm due to its simplicity (no temperature, EMA rate, or loss weighting terms to tune), but found that it was unstable to train [34]. Instead, we use MoCo-v3 [15]. MoCo-v3 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 [42]:

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

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.

In each iteration of our method, we use MoCo-v3 to fine-tune a ResNet-50 model [25] on a mixture of newly downloaded, previously downloaded, and target dataset images. Before turning to the Internet, we initialize our model using a MoCo-v3 checkpoint trained offline for 100 epochs on ImageNet and then fine-tuned with MoCo-v3 on the target dataset. Without using labels, we select the starting checkpoint for Internet Explorer by early-stopping on the SSL loss, which highly correlates with target accuracy [34].

Even though we focus on using MoCo-v3 in this paper, note that Internet Explorer is compatible with any SSL algorithm that uses images or image-text pairs, including contrastive [10, 24], non-contrastive [4, 9, 21, 59], masking-based [3, 23], or multimodal [45] approaches.

2.4. Image Ranking 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 extremely challenging to directly measure the effect of an individual training example on performance. Numerous techniques have been proposed [19,28,32,44], but they all require extensive training on new data to estimate their impact.

Instead of trying to precisely measure what is learned from each image, we rank the images by their distance in representation space to the target dataset images. The images 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" [20,22,41,47,48,57] 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:

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

where S_{cos} is the cosine similarity. A previous metric for identifying relevant data [29] used $k = 1$ nearest neighbors, but we found that this was too noisy and allowed high reward for outlier target images to distract our search. We instead use $k = 15$ to improve the accuracy of our relevance estimation. In Sec. 4.4, we compare our reward to alternatives and explore their failure modes. Our reward is used for two purposes: determining which of the downloaded images to train on and, subsequently, which concepts to search for next.

Which images to train on. At the start of each iteration, we sample a batch of queries from the internet. Many of our 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

Algorithm 1 `Internet Explorer`

Input: target dataset $\mathcal{D}$, SSL algorithm $\mathbb{A}$, `SearchEngine`, encoder $f : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^d$, replay buffer $\mathcal{B}$, image reward function r, vocabulary $\mathcal{V} = \{c_i\}_{i=1}^C$, concept reward predictor `RewardPredictor`, concept distribution p, # concepts/itr M, # query results Q, concept distribution function `CalcProbs`
for iteration $= 1, 2, \ldots$ **do**
  for $i = 1, \ldots, M$ **do**
    Sample concept c_i from $\mathcal{V}$ using distribution p
    Obtain images $\{I_j^i\}_{j=1}^Q \leftarrow$ `SearchEngine`(c_i)
    Calculate image rewards $r(f, \mathcal{D}, I_j^i)$
    Calculate concept reward from image rewards
  end for
  $\mathcal{B}_{\text{new}} = \{I_j^1\}_{j=1}^Q \cup \cdots \cup \{I_j^M\}_{j=1}^Q)$
  SSL training: $\mathcal{A}(f, \mathcal{D} \cup \mathcal{B} \cup \mathcal{B}_{\text{new}})$
  Add to replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup$ `Filter`$(\mathcal{B}_{\text{new}}, r)$
  Save new concept rewards, predict unseen rewards
  $p \leftarrow$ `CalcProbs(RewardPredictor)`
end for

$r_i = r(f_k, \mathcal{D}, I_i)$ 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 for every single one to test whether it 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 text meaning, we embed our $146,347$ WordNet concepts into a 384-dimensional space using a pre-trained sentence similarity model [46]. To provide relevant context to the text embedding model, we use the following template as the input for each concept:

`{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 (GP) [56] over the text embeddings $\{\mathbf{e}_i\}$ to predict the concept-level reward $r(\mathbf{e}_i)$ for untried concepts. GP 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 $R_{obs} = \{r(\mathbf{e}_i)\}$, GP 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 $\mu(\mathbf{e}_i) + \sigma(\mathbf{e}_i)$.

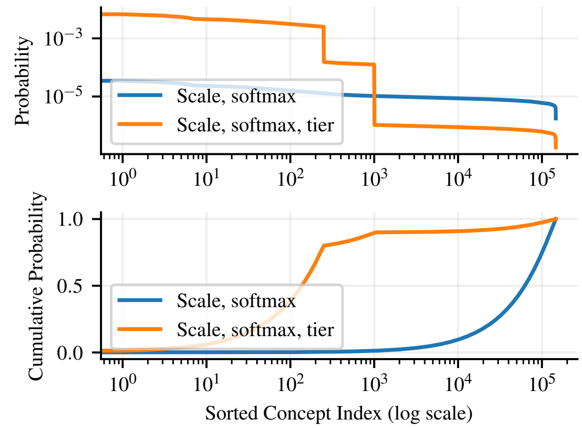

Figure 3. **Learned concept sampling distribution.** Given estimated scores for each of the $146,347$ concepts, we need to choose how often to sample each one in order to balance exploration and exploitation. **Top:** we scale our scores to a desired temperature, then take the softmax to obtain a distribution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

2.6. 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:

1. We need to sample the top concepts frequently enough to get relevant training data for SSL.
2. At the same time, we need sufficient exploration of promising untried concepts.

A greedy approach, like only searching for the top C concepts with the highest estimated reward, results in poor exploration of untried concepts. Instead, we use a sampling-based approach based on Boltzmann exploration [52]. Boltzmann exploration typically samples based on a scaled softmax distribution $p(c_i) \propto \exp(r(c_i)/\tau)$, where where τ is the temperature scaling factor. However, with a large vocabulary of $146,347$ concepts, it becomes diffi-

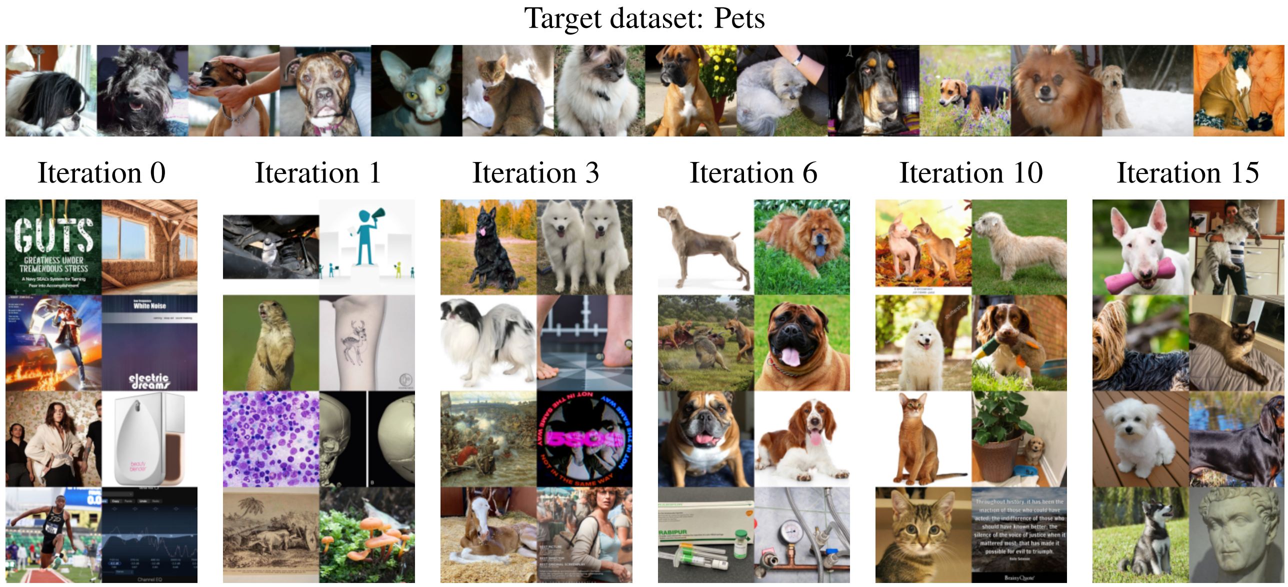

Figure 4. **Progression of downloaded images over training.** The top row shows the target distribution and the bottom ones show the sets of images queried by our Internet Explorer. As the progression goes on, the image set discovered by the Internet Explorer starts to more closely resemble the target dataset distribution.

cult to tune τ so that we sample the top concepts frequently enough without being too skewed. Thus, we use 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:

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

p^{tier} is a new distribution such that $\sum_{k=T_j}^{T_{j+1}} p^{\text{tier}} = \Delta_j$. We use $T_0 = 0$, $T_1 = 250$, $T_2 = 1000$, $T_3 = 146,347$, $\Delta_0 = 0.8$, $\Delta_1 = 0.1$, 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.

An overview of the full Internet Explorer is depicted in Figure 2 and described in Algorithm 1.

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. In contrast to previous work that uses fixed datasets [29], we use the Internet as an essentially infinite source of potential training data. Successful feature learning in this setting would lead to better performance on standard downstream tasks like classification and detection, as well as on other tasks where the labels may not be semantic. This includes depth prediction, colorization, and tasks in visual reinforcement learning and robotics. We compare three methods in this setting:

1. Random: this baseline searches for concepts that are sampled uniformly from the vocabulary.
2. Ours: we sample queries from our learned concept distribution.
3. Ours++: additionally use GPT-generated descriptors

3.2. Semi-supervised Exploration

We may sometimes know the set of labels for our task (e.g., "golden retriever", ..., "chihuahua") 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 here:

1. Labels: this baseline only searches for the labels.
2. Labels + relevant: this baseline searches for labels half of the time, and random concepts from the pruned vocabulary the other half of the time.
3. Ours: we sample labels half of the time and sample

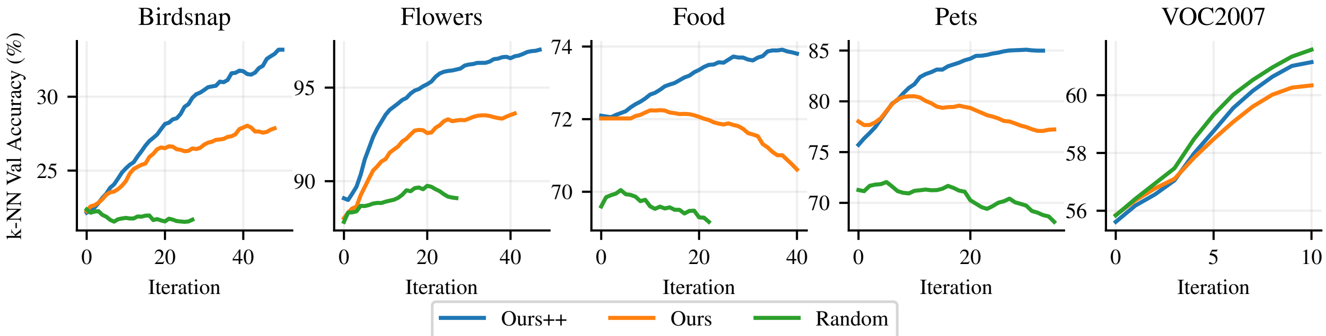

Figure 5. **Learning curves in self-supervised setting.** We show how k-NN validation accuracy improves across iterations on each target dataset. Without using any labels, Internet Explorer identifies and focuses on relevant concepts for each target dataset. This allows it to find more useful data than 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. Interestingly, the random baseline does quite well on VOC2007, perhaps because coarse-grained classification benefits from a broader variety of training data.

from our learned concept distribution the other half.

4. Ours++: additionally use GPT-generated descriptors.

We call this setting “semi-supervised,” since we have additional supervision in the form of the label set. Note that this differs from the typical semi-supervised learning setting, which learns from fixed labeled and unlabeled datasets.

3.3. Datasets and Metrics

We evaluate Internet Explorer on Birdsnap [5], Flowers-102 [40], Food101 [6], and Oxford-IIT Pets [43], which are small-scale fine-grained classification datasets commonly used to evaluate transfer learning performance for large pre-trained models [33, 45]. We also evaluate on Pascal VOC 2007 (Cls) [18], a coarse-grained multi-label classification task. 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. For these datasets, we compare the representation quality of our models using two metrics: k-nearest neighbors (k-NN) accuracy and linear probe accuracy. k-NN accuracy can be computed quickly, so we use this to plot learning curves of model performance after every iteration. We compare the linear probe accuracy of our final checkpoints. Note we do not include large-scale datasets like ImageNet [17] as target because they already contain over a million images human-curated from the internet.

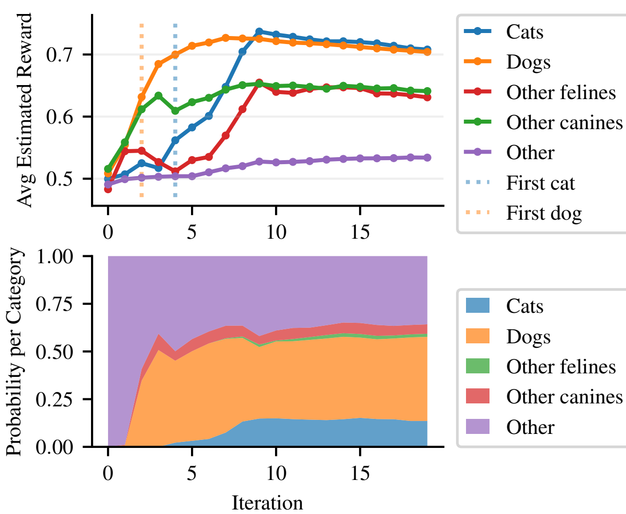

Figure 6. **Self-supervised concept discovery on Pets dataset.** When targeting the Pets dataset, self-supervised Internet Explorer quickly estimates high reward for concepts from the cat category (82 concepts) and dog category (246 concepts). It is also able to identify the felines that are not cats (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 $460/146347 = 0.3\%$ of the vocabulary.

4. Results and Analysis

4.1. Self-supervised Results

Figure 5 shows that Internet Explorer is far better than the baseline that samples queries uniformly at random from the concept vocabulary. In fact, random sampling occasionally decreases accuracy, likely due to the fact that Internet images can be unsuitable for general pre-training, with issues like watermarks, images of text, and unrealistically photogenic images [11, 36]. Table 1 shows that our method universally improves on the starting MoCo model and can outperform a CLIP [45] model of the same size while using much less compute and data. Using GPT-generated descriptors in “Ours++” also significantly improves performance by enabling Internet Explorer to generate diverse views of the most useful concepts. We show example image results with and without descriptors in the supplementary.

Model	Birdsnap	Flowers	Food	Pets	VOC2007	Images	GPU-hours
Fixed dataset Self-Supervised							
MoCo-v3 [15] (ImageNet pre-train)	26.8	83.2	70.5	79.6	-	1.2×10^6	72
MoCo-v3 [15] (ImageNet + target)	39.9	94.6	78.3	85.3	-	1.2×10^6	72 + 12
Fixed dataset Semi-Supervised							
CLIP ResNet-50 [45]	57.1	96.0	**86.4**	88.4	-	400×10^6	4000
No label set information							
Random exploration	39.6	95.3	77.0	85.6	61.8	2.2×10^6	84 + 40
Ours	43.4	97.1	80.5	86.8	60.3	2.2×10^6	84 + 40
Ours++	53.5	98.4	82.2	89.6	61.4	2.2×10^6	84 + 40
Use label set information							
Search labels only	47.1	96.3	80.9	85.7	52.9	2.2×10^6	84 + 40
Labels + semantically relevant terms	49.9	98.0	81.2	87.0	62.4	2.2×10^6	84 + 40
Ours	52.0	97.6	81.2	87.3	63.1	2.2×10^6	84 + 40
Ours++	**62.8**	**99.1**	83.8	**90.8**	**69.1**	2.2×10^6	84 + 40

Table 1. **Linear probe accuracy on targeted datasets**. Our method significantly improves performance on each dataset while using 2.5% as much compute and 0.5% as much data as CLIP. For VOC2007, we do not do ImageNet pre-training, because ImageNet's diversity and curation make it good for VOC pre-training. We report k-NN accuracy in the VOC2007 column and show LP in the supplementary.

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 in our vocabulary into 5 meaningful categories: cats, dogs, non-cat felines (e.g., lion), non-dog canines (e.g., wolf), and other. Note that this categorization is only done for this post hoc analysis and is not provided during training. Figure 6 (top) shows the average estimated reward within each category across the first 20 iterations of training, and the bottom one shows how much probability mass each category has.

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

4.3. Semi-supervised Results

Internet Explorer significantly outperforms the stronger baselines in the semi-supervised 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. Note that we train Internet Explorer from scratch on VOC2007 and do not use an ImageNet pre-trained MoCo-v3 model as initialization.

4.4. Effect of image reward type

We run an ablation on the type of reward we use to rank images. Instead of calculating the image reward based on the average similarity to the $k = 15$ nearest neighbors in representation space (as discussed in Sec. 2.3), we also try using $k = 1$ or the MoCo contrastive loss as the reward. Table 2 compares these three metrics in the semi-supervised setting and shows that $k = 15$ does best. We explain this result by qualitatively comparing the metrics' behavior on Food101 in Fig. 8. The MoCo loss generally cannot identify relevant concepts and struggles to search for relevant data. Instead, it prefers to search for images that are difficult to align across augmentations. Representation similarity with $k = 1$ also fails, as it prefers images of zebras and text. This is 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.

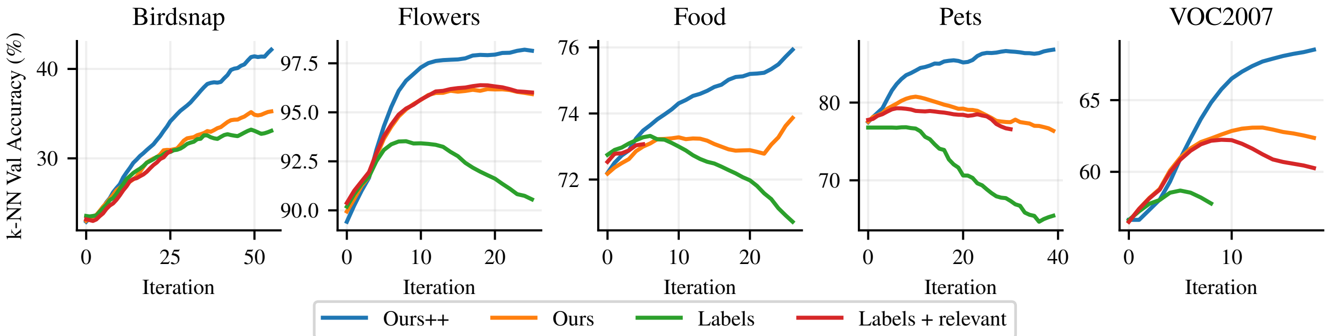

Figure 7. **Learning curves in semi-supervised setting.** Using knowledge of the label set improves the performance of all methods.

Reward Type	Food
MoCo loss	81.2
1-NN similarity	83.2
15-NN similarity (ours)	**83.8**

Table 2. **Ablation on type of image reward.** We compare LP accuracy of 3 different rewards on Food in the semi-supervised setting. MoCo loss does not identify relevant concepts, and $k = 1$ similarity is too noisy to identify useful concepts.

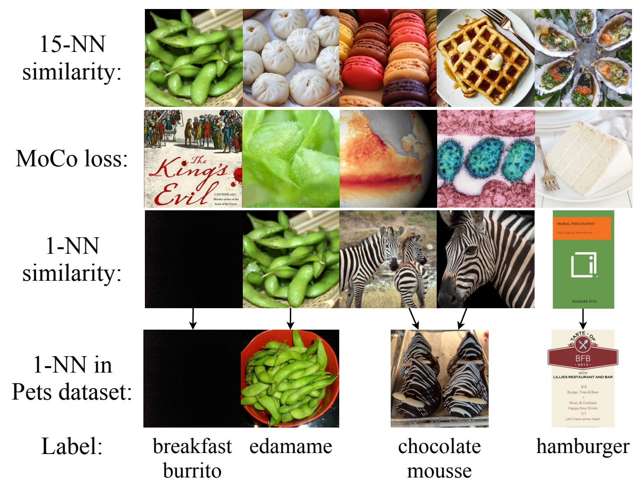

Figure 8. **Top images preferred by different rewards.** We show the top 5 downloaded images ranked by 3 possible image rewards on the Food dataset. 15-NN (ours) prefers a variety of food images, whereas MoCo prefers noisy images out of the training distribution. 1-NN is thrown off by outliers in the Food dataset and thus prefers black images, text, and zebras.

5. Related Work

Self-supervised Learning Contrastive SSL approaches pull together positive pairs (usually obtained by augmenting the same image twice) and push apart negative pairs [10, 15, 24, 26, 27, 38, 42]. Non-contrastive methods do not explicitly push apart negative pairs, but implicitly do so using self-distillation [9, 13, 21] or feature correlation [4, 59]. Finally, masking-based approaches train models to predict masked portions of the input [1–3, 23]. Internet Explorer is agnostic to the choice of SSL algorithm and can be used with any of these approaches, as long as there are low-dimensional representations to compute our image reward.

Learning from Uncurated Internet Data Many papers do self-supervised or weakly-supervised learning on large-scale, uncurated, static datasets collected from the Internet, such as YFCC-100M [53], Instagram-1B [35], or LAION-400M [50]. However, these are almost always impractically expensive experiments since they attempt to train on all of the data, not just the subset that is relevant for a target dataset. Another line of work continuously interacts with the Internet to find useful data, instead of using a fixed-size scraping. NELL [8, 39] extracts text from hundreds of millions of web pages in order to learn candidate beliefs, and NEIL [14] 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. [31] improves a visual question-answering model using a set of predetermined Bing queries. In contrast to these works, Internet Explorer uses targeted exploration to find data for self-supervised training.

6. Discussion and Conclusion

We show that interactively exploring the Internet is an efficient source of highly relevant training data – if you know how to search for it. In just 30-40 hours of training on a single GPU, Internet Explorer either significantly outperforms or closely matches compute-heavy models like CLIP [45] trained on static datasets, as well as strong baselines that search the Internet in an undirected manner.

Our approach is also general. In the supplementary material, we show that Internet Explorer does not have to rely on Google to find useful data. We use Internet Explorer to select relevant training data from a large, uncurated image-text corpus (LAION-400M [50]) by using a pre-trained text embedding model. This approach significantly improves performance compared to training on all of the data.

References

[1] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. *arXiv preprint arXiv:2204.07141*, 2022. 8

[2] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv preprint arXiv:2202.03555*, 2022. 8

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3, 8

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 3, 8

[5] 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*, pages 2011–2018, 2014. 6

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014. 6

[7] Johannes Buchner. imagehash (fork). `https://github.com/JohannesBuchner/imagehash`, 2021. 11

[8] 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. 2, 8

[9] 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*, pages 9650–9660, 2021. 3, 8

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *preprint arXiv:2002.05709*, 2020. 3, 8

[11] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1431–1439, 2015. 6, 11

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *preprint arXiv:2011.10566*, 2020. 3

[13] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 8

[14] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE international conference on computer vision*, pages 1409–1416, 2013. 2, 8

[15] 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*, pages 9640–9649, 2021. 3, 7, 8, 14, 15

[16] Joe Clinton. Google images download (fork). `https://github.com/Joeclinton1/google-images-download`, 2020. 2

[17] 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*, pages 248–255. Ieee, 2009. 1, 6

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 6

[19] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020. 3

[20] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 3

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020. 3, 8

[22] 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*, pages 2821–2829, 2017. 3

[23] 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*, pages 16000–16009, 2022. 3, 8

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 3, 8

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3, 14

[26] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020. 8

[27] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *preprint arXiv:1905.09272*, 2019. 8

[28] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022. 3

[29] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Improving contrastive learning on imbalanced data via open-world sampling. *Advances in Neural Information Processing Systems*, 34:5997–6009, 2021. 2, 3, 5

[30] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 11

[31] Amita Kamath, Christopher Clark, Tanmay Gupta, Eric Kolve, Derek Hoiem, and Aniruddha Kembhavi. Webly supervised concept expansion for general purpose vision models. *arXiv preprint arXiv:2202.02317*, 2022. 8

[32] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017. 3

[33] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019. 6

[34] Alexander C Li, Alexei A Efros, and Deepak Pathak. Understanding collapse in non-contrastive siamese representation learning. In *European Conference on Computer Vision*, pages 490–505. Springer, 2022. 3

[35] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 8

[36] Elad Mezuman and Yair Weiss. Learning about canonical views from internet image collections. *Advances in neural information processing systems*, 25, 2012. 6, 11

[37] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 3

[38] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 8

[39] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018. 8

[40] 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. 6

[41] 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*, pages 4004–4012, 2016. 3

[42] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *preprint arXiv:1807.03748*, 2018. 3, 8

[43] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 6

[44] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021. 3

[45] 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*, pages 8748–8763. PMLR, 2021. 1, 3, 6, 7, 8

[46] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 4, 11, 13

[47] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020. 3

[48] 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*, pages 815–823, 2015. 3

[49] 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. *arXiv preprint arXiv:2210.08402*, 2022. 11

[50] 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. *arXiv preprint arXiv:2111.02114*, 2021. 8

[51] Burr Settles. Active learning literature survey. 2009. 2

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

[53] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015. 8

[54] Hardik Vasa. Google images download. `https://github.com/hardikvasa/google-images-download`, 2015. 2

[55] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021. 3, 11

[56] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995. 4

[57] 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*, pages 2840–2848, 2017. 3

[58] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *preprint arXiv:1708.03888*, 2017. 14

[59] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021. 3, 8

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 taken advantage of by practitioners. Nonetheless, we show that Internet Explorer is agnostic to the choice of text-to-image search engine and can still rapidly improve even when the data source is much noisier.

To test Internet Explorer in the most minimal setting, we build a custom search engine that finds images solely using their accompanying text, without using any pre-trained visual features whatsoever. We use the LAION-5B dataset [49], 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 [46] to compute 384-dimensional text embeddings for each caption. Then, we use Faiss [30] to compute a fast, approximate nearest-neighbors lookup index. Querying our custom search engine finds 100 image results in less than a second. Fig. 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. Fig. 10 shows that each data source has its own tendencies. For the "spaghetti bolognese" query, Google Images is biased [11, 36] 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.

Fig. 11 and Tab. 3 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, even with noisy search results and no hyperparameter tuning, Internet Explorer significantly improves the starting model in less than a day of searching and training. Overall, these results are a proof of concept that Internet Explorer can effectively utilize any window into the Internet's vast ocean of image data.

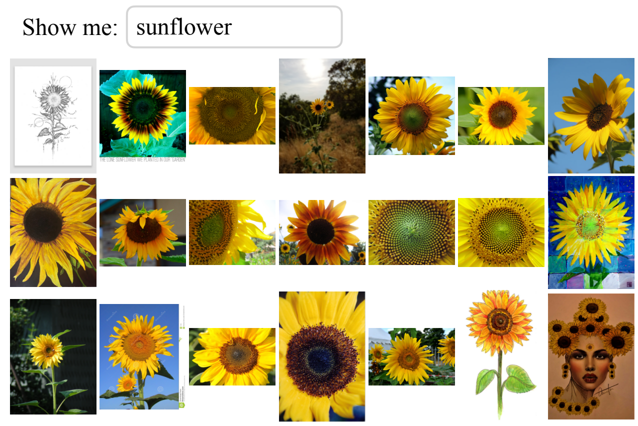

Figure 9. **Our custom LAION-5B search engine.** We build a custom text-to-image search engine that finds images within the LAION-5B dataset by doing nearest neighbor search in text embedding space. This uses no image features whatsoever.

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) [7] to compute hashes for the target dataset's training set, test set, and the downloaded data. We compare hashes to determine how many test images were leaked, and we report the number of collisions in Tab. 4. 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 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.

C. Method Details

C.1. GPT-J Descriptor Prompting

We use GPT-J-6B [55], 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
```

Figure 10. **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.

```
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 sam-

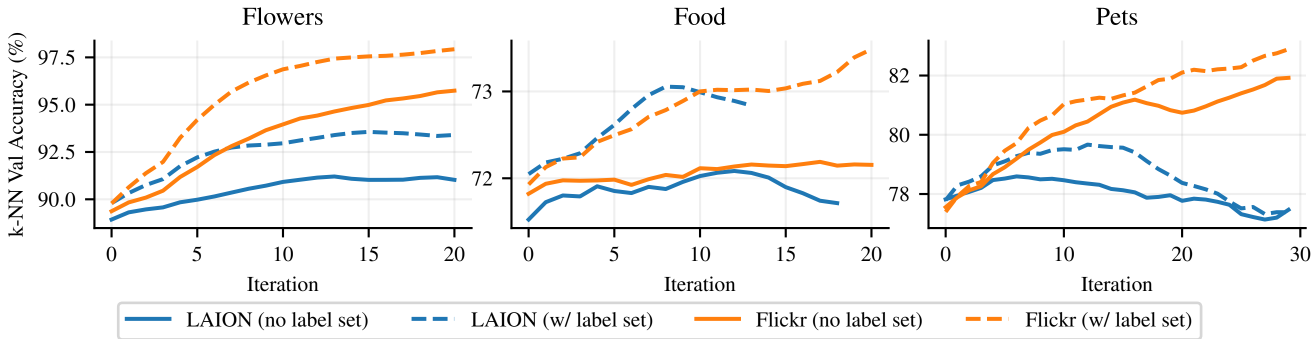

Figure 11. **Learning from Flickr and LAION-5B.** Even with the noisy search results returned by Flickr and LAION, Internet Explorer still continuously improves performance.

2*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**	**83.8**	**81.9**	**81.0**	**90.8**	**89.1**	**86.7**

Table 3. **Linear probe accuracy with other search engines**. Internet Curiosity improves its performance using any search engine, including Flickr and our custom text-only LAION search engine.

pling a descriptor, we prepend it to the concept and use the resulting phrase as our search query.

C.2. 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.

Semi-supervised vocabulary To reduce our search space in the semi-supervised 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 [46] 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 Tab. 5) 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

Model	Birdsnap	Flowers	Food	Pets	VOC2007	Images Downloaded
No exploration						
Target training set	$1/1849$	$5/6142$	$34/25246$	$21/3663$	$0/4952$	-
Internet Explorer						
Ours++ (no label set)	$28/1849$	$11/6142$	$35/25246$	$26/3663$	$1/4952$	$\approx 10^6$
Ours++ (with label set)	$57/1849$	$27/6142$	$35/25246$	$43/3663$	$1/4952$	$\approx 10^6$

Table 4. **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. We show (number of leaked images)/(number of unique test images). Surprisingly, the training sets of these datasets already leak a small fraction of the test sets. Leakage numbers for our methods include this train-test leakage, since our methods use 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.

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

Table 5. **Target Dataset "Category"**.

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 semi-supervised setting, including the random exploration methods.

C.3. 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.

C.4. Hyperparameters

Tab. 6 shows our hyperparameter values, which are shared across datasets. We perform minimal hyperparameter tuning and copy most of the values from the MoCo-v3 [15] 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.

Hyperparameter	Value
Architecture	Resnet-50 [25]
Optimizer	LARS [58]
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 6. **Online training hyperparameters**.

C.5. 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.

D. VOC2007 Linear Probe Accuracy

VOC categories are broad, unlike the fine-grained classes in Birdsnap, Flowers, Food, and Pets. Thus, comparing against ImageNet pre-training or CLIP would be unfair since these datasets are diverse, cleaned, class-balanced, and entirely relevant to VOC. Thus, we start from and compare against a MoCo-v3 checkpoint that has only been pre-trained on VOC2007. We report the linear probe accuracy for Pascal VOC2007 in Tab. 7. We find that Internet Explorer improves faster than randomly searching the Internet, especially when it has label set information to help guide it.

Model	VOC2007
Fixed dataset Self-Supervised	
MoCo-v3 [15] (VOC2007 pre-train)	58.0
No label set information	
Random exploration	64.8
Ours	63.4
Ours++	64.9
Use label set information	
Search labels only	60.8
Labels + semantically relevant terms	65.3
Ours	65.4
Ours++	**72.9**

Table 7. **Linear probe accuracy on VOC2007**.

E. 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, Fig. 12, Fig. 13, Fig. 14, and Fig. 15 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.

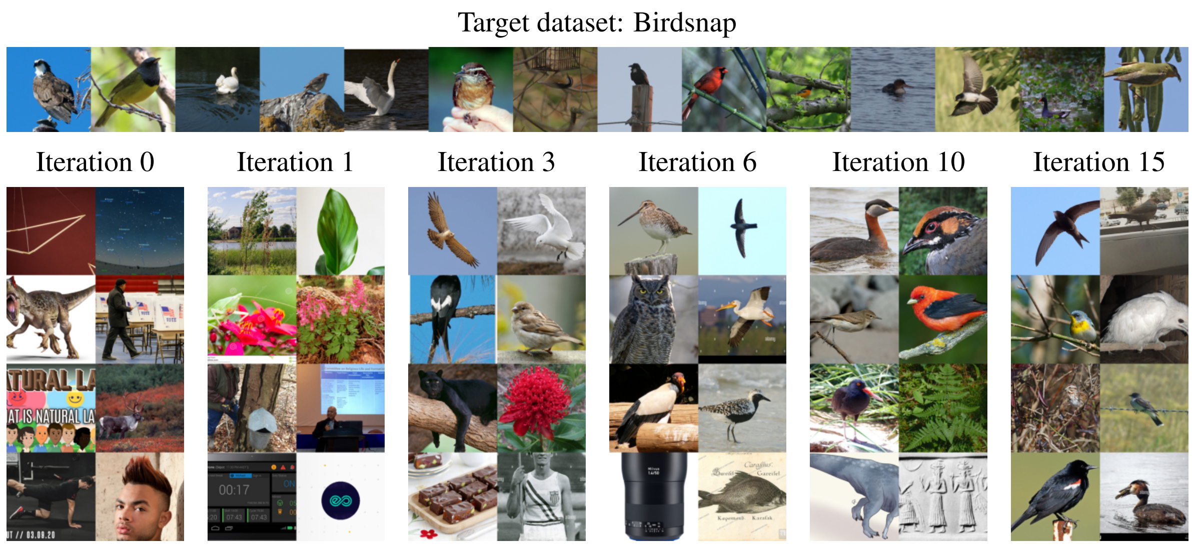

Figure 12. **Progression of downloaded Birdsnap images.** This corresponds to Ours++ without using label set information.

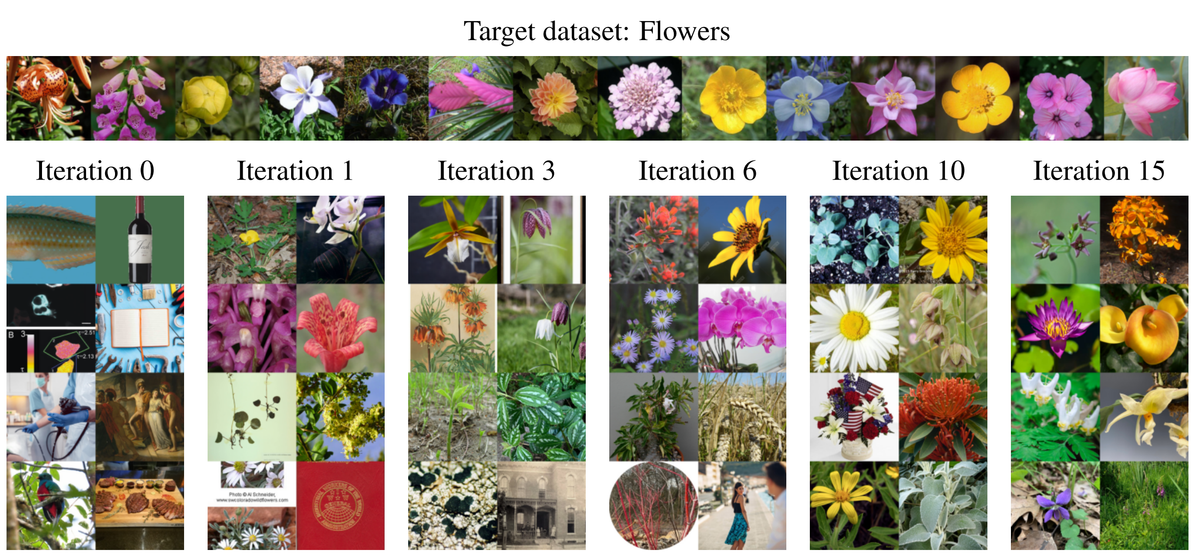

Figure 13. **Progression of downloaded Flowers images.** This corresponds to Ours++ without using label set information.

Figure 14. **Progression of downloaded Food images.** This corresponds to Ours++ without using label set information.

Figure 15. **Progression of downloaded VOC2007 images.** This corresponds to Ours++ without using label set information.