Supplementary Material: Learning Visual Representations via Language-Guided Sampling

Mohamed El Banani Karan Desai Justin Johnson
University of Michigan
{mbanani,kdexd,justincj}@umich.edu

A. Evaluation Tasks

We compare all models by evaluating the encoder's frozen features on two downstream classification tasks: linear probe and fewshot. We chose to use simple classifiers since they allow us to evaluate the features as-is and conduct a comprehensive hyperparameter sweep to ensure a fair comparison. We explain the two evaluation setups in more detail below. Our implementation can be found at https://github.com/mbanani/lgssl.

Linear Probe Classification

We follow the linear probe evaluation proposed by Kornblith *et al*. [18] of training a logistic regression classifier using the L-BFGS optimizer [22]. We follow prior work [18, 28] and perform a hyperparameter sweep over the cost values in the logistic regression loss. We sweep over 96 values in log space from 10^{-6} to 10^{6}. During the hyperparameter sweep, we train on the train split and validate on the valid split. We choose the cost value with the best validation performance and train a final classifier on the combined train and validation instances. We use the PyTorch [27] implementation of L-BFGS with all the default parameters except for the maximum number of iterations, which is set to 1000 similar to CLIP [28]. Our evaluation metric depends on the dataset, as shown in Tab. 1, to account for class imbalance.

Few-Shot Classification

We also use fewshot classification as an evaluation for frozen features. Prior work [34, 38] has shown that simple classifiers on top of frozen features are strong baselines for fewshot classification. More specifically, Wang *et al*. [38] shows that when features are normalized (mean subtraction and L2 normalization), a nearest neighbor classifier is a very effective and strong baseline for fewshot classification. Inspired by these results, we use a simple weighted nearest neighbor classifier to evaluate pre-trained frozen features. We set k to be the size of the support set and classify the features as follows:

$$y' = \arg\max_{v} \sum_{(I,y)\in\mathbb{D}_{\text{support}}} \mathbb{1}_{[v=y]} \text{sim}(f(I'), f(I)) \quad (1)$$

where $\mathbb{1}_{[v=y]}$ is an indicator variable that is 1 if y is the same class as v and 0 other wise, $\text{sim}(\cdot,\cdot)$ is cosine similarity between two vectors, f is the visual encoder, I' is the target image, $\mathbb{D}_{\text{support}}$ is the support set.

We adopt 5-way, 5-shot classification as our fewshot classification task. We sample five random classes for each episode and then sample five images for each class in the training set, resulting in 25 labeled training images. We also sample 5 images for each class from the test set as our test images. We use all available test images for classes with less than five test images for that class. This is primarily an issue for Caltech-101 [13]. We sample 5000 episodes and compute the average test accuracy across all episodes. We experimented with increasing the number of episodes to 50000 to improve evaluation but noticed little change in the mean performance. We also report the 95% confidence interval for each dataset.

B. Evaluation Datasets

We list all the evaluation datasets used in Tab. 1. We use TensorFlow datasets for evaluation to ensure easy replication [33]. For all datasets, we preprocess the images by resizing the image so that its smaller dimension is 224 using bilinear interpolation followed by a center crop to 224×224. We use bilinear interpolation since improves performance on low-resolution datasets such as CIFAR-10 and CIFAR-100. We normalize the images using ImageNet's mean and standard deviation for pixel values for all models except for pre-trained CLIP. For CLIP, we use their provided mean and standard deviation values as they greatly impact performance: an average gain of approximately 4% for linear probe evaluation. We exclude Patch Camelyon from the fewshot evaluation since it is a binary classification dataset.

Table 1. **Evaluation Datasets.** Orange rows indicate datasets that do not have an official validation split; we constructed one by randomly holding out 20% of the official train split. Blue rows indicate datasets that do not officially define splits; we randomly sample instances to construct non-overlapping splits. For ImageNet, the validation set is used as the test set, and we construct a validation set by randomly holding out 20% of the official training split. ImageNet variants, highlighted in green, are all test sets for models trained on ImageNet.

Dataset	Classes	Train	Val	Test	Metric
Food-101 [1]	101	60600	15150	25250	accuracy
CIFAR-10 [20]	10	40000	10000	10000	accuracy
CIFAR-100 [20]	100	40000	10000	10000	accuracy
CUB-2011 [39]	200	5795	1199	5794	accuracy
SUN397 [41]	397	15880	3970	19849	accuracy
Stanford Cars [19]	196	6515	1629	8041	accuracy
FGVC Aircraft [23]	100	3334	3333	3333	mean-per-cls
DTD [8]	47	1880	1880	1880	accuracy
Oxford-IIIT Pets [26]	37	2944	736	3669	mean-per-cls
Caltech-101 [13]	102	2448	612	6084	mean-per-cls
Oxford Flowers [25]	102	1020	1020	6149	mean-per-cls
STL-10 [9]	10	4000	1000	8000	accuracy
EuroSAT [15]	10	5000	5000	5000	accuracy
RESISC45 [7]	45	3150	3150	25200	accuracy
Patch Camelyon [35]	2	262144	32768	32768	accuracy
ImageNet [10]	1000	1024934	256233	50000	accuracy
ImageNet A [17]	200	N/A	N/A	7500	accuracy
ImageNet R [16]	200	N/A	N/A	30000	accuracy
ImageNet v2 [29]	1000	N/A	N/A	10000	accuracy
ImageNet Sketch [36]	1000	N/A	N/A	50889	accuracy

We also include statistics for the ImageNet dataset evaluations done in Appendix G.

C. Baselines

For fair evaluation, we retrained previous methods from scratch with several methods reimplemented. We also provided several system-level comparisons using pre-trained checkpoints provided by prior work. Below, we provide additional details on our baselines.

Pre-trained model checkpoints

We use publicly available checkpoints of various pre-trained models for sampling and experimental comparisons:

- **SBERT [30]:** We use two checkpoints from SBERT: `all-mpnet-base-v2` (MPNet backbone [31]), and `all-MiniLM-L12-v2` (MiniLM backbone [37]). Those models were used for sampling, while MPNet was also used as a frozen backbone in analysis experiments.
- **CLIP [28]:** We use checkpoints available in the official Github repository[1] for both system-level comparisons and sampling. We use the `RN50` checkpoint in the system-level comparisons to match the backbone for other models. We use the `ViT-B/32` checkpoint for sampling to provide the strongest visual sampling performance in evaluating different sampling modalities.
- **ImageNet pre-trained model:** We use the checkpoints provided by `torchvision` package.[2] For system-level comparisons, we use the ResNet-50 `IMAGENET1K_V2` checkpoint [40] as it achieves better performance than the original ResNet-50 checkpoint [14]. We also use `ViT-B/32` [11] checkpoint to compare with the sampling strategy using a CLIP checkpoint.
- **SimCLR [4]:** We use the SimCLR v2 checkpoint provided by PyTorch Lightning Bolts.[3] While SimCLR released some checkpoints for TensorFlow, we found that converting them to PyTorch using the recommended tools resulted in lower performance. We use the same checkpoint for both sampling and system-level comparison. Note that SimCLR only released models trained for 800 epochs.
- **SimSiam [3]:** We use the checkpoint trained with 512 batch size from the official Github repository[4] as it more closely matches our training setup.
- **MoCo [6]:** We use the official checkpoint for MoCo v3.[5] We use the checkpoint for the model trained for 100 epochs to match other checkpoints more closely.
- **SwAV [2]:** We use the official SwAV checkpoint.[6] We use the checkpoint trained for 100 epochs to match the training duration of other methods. Unlike our implementation, the full SwAV model is trained using Multi-Crop augmentation strategy.

Retrained models

We reimplement and retrain all baselines. When an official implementation was available, we adapted it to fit within our pipeline. For all models, we use a ResNet-50 backbone from torchvision with random initialization and a feature dimension of 2048 (the `fc` layer is removed). We use a linear layer or a multi-layer perceptron (MLP) for projection layers. Every layer except for the last is followed by batch normalization and a ReLU non-linearity. We describe an N-layer MLP with $N + 1$ numbers depicting the input dimension for the first layer, followed by the output dimension for all layers.

We use two forms of augmentation: SimCLR or global crop. Global crop consists of a random resized square crop

[1] https://github.com/openai/CLIP

[2] https://github.com/pytorch/vision

[3] https://lightning-bolts.readthedocs.io/

[4] https://github.com/facebookresearch/simsiam

[5] https://github.com/facebookresearch/moco-v3

[6] https://github.com/facebookresearch/swav

with a scale of (0.5, 1.0) to an image size of 224×224. SimCLR augmentations consist of random resized square crop, color jittering, random grayscale, random horizontal flipping, and Gaussian blur. We use the same augmentation parameters as prior work [4, 6]. All images are normalized using ImageNet's mean and standard deviation statistics.

We provide baseline-specific details below and refer the reader to our implementation for more details:

- **SimCLR:** We use a 3-layer MLP as a projection layer with feature dimensions (2048, 2048, 2048, 128) similar to the original paper [4]. We use the SimCLR loss implementation from Mu *et al*. [24], which adapts the original loss for the distributed settings for inference and gradients. We use SimCLR augmentations for SimCLR and global crop augmentations for LGSimCLR. We experimented with mixing SimCLR augmentation and language sampled pairs and found that it results in slightly inferior performance: adding augmentations reduces the linear probe average accuracy from 78.3 to 77.9.

- **CLIP:** We use a linear projection layer to a feature dimension of 512 similar to the original paper. We use the smallest CLIP language encoder, similar to SLIP [24], with a feature dimension of 512 and a linear language projection layer. We use the loss implementation from SLIP [24] but adapt it to share the loss gradients similar to the SimCLR loss. We use global crop augmentation for CLIP since SLIP [24] reported that it performs better than CLIP's original center crop preprocessing.

- **SLIP:** We follow SLIP's implementation and combine the augmentations, projections, and losses from SimCLR and CLIP. We use the same language transformer as our CLIP implementation. We generate two augmented views with SimCLR augmentation for the SimCLR loss and one with global cropping for the CLIP loss. Those views are passed through their respective projections (3-layer MLP for SimCLR and linear projection for CLIP) and losses. For LGSLIP, we only use the global crop augmentation, resulting in only two augmented views and forward passes through the encoder instead of 3 for SLIP. We apply the SimCLR loss between the language-sampled image pair and the CLIP loss between only one of the images and its caption.

- **SimSiam:** We follow the original SimSiam implementation and use a 3-layer MLP as our projection head (2048, 2048, 2048, 2048) and a 2-layer MLP as our prediction head (2048, 512, 2048). We use the loss formulation from the original paper. For LGSimSiam, we use the same formulation but use global crop instead of SimCLR augmentations.

- **SwAV.** We follow the original SwaV implementation and use a 2-layer MLP (2048, 2048, 128) as our projection head and a linear layer as our prototype head with an output dimension of 3000. The prototypes are initially frozen to improve training dynamics as suggested by the SwAV repository. We use the distributed Sinkhorn clustering implementation from the official code release.

- **NNCLR.** We rely on the implementation of NNCLR provided by Lightly [32] since NNCLR [12] did not release an implementation. Specifically, we use the memory bank implementation from Lightly and reimplement NNCLR. While our NNCLR implementation outperforms SimCLR on ImageNet, as reported in the paper, it underperforms on RedCaps. We use a 3-layer MLP (2048, 2048, 2048, 256) as our projection head and a 2-layer MLP (256, 4098, 256) as our prediction head. We also use a queue of length 16384 (equivalent to 32 batches) for the memory bank. For Language NNCLR, we also use the memory bank from Lightly, similar to DeCLIP [21]. We augment the CLIP implementation with a memory bank for the language encoder. We use a weighting of 0.8 for the CLIP loss and 0.2 for the language NNS loss, similar to DeCLIP [21].

Table 2. **Batch Size Scaling**. The performance of both SimCLR and LGSimCLR scales with larger batch sizes with the LGSimCLR outperforming SimCLR even for larger batch sizes.

	SimCLR		LGSimCLR	
Batch Size	Linear	Fewshot	Linear	Fewshot
256	67.5	67.2	77.7	82.3
512	68.5	66.7	78.2	82.5
1024	69.3	68.4	78.6	82.6
2048	69.8	68.6	79.1	83.1

D. Batch Size Scaling

We explore the scaling performance of our approach for batch size. Prior work has shown that contrastive methods can benefit larger batch sizes [3, 4, 28]. While we use a batch size of 512 to allow us to perform comprehensive experiments and evaluations given limited compute, we conduct a few experiments to evaluate scaling for SimCLR and LGSimCLR. Our results, shown in Tab. 2, indicate that our approach scales with batch size and maintains our performance gains over SimCLR for larger batch sizes. Furthermore, we show in **??** that our model benefits from larger datasets. Our experiments also show that data scaling comes in two ways: training on more instances and sampling nearest neighbors from a larger pool of images. We explore this more in Appendix E.

E. Dataset Size Scaling

Scaling up the dataset size not only increases the training instances, but also *broadens* the scope to sample nearest neighbors from. In this section, we study the impact of data scaling on model performance. First, we compare our RedCaps-trained LGSimCLR model with another model trained using a subset of RedCaps instances belonging to the year 2020. We also train multiple LGSimCLR models using RedCaps, each having a restricted scope of nearest neighbor sampling. RedCaps has a natural structure to make this possible: instances (posts) are grouped in different subreddits, across multiple years. We expect instances within the same year to be weakly related, and instances within a subreddit to share a consistent theme; *e.g.*, **`r/food`** has images of food dishes with text describing main ingredients. We consider three restricted sampling variants of RedCaps: Year, Subreddit, and Subreddit-Year.

We hypothesize that Subreddit sampling will limit the pool of nearest neighbor sampling to images within a similar domain resulting in higher quality neighbors. In contrast, the Year variant will only limit the number of images to consider, reducing the probability of finding images with similar captions as neighbors. While posts within the same year might be related to major events (*e.g.* COVID-19 pandemic increased the proportion of indoor images in year 2020), the relationship is much weaker than domain-specific subreddits. Additionally, Subreddit-Year, which only samples the nearest neighbors from the same subreddit posted in the same year, will combine both effects.

Our results, presented in Tab. 3, show that domain-specific sampling improves performance. Meanwhile, more random sampling minimally degrades performance. Finally, restricting the scope of the nearest neighbor sampling is not the same as subsampling the data. This is shown by the higher performance of RedCaps with Year sampling compared to RedCaps-2020. Those results indicate two opportunities: First, our approach can scale to very large datasets by only performing nearest-neighbor searches within subsets of the data. This is especially beneficial in some domains, such as federated learning. Second, identifying other domain structures within the dataset can improve performance by allowing the model to sample nearest neighbor images within the same domain.

This result indicates that our approach could scale to gigantic datasets without requiring the nearest neighbor search over the full dataset. While identifying semantically-related partitions in the datasets could improve performance, the model could perform very well by splitting randomly or using metadata information such as year or location, which might provide some relevant, although very weak, structure.

Table 3. **Impact of Sampling Scope**. LGSimCLR can still learn good features if it is restricted to only sampling from a subset of the dataset. Domain-specific partitioning (*e.g.*, subreddits) improves performance, while domain agnostic partitions (*e.g.*, Year or Subreddit-Year) minimally degrades performance.

Dataset	Sampling Scope	#Partitions	Linear	FewShot
RedCaps 2020	All	1	73.8	78.8
RedCaps	All	1	78.2	82.5
	Year	4	77.2	80.2
	Subreddit	350	79.0	80.5
	Subreddit-Year	1391	77.6	78.2

F. Qualitative Analysis of Sampled Pairs

We present language-sampled nearest neighbors for different datasets in Fig. 1. We note that CC3M and CC12M have many stock images with slightly robotic descriptions, which is explained by how those datasets were collected. For example, see Fig. 1 top row: the caption indicates 'animal' instead of referring to the dog in the image. Meanwhile, RedCaps captions can be more descriptive, referring to pet names or specific product brands or models. Empirically, we find that RedCaps results in better performance.

We also observe some repeated patterns in the types of nearest neighbor captions we get. We identify four patterns, shown in Fig. 2, and discuss them below:

Similar objects in different contexts: The first set of results shows examples where language-guided sampling results in diverse images depicting the same concept. For example, while pairs sampled using visual models (regardless of whether they are self-supervised or supervised) depict shoes on their own, language samples three images of the same shoe model in very different contexts. The third row also depicts hummingbirds in very different poses. At the same time, self-supervised models provide three birds on a branch, and supervised models provide three hummingbirds taken in similar poses as the source image.

Visual similarity misses the object: The second set shows examples where visual similarity misses the salient object in the image. The fourth row is a halibut dish with vegetables. Visual sampling results in other dishes pictured from the top, while language sampling gives us three other halibut dishes with vegetables that look different from the source image. Rows 5 and 6 show examples where visual sampling focused on the overall appearance and missing the herb scissors (row 5) and coyote (row 6). Self-supervised models provide the nearest neighbors with animals in the snow, but different animals like a lynx or a dog.

Captions capture subtle relationships: The third set shows examples where the language captures subtle relationships. Can you guess what the captions were? In row 7, the source image was captioned "*itap of a tunnel cre-*

ated by the autumn leaves." Visual similarity focuses on the trees, while language similarity results in images depicting autumn more clearly. In row 8, the source caption mentions a cheetah which can be seen at the right corner of the source image, but the overall sunset appearance results in different sets of visual nearest neighbors. Finally, the caption for row 9 mentions a mating ritual between birds. This element is captured by language guidance, while visual similarity retrieves images of animals in the grass. These results suggest that conditioning the model similarity on the caption could result in a better-posed learning problem.

Vague Captions: Those examples show cases where the caption is very vague or unrelated to the image content, resulting in odd nearest neighbors in the language space. Can you guess the captions from the nearest neighbor images? Answers are in the footnote.[7] The caption of row 10 refers to the appearance of the eyes of the penguin, but since the "googly eyes" can also refer to a small toy, it retrieves images of that toy being used on a coffee machine and a wall. In row 11, the caption asks what the object is, but this is independent of the object. This results in language retrievals with miscellaneous objects, while visual retrievals return other insects. Finally, row 12 shows a case where the retrieval uses the dog's name in some context, resulting in the retrieval of other pets playing in gardens. These cases represent limitations of language sampling that might result in poor learning. However, since the core issue arises from misalignment or vagueness in the caption, it is a limitation shared by any model that uses captions and images.

G. Additional Results

Due to space limitations, we only report average performance for several methods in the main paper. Here, we report the complete performance breakdown for all methods on linear probe in Tab. 4 and fewshot classification Tab. 5. For fewshot classification, we also report the 95% confidence interval as a subscript.

We also evaluate all approaches on several ImageNet evaluation benchmarks. We use the same evaluation setups described in Appendix A and report results in Tab. 6 Specifically, we train on the ImageNet train set and evaluate on the ImageNet validation set [10], and several alternative ImageNet test splits that assess robustness. ImageNet A(dverserial) [17], ImageNet R(enditions) [16], ImageNet v2 [29], and ImageNet Sketch [36]. We observe similar performance trends for the RedCaps-trained models, with LGSimCLR outperforming all baselines, LGSLIP outperforming LGSimCLR, and training on larger datasets or with larger batch sizes improving performance. We also note that the difference in performance between our models and the ImageNet pre-trained checkpoints is larger due to the smaller domain shift they experience from ImageNet training.

[7] Source Image Captions:
row 10: "*Built-in googly eyes.*"
row 11: "*I found this today. Anyone knows what it is?*"
row 12: "*Cinda having fun in the garden!*"

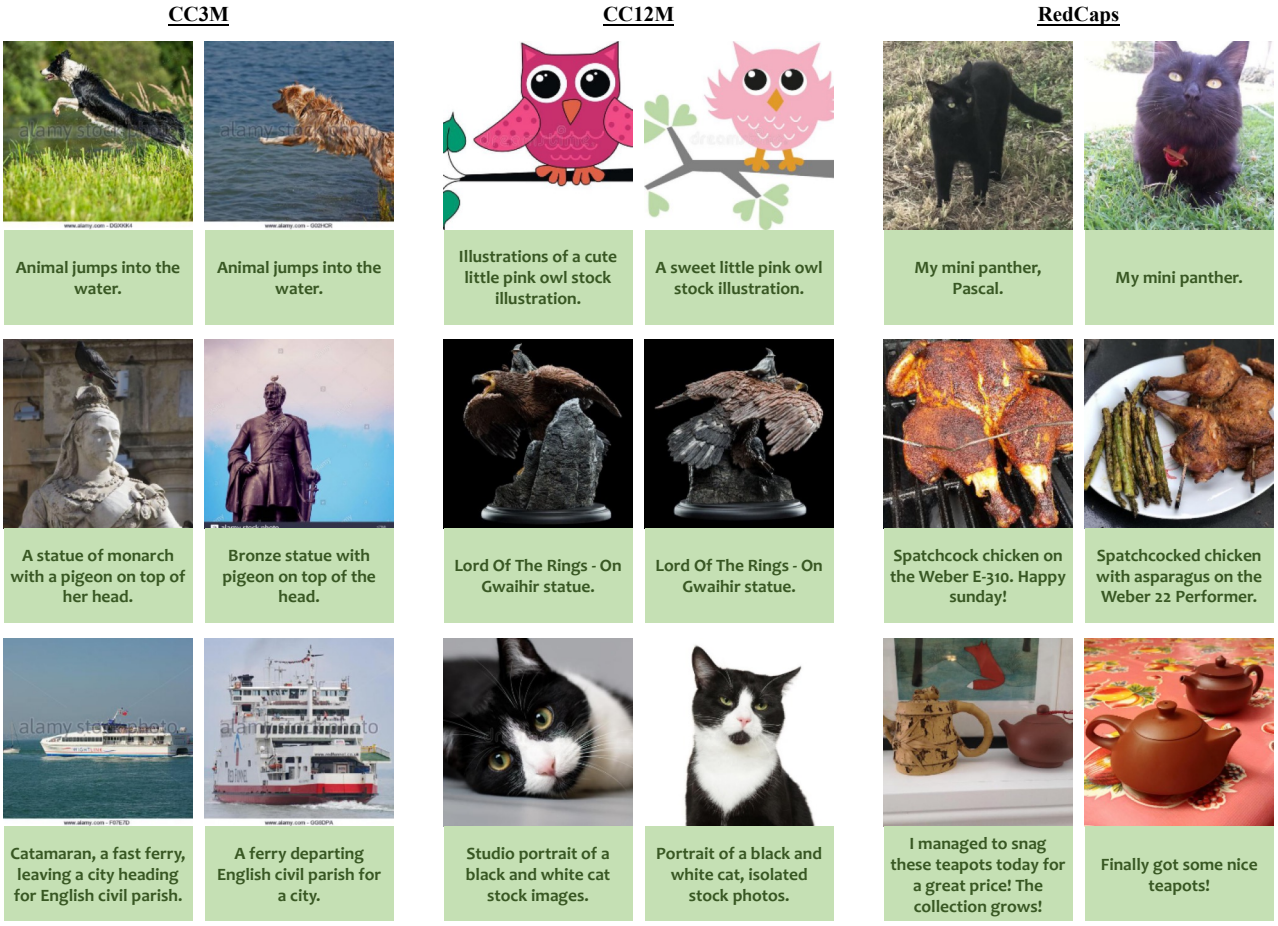

Figure 1. **Language sampled image pairs for the different pretraining datasets.** We show qualitative results for language guided sampling for our pretraining datasets. While the captions for conceptual captions can be generic, RedCaps captions are more natural; including longer and more natural descriptions as well as irrelevant details.

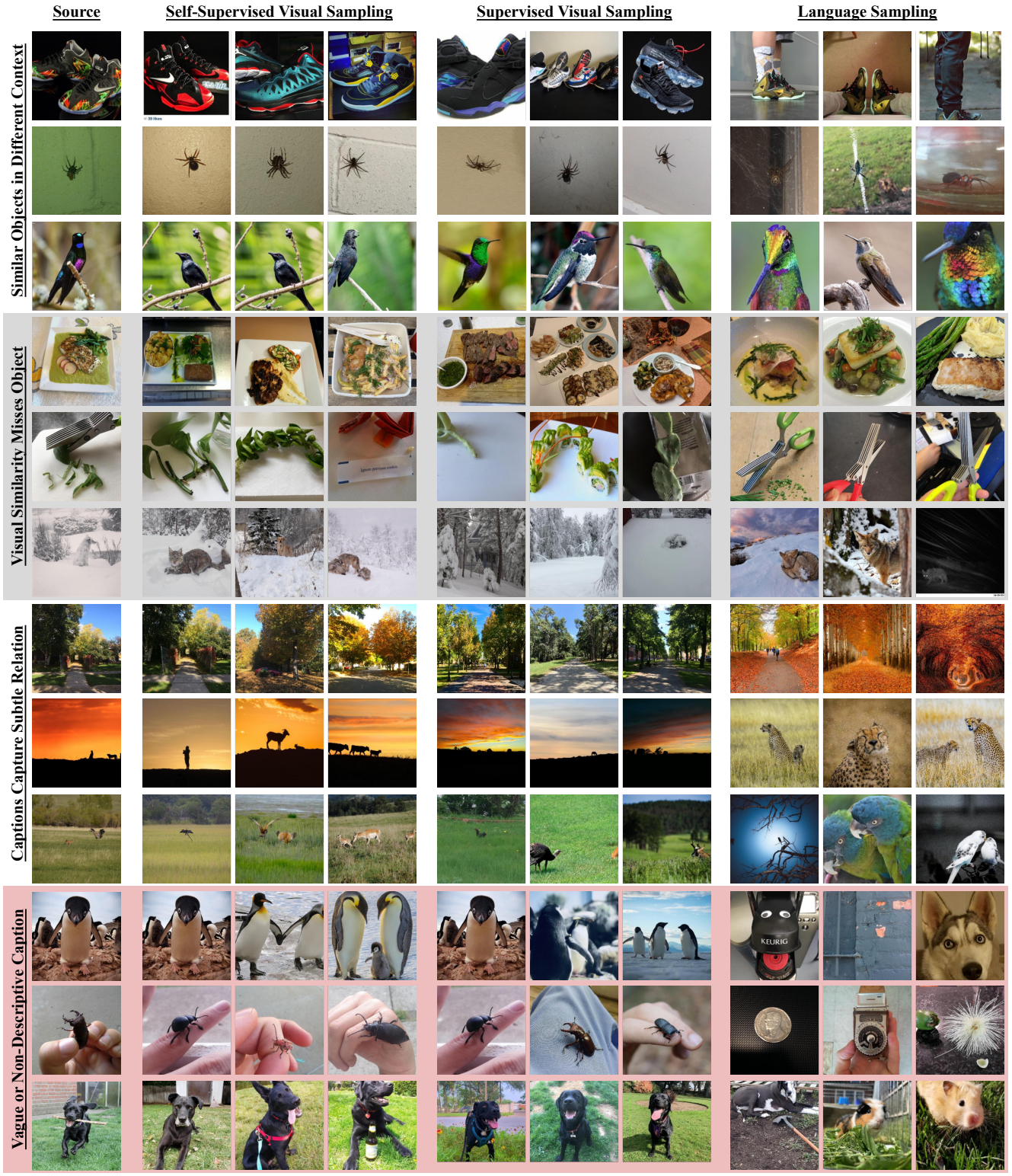

Figure 2. **Nearest Neighbors on RedCaps for multiple sampling options.** We sample the three nearest neighbors using a self-supervised visual model [4], an ImageNet supervised model [11], and a self-supervised language model [30]. The examples are representative of some patterns we observe in the sampled pairs. Language guided sampling allows us to get pairs that depict similar objects in different poses and contexts in ways that go beyond visual sampling. However, sometimes the relationships depicted in the language can be too subtle. Furthermore, sometimes captions are noisy resulting in unrelated language-sampled pairs.

Table 4. **Linear Probe Evaluations.** We report the linear probe classification performance of all baselines and models. Models are grouped by experiment.

Model	Dataset	Sampling Space	Food-101	CIFAR-10	CIFAR-100	CUB	SUN397	Cars	Aircraft	DTD	Pets	Caltech-101	Flowers	STL-10	EuroSAT	RESISC45	PCAM	Avg.
Pre-trained Checkpoints																		
Supervised [40]	ImageNet	-	71.0	93.2	77.0	68.4	63.0	48.7	41.0	73.0	92.6	91.9	88.3	98.2	95.8	85.3	82.4	78.0
SimSiam [5]	ImageNet	-	70.6	92.2	74.9	47.1	0.3	52.4	52.6	74.4	83.3	89.3	91.8	95.9	96.7	86.4	85.2	72.9
MoCo v3 [6]	ImageNet	-	71.4	93.3	77.9	51.5	60.4	52.4	53.0	73.6	85.8	90.4	92.1	96.8	96.3	85.0	85.0	77.7
SwAV [2]	ImageNet	-	72.8	93.0	77.5	48.8	63.2	55.5	52.7	77.2	84.5	89.9	93.4	97.2	96.7	86.9	83.7	78.2
SimCLR [4]	ImageNet	-	71.4	91.3	73.9	44.3	60.3	44.6	46.7	74.9	83.9	87.4	90.2	96.2	95.9	84.4	85.1	75.4
CLIP [28]	CLIP (400M)	-	86.4	88.7	70.2	69.8	72.5	78.4	49.4	76.3	88.0	88.9	96.1	97.2	94.7	87.9	82.7	81.8
RedCaps-trained Baselines																		
SwAV	RedCaps	-	63.6	81.3	57.5	21.6	47.5	22.9	35.4	68.1	61.1	70.5	78.0	87.7	94.3	79.9	84.3	63.6
SimSiam	RedCaps	-	64.1	79.9	56.1	28.2	48.3	29.5	41.2	66.2	69.1	73.6	83.6	85.7	94.4	82.1	83.3	65.7
SimCLR	RedCaps	-	69.0	82.9	61.6	30.6	52.6	33.7	43.7	69.8	70.5	74.1	86.9	88.0	95.4	84.6	84.4	68.5
Visual NNCLR	RedCaps	-	65.4	82.8	60.2	26.6	50.0	26.6	40.9	68.0	65.2	75.4	83.5	88.5	95.3	82.2	83.8	66.3
CLIP	RedCaps	-	80.9	84.7	62.7	50.4	57.4	45.8	36.7	67.6	79.8	84.0	91.0	93.5	93.9	82.2	82.6	72.9
CLIP (SBERT Encoder)	RedCaps	-	80.5	81.3	59.4	50.6	56.9	45.9	35.7	69.1	76.7	81.7	90.2	93.6	92.9	81.1	81.3	71.8
Language NNCLR	RedCaps	-	81.2	83.1	61.9	48.6	56.5	45.1	37.2	68.8	78.1	82.0	90.2	93.4	92.5	81.1	80.7	72.0
SLIP	RedCaps	-	77.7	87.2	67.0	42.4	58.1	48.7	45.2	72.3	79.5	82.7	92.1	92.7	95.6	85.5	83.4	74.0
Sampling Space - Language																		
LGSimCLR	RedCaps	SBERT (MiniLM)	83.2	88.0	69.3	60.4	59.7	64.0	54.0	72.7	82.6	88.5	95.7	94.1	96.4	88.1	82.2	78.6
LGSimCLR	RedCaps	CLIP (400M)	83.3	87.6	68.9	60.1	59.9	62.9	53.7	70.5	82.6	88.7	95.6	94.3	96.2	88.2	82.0	78.3
LGSimCLR	RedCaps	CLIP (RedCaps)	83.7	88.0	67.8	59.6	60.7	60.8	53.7	71.4	82.4	89.1	95.9	93.8	96.1	88.4	82.7	78.3
LGSimCLR	RedCaps	FastText BoW	80.8	85.5	66.7	54.2	58.7	56.6	51.1	69.9	78.3	88.0	94.4	92.5	96.2	87.7	81.5	76.1
Sampling Space - Visual																		
LGSimCLR	RedCaps	ImageNet Supervised	75.7	92.2	75.4	57.5	60.2	53.7	52.2	71.7	90.3	90.2	93.1	95.5	96.8	87.7	83.0	78.3
LGSimCLR	RedCaps	SimCLR	71.4	87.0	67.5	36.8	57.9	41.8	46.3	74.2	82.8	82.6	90.7	93.4	95.7	85.2	83.2	73.1
LGSimCLR	RedCaps	CLIP (400M)	83.6	90.7	72.1	58.3	62.5	59.2	51.3	75.5	88.7	90.3	95.2	95.4	96.2	88.6	82.9	79.4
Sampling Scope																		
LGSimCLR	RedCaps	SBERT - Year	82.6	85.8	66.1	58.1	59.0	57.1	52.8	71.9	80.7	88.1	95.6	93.1	96.1	88.0	82.8	77.2
LGSimCLR	RedCaps	SBERT - Sub-Year	82.4	86.8	66.9	56.4	59.5	54.5	51.4	72.1	89.0	89.8	94.9	95.2	95.8	88.1	81.9	77.6
LGSimCLR	RedCaps	SBERT - Sub	83.2	88.7	69.5	60.4	59.9	60.0	53.0	72.4	89.7	90.3	96.2	94.8	96.2	88.4	82.2	79.0
Pre-training Datasets																		
LGSimCLR	CC3M	SBERT (MPNet)	64.4	84.6	65.4	44.4	59.1	41.9	46.8	66.0	70.7	83.9	91.2	91.6	95.6	86.1	80.5	71.5
LGSimCLR	CC12M	SBERT (MPNet)	73.4	88.6	70.1	50.4	66.0	58.7	52.4	72.6	79.0	88.3	92.6	94.5	95.6	87.5	81.6	76.8
LGSimCLR	RedCaps 2020	SBERT (MPNet)	77.8	84.3	64.5	53.9	53.9	51.7	48.1	66.4	76.2	83.9	93.9	89.9	95.4	86.4	81.4	73.8
Batch Size Scaling																		
SimCLR (256)	RedCaps	-	67.1	83.1	60.5	28.5	51.0	32.5	42.4	70.0	68.3	73.9	85.8	86.5	96.2	84.2	83.1	67.5
SimCLR (1024)	RedCaps	-	70.0	84.4	62.8	31.9	52.4	35.8	44.2	70.9	72.7	74.7	87.9	88.3	95.6	84.8	83.3	69.3
SimCLR (2048)	RedCaps	-	70.4	83.9	62.6	32.5	53.3	36.7	44.9	70.9	73.1	75.5	88.1	88.8	96.5	85.1	84.2	69.8
LGSimCLR (256)	RedCaps	SBERT (MPNet)	82.9	87.3	68.0	58.7	60.2	58.2	52.6	73.2	81.1	88.2	95.2	94.0	96.1	87.7	81.8	77.7
LGSimCLR (1024)	RedCaps	SBERT (MPNet)	83.7	87.1	68.1	62.0	60.7	63.4	53.7	73.4	80.8	89.8	95.7	93.7	95.7	88.3	82.6	78.6
LGSimCLR (2048)	RedCaps	SBERT (MPNet)	84.2	88.2	69.1	63.2	60.9	65.2	55.5	71.4	81.7	89.7	96.0	94.4	96.3	88.5	82.1	79.1
Alternative Formulations																		
LGSimCLR	RedCaps	SBERT (MPNet)	83.2	87.8	69.0	59.3	60.3	62.3	53.4	71.2	81.8	89.4	95.9	94.0	95.6	88.0	81.1	78.2
LGSimSiam	RedCaps	SBERT (MPNet)	73.8	83.4	62.6	40.6	54.6	41.1	47.3	68.6	66.5	85.2	90.3	90.8	95.7	85.6	81.3	71.2
LGSLIP	RedCaps	SBERT (MPNet)	84.5	87.4	69.2	60.7	62.3	62.2	52.5	73.1	83.1	90.2	96.3	94.8	95.3	88.4	82.7	78.8

Table 5. **Few-Shot Evaluations.** We report the 5-way, 5-shot classification performance of all baselines and models. Models are grouped by experiment. The subscript reports the 95% confidence interval in prediction across 5000 episodes.

Model	Dataset	Sampling Space	Food-101	CIFAR-10	CIFAR-100	CUB	SUN397	Cars	Aircraft	DTD	Pets	Caltech-101	Flowers	STL-10	EuroSAT	RESISC45	Avg.
Pre-trained Checkpoints																	
Supervised [40]	ImageNet	-	$81.6_{0.3}$	$84.1_{0.2}$	$87.8_{0.2}$	$91.9_{0.2}$	$95.0_{0.1}$	$75.7_{0.3}$	$53.1_{0.4}$	$80.4_{0.3}$	$97.6_{0.1}$	$97.4_{0.1}$	$91.4_{0.2}$	$95.2_{0.1}$	$83.6_{0.2}$	$85.2_{0.3}$	85.7
SimSiam [5]	ImageNet	-	$70.5_{0.3}$	$77.5_{0.3}$	$82.0_{0.3}$	$67.4_{0.4}$	$92.1_{0.2}$	$51.9_{0.3}$	$43.7_{0.4}$	$81.8_{0.3}$	$86.7_{0.3}$	$94.9_{0.2}$	$93.7_{0.2}$	$89.7_{0.2}$	$87.7_{0.2}$	$82.2_{0.3}$	78.7
MoCo v3 [6]	ImageNet	-	$72.7_{0.3}$	$82.3_{0.2}$	$84.9_{0.3}$	$74.7_{0.3}$	$92.5_{0.2}$	$52.0_{0.4}$	$42.4_{0.4}$	$80.7_{0.3}$	$89.1_{0.2}$	$95.8_{0.1}$	$93.6_{0.2}$	$91.5_{0.2}$	$86.3_{0.2}$	$83.0_{0.3}$	80.1
SwAV [2]	ImageNet	-	$68.3_{0.3}$	$78.1_{0.3}$	$82.1_{0.3}$	$65.4_{0.4}$	$93.7_{0.2}$	$52.7_{0.4}$	$40.3_{0.4}$	$83.8_{0.2}$	$83.8_{0.3}$	$94.5_{0.2}$	$93.4_{0.2}$	$91.2_{0.2}$	$88.0_{0.2}$	$83.8_{0.3}$	78.5
SimCLR [4]	ImageNet	-	$70.0_{0.3}$	$76.9_{0.3}$	$80.9_{0.3}$	$67.5_{0.4}$	$92.5_{0.2}$	$51.9_{0.3}$	$42.1_{0.4}$	$82.2_{0.3}$	$85.0_{0.3}$	$93.0_{0.2}$	$90.3_{0.2}$	$88.8_{0.2}$	$83.6_{0.3}$	$78.5_{0.3}$	77.4
CLIP [28]	CLIP (400M)	-	$92.1_{0.2}$	$76.3_{0.3}$	$79.2_{0.3}$	$92.9_{0.2}$	$96.9_{0.1}$	$93.3_{0.2}$	$73.2_{0.4}$	$81.8_{0.3}$	$86.1_{0.3}$	$95.9_{0.1}$	$97.9_{0.1}$	$95.6_{0.1}$	$77.5_{0.3}$	$90.5_{0.2}$	87.8
RedCaps-trained Baselines																	
SwAV	RedCaps	-	$64.5_{0.4}$	$54.0_{0.3}$	$61.8_{0.3}$	$45.8_{0.4}$	$84.9_{0.3}$	$36.5_{0.3}$	$34.1_{0.3}$	$74.8_{0.3}$	$66.5_{0.4}$	$78.1_{0.3}$	$75.5_{0.3}$	$72.6_{0.3}$	$80.4_{0.3}$	$72.9_{0.4}$	64.5
SimSiam	RedCaps	-	$63.9_{0.3}$	$49.9_{0.3}$	$57.2_{0.3}$	$49.5_{0.4}$	$84.5_{0.3}$	$39.3_{0.3}$	$37.9_{0.3}$	$75.7_{0.3}$	$67.8_{0.4}$	$79.7_{0.3}$	$81.5_{0.3}$	$69.6_{0.3}$	$80.6_{0.3}$	$79.4_{0.3}$	65.5
SimCLR	RedCaps	-	$66.9_{0.3}$	$45.7_{0.3}$	$51.0_{0.3}$	$51.5_{0.4}$	$87.1_{0.2}$	$44.0_{0.3}$	$38.4_{0.3}$	$77.6_{0.3}$	$70.1_{0.3}$	$80.0_{0.3}$	$86.9_{0.2}$	$69.6_{0.3}$	$83.5_{0.3}$	$81.3_{0.3}$	66.7
Visual NNCLR	RedCaps	-	$65.6_{0.3}$	$54.1_{0.3}$	$61.7_{0.3}$	$45.8_{0.3}$	$85.3_{0.3}$	$37.9_{0.3}$	$34.9_{0.3}$	$75.2_{0.3}$	$67.3_{0.4}$	$81.1_{0.3}$	$75.4_{0.3}$	$74.3_{0.3}$	$83.6_{0.3}$	$76.7_{0.3}$	65.6
CLIP	RedCaps	-	$88.9_{0.2}$	$64.6_{0.3}$	$73.1_{0.3}$	$78.3_{0.3}$	$90.9_{0.2}$	$69.7_{0.3}$	$40.7_{0.3}$	$75.7_{0.3}$	$77.5_{0.3}$	$91.6_{0.2}$	$94.7_{0.2}$	$89.8_{0.2}$	$75.3_{0.3}$	$74.8_{0.3}$	77.5
CLIP (SBERT Encoder)	RedCaps	-	$89.9_{0.2}$	$59.9_{0.3}$	$67.9_{0.3}$	$83.2_{0.3}$	$91.1_{0.2}$	$70.2_{0.3}$	$41.0_{0.3}$	$75.0_{0.3}$	$79.4_{0.3}$	$91.2_{0.2}$	$94.5_{0.2}$	$89.4_{0.2}$	$72.3_{0.3}$	$74.9_{0.3}$	77.1
Language NNCLR	RedCaps	-	$89.3_{0.2}$	$65.3_{0.3}$	$73.4_{0.3}$	$78.6_{0.3}$	$90.8_{0.2}$	$68.4_{0.3}$	$40.4_{0.3}$	$75.2_{0.3}$	$78.8_{0.3}$	$90.9_{0.2}$	$94.3_{0.2}$	$89.6_{0.2}$	$75.2_{0.3}$	$71.9_{0.3}$	77.3
SLIP	RedCaps	-	$81.5_{0.3}$	$63.5_{0.3}$	$70.8_{0.3}$	$63.1_{0.4}$	$91.3_{0.2}$	$62.9_{0.3}$	$42.1_{0.4}$	$79.6_{0.3}$	$76.4_{0.3}$	$88.4_{0.2}$	$92.2_{0.2}$	$83.4_{0.2}$	$82.7_{0.3}$	$80.8_{0.3}$	75.6
Sampling Space - Language																	
LGSimCLR	RedCaps	SBERT (MiniLM)	$90.4_{0.2}$	$67.1_{0.3}$	$76.7_{0.3}$	$83.9_{0.3}$	$92.7_{0.2}$	$79.2_{0.3}$	$52.1_{0.4}$	$81.2_{0.3}$	$86.2_{0.3}$	$95.5_{0.1}$	$97.6_{0.1}$	$87.4_{0.2}$	$86.9_{0.2}$	$89.0_{0.2}$	83.3
LGSimCLR	RedCaps	CLIP (400M)	$90.7_{0.2}$	$65.8_{0.3}$	$75.6_{0.3}$	$83.8_{0.3}$	$92.8_{0.2}$	$80.9_{0.3}$	$52.0_{0.4}$	$81.4_{0.3}$	$85.6_{0.3}$	$95.5_{0.1}$	$97.5_{0.1}$	$87.3_{0.2}$	$84.8_{0.2}$	$89.3_{0.2}$	83.1
LGSimCLR	RedCaps	CLIP (RedCaps)	$90.4_{0.2}$	$64.8_{0.3}$	$75.3_{0.3}$	$82.2_{0.3}$	$92.8_{0.2}$	$76.6_{0.3}$	$50.4_{0.4}$	$81.3_{0.3}$	$84.6_{0.3}$	$95.2_{0.2}$	$97.7_{0.1}$	$86.9_{0.2}$	$86.5_{0.2}$	$89.1_{0.2}$	82.4
LGSimCLR	RedCaps	FastText BoW	$88.4_{0.2}$	$62.1_{0.3}$	$73.7_{0.3}$	$79.3_{0.3}$	$92.2_{0.2}$	$74.0_{0.3}$	$52.5_{0.4}$	$79.4_{0.3}$	$82.7_{0.3}$	$94.3_{0.2}$	$97.5_{0.1}$	$83.0_{0.2}$	$85.4_{0.2}$	$88.5_{0.2}$	80.9
Sampling Space - Visual																	
LGSimCLR	RedCaps	ImageNet Supervised	$79.6_{0.3}$	$75.6_{0.3}$	$83.0_{0.3}$	$76.6_{0.3}$	$92.5_{0.2}$	$64.6_{0.4}$	$46.1_{0.4}$	$80.7_{0.3}$	$94.3_{0.2}$	$96.3_{0.1}$	$94.8_{0.2}$	$87.4_{0.2}$	$86.4_{0.2}$	$87.3_{0.2}$	81.8
LGSimCLR	RedCaps	SimCLR	$72.0_{0.3}$	$62.9_{0.3}$	$71.9_{0.3}$	$58.9_{0.4}$	$90.8_{0.2}$	$51.3_{0.3}$	$38.7_{0.3}$	$81.8_{0.3}$	$86.4_{0.3}$	$91.3_{0.2}$	$90.7_{0.2}$	$83.8_{0.2}$	$85.5_{0.2}$	$78.6_{0.3}$	74.6
LGSimCLR	RedCaps	CLIP (400M)	$88.8_{0.2}$	$72.5_{0.3}$	$79.7_{0.3}$	$77.6_{0.3}$	$93.1_{0.2}$	$73.3_{0.3}$	$45.6_{0.4}$	$82.2_{0.3}$	$90.9_{0.2}$	$94.6_{0.2}$	$96.3_{0.1}$	$89.2_{0.2}$	$84.6_{0.2}$	$87.4_{0.2}$	82.6
Sampling Scope																	
LGSimCLR	RedCaps	SBERT - Year	$89.7_{0.2}$	$61.2_{0.3}$	$72.2_{0.3}$	$81.2_{0.3}$	$92.0_{0.2}$	$71.9_{0.3}$	$48.4_{0.4}$	$80.1_{0.3}$	$79.3_{0.3}$	$93.8_{0.2}$	$97.4_{0.1}$	$84.5_{0.2}$	$84.0_{0.2}$	$87.8_{0.2}$	80.2
LGSimCLR	RedCaps	SBERT - Sub-Year	$88.0_{0.2}$	$59.0_{0.3}$	$66.6_{0.3}$	$76.5_{0.3}$	$90.6_{0.2}$	$63.6_{0.4}$	$45.2_{0.3}$	$78.5_{0.3}$	$80.7_{0.3}$	$94.8_{0.2}$	$97.4_{0.1}$	$83.2_{0.2}$	$82.2_{0.3}$	$88.7_{0.2}$	78.2
LGSimCLR	RedCaps	SBERT - Sub	$88.7_{0.2}$	$70.4_{0.3}$	$78.4_{0.3}$	$75.3_{0.3}$	$90.7_{0.2}$	$67.3_{0.3}$	$47.6_{0.4}$	$78.3_{0.3}$	$76.6_{0.3}$	$95.4_{0.1}$	$97.8_{0.1}$	$86.5_{0.2}$	$85.3_{0.2}$	$89.1_{0.2}$	80.5
Pre-training Datasets																	
LGSimCLR	CC3M	SBERT (MPNet)	$69.2_{0.3}$	$60.6_{0.3}$	$71.7_{0.3}$	$72.0_{0.3}$	$92.8_{0.2}$	$58.8_{0.3}$	$48.9_{0.4}$	$77.4_{0.3}$	$77.8_{0.3}$	$92.9_{0.2}$	$95.0_{0.2}$	$83.0_{0.3}$	$82.4_{0.3}$	$86.3_{0.3}$	76.3
LGSimCLR	CC12M	SBERT (MPNet)	$79.6_{0.3}$	$72.0_{0.3}$	$78.5_{0.3}$	$71.2_{0.3}$	$95.2_{0.1}$	$78.2_{0.3}$	$55.5_{0.4}$	$81.8_{0.3}$	$82.1_{0.3}$	$96.2_{0.1}$	$95.3_{0.1}$	$90.0_{0.2}$	$83.6_{0.3}$	$88.0_{0.2}$	81.9
LGSimCLR	RedCaps 2020	SBERT (MPNet)	$86.0_{0.2}$	$60.3_{0.3}$	$70.5_{0.3}$	$79.9_{0.3}$	$90.1_{0.2}$	$69.8_{0.3}$	$48.6_{0.4}$	$77.0_{0.3}$	$81.0_{0.3}$	$92.3_{0.2}$	$96.8_{0.1}$	$78.8_{0.3}$	$84.7_{0.2}$	$87.0_{0.2}$	78.8
Batch Size Scaling																	
SimCLR (256)	RedCaps	-	$64.9_{0.3}$	$52.7_{0.3}$	$57.9_{0.3}$	$51.4_{0.4}$	$86.6_{0.2}$	$43.6_{0.3}$	$38.3_{0.3}$	$77.1_{0.3}$	$68.7_{0.3}$	$79.2_{0.3}$	$85.9_{0.3}$	$69.7_{0.3}$	$84.1_{0.3}$	$81.3_{0.3}$	67.2
SimCLR (1024)	RedCaps	-	$67.5_{0.3}$	$54.0_{0.3}$	$59.2_{0.3}$	$53.0_{0.4}$	$87.2_{0.2}$	$44.7_{0.3}$	$38.9_{0.3}$	$77.9_{0.3}$	$71.5_{0.3}$	$80.4_{0.3}$	$87.9_{0.2}$	$71.8_{0.3}$	$82.5_{0.3}$	$81.8_{0.3}$	68.4
SimCLR (2048)	RedCaps	-	$68.4_{0.3}$	$51.8_{0.3}$	$57.8_{0.3}$	$53.3_{0.4}$	$87.3_{0.2}$	$45.4_{0.3}$	$38.8_{0.3}$	$77.8_{0.3}$	$73.2_{0.3}$	$81.6_{0.3}$	$87.9_{0.2}$	$71.6_{0.3}$	$84.0_{0.3}$	$81.6_{0.3}$	68.6
LGSimCLR (256)	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$66.6_{0.3}$	$75.8_{0.3}$	$81.6_{0.3}$	$92.6_{0.2}$	$75.3_{0.3}$	$50.5_{0.4}$	$81.6_{0.3}$	$83.2_{0.3}$	$95.3_{0.1}$	$97.6_{0.1}$	$86.8_{0.2}$	$86.4_{0.2}$	$88.9_{0.2}$	82.3
LGSimCLR (1024)	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$64.9_{0.3}$	$75.7_{0.3}$	$83.8_{0.3}$	$92.6_{0.2}$	$78.2_{0.3}$	$52.6_{0.4}$	$80.9_{0.3}$	$83.6_{0.3}$	$95.6_{0.1}$	$97.6_{0.1}$	$86.7_{0.2}$	$86.0_{0.2}$	$88.6_{0.2}$	82.6
LGSimCLR (2048)	RedCaps	SBERT (MPNet)	$90.6_{0.2}$	$67.5_{0.3}$	$76.6_{0.3}$	$83.9_{0.3}$	$92.6_{0.2}$	$79.7_{0.3}$	$51.5_{0.4}$	$80.6_{0.3}$	$83.8_{0.3}$	$95.8_{0.1}$	$97.6_{0.1}$	$87.1_{0.2}$	$86.6_{0.2}$	$89.2_{0.2}$	83.1
Alternative Formulations																	
LGSimCLR	RedCaps	SBERT (MPNet)	$90.3_{0.2}$	$66.3_{0.3}$	$75.5_{0.3}$	$83.1_{0.3}$	$92.7_{0.2}$	$77.6_{0.3}$	$50.6_{0.4}$	$81.1_{0.3}$	$84.1_{0.3}$	$95.4_{0.1}$	$97.6_{0.1}$	$86.5_{0.2}$	$85.0_{0.2}$	$89.0_{0.2}$	82.5
LGSimSiam	RedCaps	SBERT (MPNet)	$81.2_{0.3}$	$61.6_{0.3}$	$71.2_{0.3}$	$63.1_{0.4}$	$90.2_{0.2}$	$60.9_{0.3}$	$44.6_{0.4}$	$78.8_{0.3}$	$68.0_{0.3}$	$92.8_{0.2}$	$93.7_{0.2}$	$81.2_{0.3}$	$85.1_{0.2}$	$86.7_{0.2}$	75.7
LGSLIP	RedCaps	SBERT (MPNet)	$91.3_{0.2}$	$67.2_{0.3}$	$77.2_{0.3}$	$81.8_{0.3}$	$92.6_{0.2}$	$77.3_{0.3}$	$50.4_{0.4}$	$81.8_{0.3}$	$81.8_{0.3}$	$96.1_{0.1}$	$97.8_{0.1}$	$89.2_{0.2}$	$85.3_{0.2}$	$89.1_{0.2}$	82.8

Table 6. **ImageNet Evaluations.** We evaluate all models on several ImageNet robustness benchmarks. All models were trained using the ImageNet train set. We report the linear probe and few-shot classification performance. Subscripts show the 95% confidence interval across 5000 episodes.

			Linear Probe					Fewshot Classification				
Model	**Dataset**	**Sampling Space**	**ImageNet**	**ImageNet A**	**ImageNet R**	**ImageNet V2**	**ImageNet Sketch**	**ImageNet**	**ImageNet A**	**ImageNet R**	**ImageNet V2**	**ImageNet Sketch**
Pre-trained Checkpoints												
Supervised [40]	ImageNet	-	80.7	5.4	27.6	68.9	28.8	$97.4_{0.1}$	$51.6_{0.2}$	$61.4_{0.2}$	$94.5_{0.1}$	$64.8_{0.3}$
SimSiam [5]	ImageNet	-	30.8	1.0	10.6	25.0	9.9	$90.4_{0.2}$	$39.8_{0.2}$	$52.9_{0.2}$	$85.6_{0.2}$	$56.3_{0.3}$
MoCo v3 [6]	ImageNet	-	69.5	1.1	20.3	57.2	20.5	$91.1_{0.1}$	$36.0_{0.2}$	$52.4_{0.2}$	$85.8_{0.2}$	$55.2_{0.3}$
SwAV [2]	ImageNet	-	70.6	1.2	16.6	57.5	17.6	$91.2_{0.1}$	$41.7_{0.2}$	$46.6_{0.2}$	$86.2_{0.2}$	$49.3_{0.3}$
SimCLR [4]	ImageNet	-	68.7	0.9	16.0	56.1	15.7	$90.2_{0.2}$	$36.2_{0.2}$	$43.7_{0.2}$	$84.8_{0.2}$	$44.9_{0.3}$
CLIP [28]	CLIP (400M)	-	73.2	8.2	31.9	61.5	31.8	$95.5_{0.1}$	$68.0_{0.2}$	$69.0_{0.2}$	$93.2_{0.1}$	$74.5_{0.3}$
RedCaps-trained Baselines												
SwAV	RedCaps	-	52.1	0.8	7.0	39.0	6.6	$80.4_{0.2}$	$38.6_{0.2}$	$35.1_{0.2}$	$75.5_{0.2}$	$33.1_{0.2}$
SimSiam	RedCaps	-	52.9	0.8	8.0	40.3	8.7	$78.8_{0.2}$	$39.1_{0.2}$	$38.9_{0.2}$	$73.4_{0.2}$	$39.3_{0.2}$
SimCLR	RedCaps	-	56.2	0.8	8.4	42.4	8.9	$79.8_{0.2}$	$39.6_{0.2}$	$38.7_{0.2}$	$74.4_{0.2}$	$38.6_{0.2}$
Visual NNCLR	RedCaps	-	54.4	0.8	8.3	41.0	8.3	$81.3_{0.2}$	$39.5_{0.2}$	$38.0_{0.2}$	$76.3_{0.2}$	$36.8_{0.2}$
CLIP	RedCaps	-	62.6	2.1	14.5	49.8	13.7	$88.7_{0.2}$	$44.7_{0.2}$	$46.6_{0.2}$	$84.7_{0.2}$	$46.1_{0.3}$
CLIP (SBERT Encoder)	RedCaps	-	61.5	2.1	14.4	49.2	13.2	$89.1_{0.2}$	$42.6_{0.2}$	$46.4_{0.2}$	$84.7_{0.2}$	$49.2_{0.3}$
Language NNCLR	RedCaps	-	61.6	2.1	13.7	49.6	13.3	$89.0_{0.2}$	$45.1_{0.2}$	$47.4_{0.2}$	$84.9_{0.2}$	$48.8_{0.3}$
SLIP	RedCaps	-	62.6	0.9	12.6	49.2	12.5	$86.7_{0.2}$	$43.0_{0.2}$	$43.5_{0.2}$	$82.0_{0.2}$	$43.8_{0.3}$
Sampling Space - Language												
LGSimCLR	RedCaps	SBERT (MiniLM)	65.3	1.2	16.8	52.8	16.8	$91.0_{0.1}$	$45.1_{0.2}$	$58.4_{0.2}$	$86.6_{0.2}$	$60.8_{0.3}$
LGSimCLR	RedCaps	CLIP (400M)	65.7	1.2	16.7	53.0	16.8	$90.9_{0.2}$	$45.4_{0.2}$	$57.1_{0.2}$	$86.6_{0.2}$	$59.2_{0.3}$
LGSimCLR	RedCaps	CLIP (RedCaps)	65.4	1.3	16.9	53.0	16.5	$90.6_{0.2}$	$45.4_{0.2}$	$57.1_{0.2}$	$86.2_{0.2}$	$58.5_{0.3}$
LGSimCLR	RedCaps	FastText BoW	62.6	0.6	15.5	49.7	14.9	$89.7_{0.2}$	$44.0_{0.2}$	$55.8_{0.2}$	$84.9_{0.2}$	$57.0_{0.3}$
Sampling Space - Visual												
LGSimCLR	RedCaps	ImageNet Supervised	66.9	0.8	19.0	53.6	16.7	$91.2_{0.1}$	$41.4_{0.2}$	$56.0_{0.2}$	$86.1_{0.2}$	$53.4_{0.3}$
LGSimCLR	RedCaps	SimCLR	63.0	0.8	12.8	49.1	12.2	$88.4_{0.2}$	$38.7_{0.2}$	$45.8_{0.2}$	$83.1_{0.2}$	$46.0_{0.3}$
LGSimCLR	RedCaps	CLIP (400M)	68.4	1.3	18.1	55.2	16.7	$90.9_{0.2}$	$45.2_{0.2}$	$55.1_{0.2}$	$86.3_{0.2}$	$52.7_{0.3}$
Sampling Scope												
LGSimCLR	RedCaps	SBERT - Year	64.5	0.9	15.6	51.6	15.9	$88.9_{0.2}$	$43.2_{0.2}$	$53.2_{0.2}$	$84.2_{0.2}$	$51.8_{0.3}$
LGSimCLR	RedCaps	SBERT - Sub-Year	66.2	1.3	19.0	53.2	20.1	$85.8_{0.2}$	$41.2_{0.2}$	$53.3_{0.3}$	$80.6_{0.2}$	$53.1_{0.3}$
LGSimCLR	RedCaps	SBERT - Sub	67.0	1.5	19.5	54.4	20.8	$80.1_{0.2}$	$38.8_{0.2}$	$48.2_{0.3}$	$75.3_{0.2}$	$42.9_{0.3}$
Pre-training Datasets												
LGSimCLR	CC3M	SBERT (MPNet)	17.5	1.0	8.9	13.4	5.4	$87.9_{0.2}$	$41.2_{0.2}$	$56.6_{0.2}$	$83.5_{0.2}$	$60.8_{0.3}$
LGSimCLR	CC12M	SBERT (MPNet)	65.0	0.7	22.8	52.1	27.1	$91.9_{0.1}$	$45.7_{0.2}$	$66.0_{0.2}$	$87.8_{0.2}$	$73.0_{0.3}$
LGSimCLR	RedCaps 2020	SBERT (MPNet)	58.7	0.6	13.1	45.4	11.4	$87.5_{0.2}$	$41.1_{0.2}$	$52.1_{0.2}$	$82.3_{0.2}$	$51.3_{0.3}$
Batch Size Scaling												
SimCLR (256)	RedCaps	-	54.8	0.7	7.9	41.6	8.1	$79.5_{0.2}$	$39.5_{0.2}$	$38.4_{0.2}$	$74.0_{0.2}$	$38.3_{0.2}$
SimCLR (1024)	RedCaps	-	57.2	0.7	8.8	43.7	8.7	$80.5_{0.2}$	$39.4_{0.2}$	$39.2_{0.2}$	$75.1_{0.2}$	$38.3_{0.2}$
SimCLR (2048)	RedCaps	-	58.1	0.8	9.0	44.5	9.2	$81.1_{0.2}$	$39.3_{0.2}$	$39.5_{0.2}$	$75.8_{0.2}$	$38.9_{0.2}$
LGSimCLR (256)	RedCaps	SBERT (MPNet)	64.9	1.2	16.7	52.0	16.6	$90.6_{0.2}$	$46.3_{0.2}$	$57.4_{0.2}$	$86.3_{0.2}$	$59.7_{0.3}$
LGSimCLR (1024)	RedCaps	SBERT (MPNet)	65.8	1.2	16.7	52.9	16.8	$90.7_{0.2}$	$44.8_{0.2}$	$57.2_{0.2}$	$86.3_{0.2}$	$58.8_{0.3}$
LGSimCLR (2048)	RedCaps	SBERT (MPNet)	66.2	1.1	17.3	53.1	17.4	$90.8_{0.2}$	$44.7_{0.2}$	$58.0_{0.2}$	$86.3_{0.2}$	$60.1_{0.3}$
Alternative Formulations												
LGSimCLR	RedCaps	SBERT (MPNet)	65.2	1.1	16.6	52.5	16.2	$90.9_{0.2}$	$45.0_{0.2}$	$57.4_{0.2}$	$86.4_{0.2}$	$59.2_{0.3}$
LGSimSiam	RedCaps	SBERT (MPNet)	58.9	0.7	12.2	45.9	12.2	$88.1_{0.2}$	$44.1_{0.2}$	$48.8_{0.2}$	$83.5_{0.2}$	$50.7_{0.3}$
LGSLIP	RedCaps	SBERT (MPNet)	66.8	1.4	18.2	54.3	18.9	$90.4_{0.2}$	$46.2_{0.2}$	$58.4_{0.2}$	$86.2_{0.2}$	$59.7_{0.3}$

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 2

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 2, 8, 9, 10

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. 2, 3

[4] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 2, 3, 7, 8, 9, 10

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8, 9, 10

[6] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 2, 3, 8, 9, 10

[7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, Oct 2017. 2

[8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[9] Adam Coates, Andrew Ng, and Honglak Lee. An Analysis of Single Layer Networks in Unsupervised Feature Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 5

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 2, 7

[12] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9588–9597, October 2021. 3

[13] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004. 1, 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[15] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 2

[16] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2021. 2, 5

[17] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 5

[18] 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. 1

[19] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 2

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. 2

[21] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision Exists Everywhere: A Data Efficient Contrastive Language-Image Pre-training Paradigm. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 3

[22] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 1989. 1

[23] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 2

[24] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2022. 3

[25] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 2

[26] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1

[28] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. 1, 2, 3, 8, 9, 10

[29] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 2, 5

[30] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. 2, 7

[31] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[32] Igor Susmelj, Matthias Heller, Philipp Wirth, Prescott Jeremey, and Malte Ebner. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020. 3

[33] TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets. 1

[34] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 1

[35] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, 2018. 2

[36] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2, 5

[37] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[38] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 1

[39] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 2

[40] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS Workshop on ImageNet: Past, Present, and Future*, 2021. 2, 8, 9, 10

[41] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 2