Task-customized Self-supervised Pre-training with Scalable Dynamic Routing

Zhili Liu[1,2], Jianhua Han[2], Kai Chen[1], Lanqing Hong[2], Hang Xu[2], Chunjing Xu[2], Zhenguo Li[2]

[1] Department of Computer Science and Engineering,
Hong Kong University of Science and Technology
[2] Huawei Noah's Ark Lab
{zhili.liu, kai.chen}@connect.ust.hk, {hanjianhua4, honglanqing, xu.hang, xuchunjing, li.zhenguo}@huawei.com

Abstract

Self-supervised learning (SSL), especially contrastive methods, has raised attraction recently as it learns effective transferable representations without semantic annotations. A common practice for self-supervised pre-training is to use as much data as possible. For a specific downstream task, however, involving irrelevant data in pre-training may degenerate the downstream performance, observed from our extensive experiments. On the other hand, for existing SSL methods, it is burdensome and infeasible to use different downstream-task-customized datasets in pre-training for different tasks. To address this issue, we propose a novel SSL paradigm called Scalable Dynamic Routing (SDR), which can be trained once and deployed efficiently to different downstream tasks with task-customized pre-trained models. Specifically, we construct the SDRnet with various sub-nets and train each sub-net with only one subset of the data by data-aware progressive training. When a downstream task arrives, we route among all the pre-trained sub-nets to get the best along with its corresponding weights. Experiment results show that our SDR can train 256 sub-nets on ImageNet simultaneously, which provides better transfer performance than a unified model trained on the full ImageNet, achieving state-of-the-art (SOTA) averaged accuracy over 11 downstream classification tasks and AP on PASCAL VOC detection task.

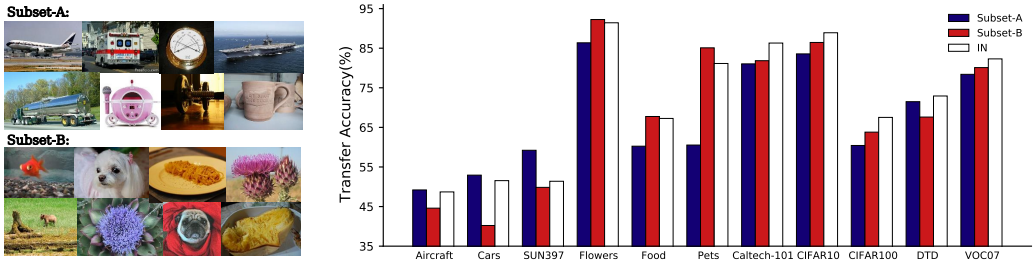
1 Introduction

Self-supervised learning (SSL) has attracted lots of attention in recent years (Caron et al. 2020; He et al. 2020; Grill et al. 2020), which proposes to learn representations via pretext tasks without semantic annotations. Recent works in SSL (Xu et al. 2020; Chen et al. 2021) show competitive or even better transfer performance on various downstream tasks compared with supervised learning. Without the dependence of annotation, SSL makes it possible to use a large amount of unlabeled data (e.g., YFCC100M (Tian, Henaff, and Oord 2021) and Instagram (Goyal et al. 2021)) in model pre-training. However, will more data in self-supervised pre-training always lead to better transfer performance? In other words, *for a specific downstream task, will irrelevant data in pre-training hurt the downstream performance instead?*

To answer the above questions, we first conduct a preliminary experiment in Sec. 3 to evaluate the transfer performance of SSL models pre-trained on datasets with different semantics. We deliberately split the ImageNet into two disjoint subsets, namely Subset-A and Subset-B, based on their semantic dissimilarity in WordNet Tree (Miller 1998). We pre-train models with Subset-A, Subset-B and the full ImageNet separately using SimSiam (Chen and He 2021) without data annotations and evaluate the transfer performance on 11 downstream classification datasets. The training epochs for the three models are the same. As shown in Fig. 1(b), the model pre-trained on Subset-A shows the best transfer performance on Aircraft, Cars and SUN397, while the model pre-trained on Subset-B performs the best on Flowers, Pets, and Food. Only five out of eleven downstream tasks benefit more from the full ImageNet. The results indicate that involving irrelevant data in pre-training might instead hurt the downstream performance. This phenomenon is identified as the *negative transfer* in self-supervised pre-training. Similar observations have also been discussed in (Cole et al. 2021) and (Tian, Henaff, and Oord 2021). (Cole et al. 2021) further investigate the importance of using semantic-similar data in model pre-training for better transfer performance.

Prevailing SSL methods, such as MoCo-v2 (Chen et al. 2020c) and SimSiam (Chen and He 2021), usually neglect the influence of negative transfer and provide a common pre-trained model for different downstream tasks. A naive extension to eliminate the effects of negative transfer is to pre-train models with task-customized datasets. However, such an extension is actually impractical considering the burdensome computational cost of pre-training. (Tian, Henaff, and Oord 2021) simply splits a large-scale dataset (i.e., YFCC100M) into different subsets for customized model pre-training, which is not scalable for a large number of downstream tasks. It is desirable to develop an efficient SSL paradigm that provides task-customized pre-training models.

In this work, we propose a novel SSL paradigm called Scalable Dynamic Routing (SDR), which achieves dynamic pre-training and efficient deployment for different downstream tasks. Specifically, we construct the SDRnet with various sub-nets and train each sub-net with different subsets of the data, which contain different semantic clusters. We further propose a data-aware progressive training framework to stabilize the pre-training procedure of sub-nets and avoid collapse. When a downstream task arrives, we route among all sub-nets to obtain the best pre-trained model along with its weights. By



(a) Samples from Subset-A/B (b) Performance of models pre-trained on different datasets

Figure 1: Transfer performance for models pre-trained on ImageNet Subset-A, ImagNet Subset-B and the full ImageNet on different downstream datasets. (a) Subset-A is occupied by inanimate objects mostly, while Subset-B mainly contains organisms; (b) The model pre-trained on the full ImageNet only has the best performance on five out of the eleven tasks.

using SDR, we are able to pre-train a series of sub-nets simultaneously for the efficient deployment of various downstream tasks. To summarize, our main contributions are:

- With extensive experiments, we identify the negative transfer phenomenon in SSL that pre-training with irrelevant data might degenerate the transfer performance in specific downstream tasks.
- We propose Scalable Dynamic Routing (SDR), a novel SSL paradigm that can alleviate the effects of negative transfer by providing efficient and scalable task-customized self-supervised pre-training models.
- We successfully train 256 sub-nets simultaneously on ImageNet and achieve the state-of-the-art averaged accuracy among 11 downstream classification datasets and AP on PASCAL VOC detection task.

2 Related work

Self-supervised learning, especially contrastive learning, learns representations without data annotation by "learning to compare" through a Noise Contrastive Estimation (NCE) (He et al. 2020) objective. Primitive works in SSL mainly adopt the context-instance contrastive learning framework, which focuses on modeling the belonging relationship between the local feature of a sample and its global context representation (Kim et al. 2018; Bachman, Hjelm, and Buchwalter 2019). Recently, instance-instance contrastive learning becomes prevailing, which directly studies the relationships between instance-level local representations of different samples. The prototype of leveraging instance-level discrimination as a pretext task is InsDis (Wu et al. 2018). MoCo (He et al. 2020) further proposes the idea of leveraging instance discrimination via momentum contrast of positive/negative sample pairs, while SimCLR (Chen et al. 2020a) introduces various forms of data augmentations to produce a pretext-invariant representation. BYOL (Grill et al. 2020) radically discards negative sampling in SSL but still achieves satisfactory transfer performance. SimSiam (Chen and He 2021) further claims meaningful representations can be learned without (i) negative sample pairs, (ii) large batches, and (iii) momentum encoders. Besides, clustering-based methods, including PCL-v1, PCL-v2 (Li et al. 2021), and SwAV (Caron et al. 2020), leverage clustering to yield pseudo labels for learning representations. However, existing SSL methods usually offer a unified pre-trained model which may not be applicable for various downstream tasks when negative transfer occurs, as shown in Sec. 3. It is impractical to pre-train different SSL models for different tasks due to the burdensome computational cost. It is desirable to develop an efficient and scalable task-customized SSL paradigm.

Dynamic neural network is an emerging topic (Han et al. 2021). Unlike static networks with fixed computational graphs and weights during inference, dynamic networks can adapt their structures or parameters to different scenarios, leading to advantages in terms of efficiency, adaptiveness, and performance. Based on the dynamic nature, they can be categorized into instance-wise (Li et al. 2017; Figurnov et al. 2017), spatial-wise (Cao et al. 2019; Wang et al. 2019) and temporal-wise networks (Campos et al. 2017; Hansen et al. 2019; Tao et al. 2019). In order to allow the adaptiveness of our pre-trained backbone to different tasks and datasets, we need to explore task-wise/dataset-wise dynamic networks. Compared with instance-wise dynamic networks (Odena, Lawson, and Olah 2017; Liu and Deng 2018), our method focuses on selecting the best candidate model for a downstream task/dataset and fixes the network structure during inference.

Multi-task Learning (MTL) aims at learning a model that can perform well on several downstream tasks, which are usually pre-defined during training, while SDR can not foresee any downstream tasks when pre-training. (McDermott et al. 2021), (Liu et al. 2019) and (Hu et al. 2019) show that the model using a shared backbone for all tasks and multi-heads for different specific tasks, namely hard-parameter sharing, is useful on time-series data, language and graph data separately. (Gao et al. 2021) shows that network design can better benefit the task relationship, while (Gao et al. 2021) trains a mask along with the model parameters, so each task has its own mask. SDR is designed differently by super-sub-net structure, neither requiring multi-heads nor masks, making SDR more parameter-efficient. Furthermore, SDR is also scalable for training 256 sub-tasks simultaneously, which is significantly larger than most MTL methods.

3 Preliminary on Negative Transfer

In this section, we conduct a preliminary experiment to evaluate the transfer performance of models pre-trained on datasets with different semantic annotations. Following (Huh, Agrawal, and Efros 2016), we split the ImageNet dataset

into two disjoint subsets, namely Subset-A and Subset-B, based on their semantic dissimilarity in WordNet Tree (Miller 1998), which can be achieved by searching the WordNet hierarchy to avoid two splits having the same ancestor at depth four. In this case, classes in Subset-A are sufficiently disjoint from Subset-B. Specifically, images in Subset-A are primarily inanimate objects, such as cars and airplanes, while Subset-B mainly contains organisms, such as plants and animals. See Fig. 1(a) as an illustration.

Then, we pre-train with Subset-A, Subset-B and the full ImageNet separately using SimSiam (Chen and He 2021) without data annotations, and evaluate the transfer performance on 11 downstream classification datasets via the many-shot classification protocol following (Ericsson, Gouk, and Hospedales 2021). See more experimental details and hyper-parameters in Appendix A.

The results are summarized in Fig. 1(b). As can be seen, the model pre-trained on Subset-A shows the best performance on Aircraft, Cars and SUN397. Specifically, for SUN397, the model with Subset-A results in a 7.83% improvement on classification accuracy compared with the model pre-trained on the full ImageNet. On the other hand, the model pre-trained on Subset-B performs the best on Flowers, Pets, and Food. These results are consistent with the observations that Subset-A is mostly inanimate objects, while Subset-B mainly contains organisms. Only five out of the eleven downstream tasks benefit from the full ImageNet, suggesting that more data in pre-training is not always better. Involving semantic-irrelevant data in pre-training might hurt the downstream-task performance. The observation of *negative transfer* in self-supervised pre-training motivates us to develop an efficient but scalable task-customized SSL paradigm.

4 Method

In this section, we start by a brief introduction of the SimSiam (Chen et al. 2020a), our simple yet effective SSL baseline, in Sec. 4.1. Then we introduce the proposed Scalable Dynamic Routing (SDR) paradigm for the simultaneous pre-training of a series of sub-nets in Sec. 4.2. Finally, we discuss the efficient deployment of these sub-nets to different downstream tasks in Sec. 4.3.

4.1 Overview of SimSiam

SimSiam (Chen et al. 2020a) takes two randomly augmented views x_1 and x_2 from an image x as inputs. The two views are processed by an encoder f_θ, which contains a backbone and a projection MLP head. The encoder f_θ shares weights between x_1 and x_2. Furthermore, a prediction MLP head h_θ transforms the output of one view and matches it with the other. SimSiam learns representations by comparing similarity of the encoder output $f_\theta(\cdot)$ and the prediction head output $h_\theta(\cdot)$. Finally, a consistency loss is calculated as:

$$\mathcal{L}_{SSL}(D;\theta) = \mathop{\mathbb{E}}_{\substack{x_1,x_2\sim\tau(x)\\ x\sim D}} \frac{h_\theta(f_\theta(x_1))}{\|h_\theta(f_\theta(x_1))\|_2}\frac{f_\theta(x_2)}{\|f_\theta(x_2)\|_2}, \quad (1)$$

where $\|\cdot\|_2$ denotes the l_2-norm, and D, $\tau(\cdot)$ indicate the unlabeled training dataset and distribution of data augmentation respectively. Moreover, the stop-gradient operation is adopted to avoid collapse solutions in the implementation.

4.2 Scalable Dynamic Routing

As shown in Fig. 2, our Scalable Dynamic Routing (SDR) paradigm consists of three steps. First, we cluster the dataset into disjoint subsets, then we construct the SDRnet model containing many sub-nets and dynamically train each sub-net with its corresponding subsets through data-aware progressive training. Refer to Algorithm 1 for the entire training procedure. After pre-training, we can route among all sub-nets to find the one that transfers best to a specific downstream task. Following are the details of the three steps.

Data clustering. The basic idea of SDR is to apply data with different semantics to train different networks simultaneously and efficiently. A clustering procedure is adopted to group the unlabeled training data into different semantic clusters. We first pre-train a SimSiam model using the entire dataset and collect all images features, denoted as $\boldsymbol{F} = [\boldsymbol{f_1}, \boldsymbol{f_2}, ..., \boldsymbol{f_n}]$. Large-scale clustering is performed on **fixed** $\boldsymbol{F}$ following (Caron et al. 2020). Specifically, we set k to be our desired number of clusters and define the learnable centroids of the clusters as $\boldsymbol{C} = [\boldsymbol{c_1}, \boldsymbol{c_2}, ..., \boldsymbol{c_k}]$. Then the assignment of features to the clusters can be computed as $\boldsymbol{S} = \boldsymbol{F}^T\boldsymbol{C}$. We define an auxiliary matrix $\boldsymbol{U} = [\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}]$, which can be regarded as the posterior distribution of clustering (Asano, Rupprecht, and Vedaldi 2019). Our goal is to maximize the similarity between $\boldsymbol{U}$ and $\boldsymbol{S}$, which can be denoted as follows,

$$\max_{\boldsymbol{U}}[Tr(\boldsymbol{U}^T\boldsymbol{S}) + \epsilon H(\boldsymbol{U})], \quad (2)$$

where $H(\boldsymbol{U})$ denotes the entropy of $\boldsymbol{U}$. We optimize $\boldsymbol{U}$ and $\boldsymbol{C}$ iteratively. $\boldsymbol{U}$ is solved by the iterative Sinkhorn-Knopp algorithm (Cuturi 2013), while $\boldsymbol{C}$ is learned through SGD to minimize the cross entropy between $\boldsymbol{U}$ and $\boldsymbol{S} = \boldsymbol{F}^T\boldsymbol{C}$. After several epochs of training, we adopt $\boldsymbol{S}$ to be our assignment matrix. The final clustering result is denoted as $D_i (i = 1, 2, ..., k)$, and D_0 represents the entire dataset.

Framework optimization. The whole SDRnet will be trained by the entire training set D_0, while the i-th sub-net will be additionally trained with its corresponding sub-dataset D_i. Let W_0 be the weights of the total network, and $W_i \subseteq W_0 (i = 1, \cdots, k)$ is the weights corresponding to the i-th sub-net. The training loss can be formalized as:

$$\min_{W_0}[\mathcal{L}_{SSL}(D_0; W_0) + \sum_i \mathcal{L}_{SSL}(D_i; W_i)], \quad (3)$$

and the overall objective optimizes the weights of the SDRnet and sub-nets simultaneously on their corresponding datasets.

Splits of sub-nets and SDR block. Here we introduce our design of SDR block that is modified from ResNet-block (He et al. 2016) and scalable to a large number of sub-nets. Without loss of generality, we denote every column in Fig. 3 as a block since our discussion of block behavior is the same as the layer behavior and all layers in the same block perform identically. We split the channels of each block into two parts: individual groups and shared groups. A path is defined as an **arbitrary** connection of individual groups between consecutive blocks. There are 3 blocks(columns) and

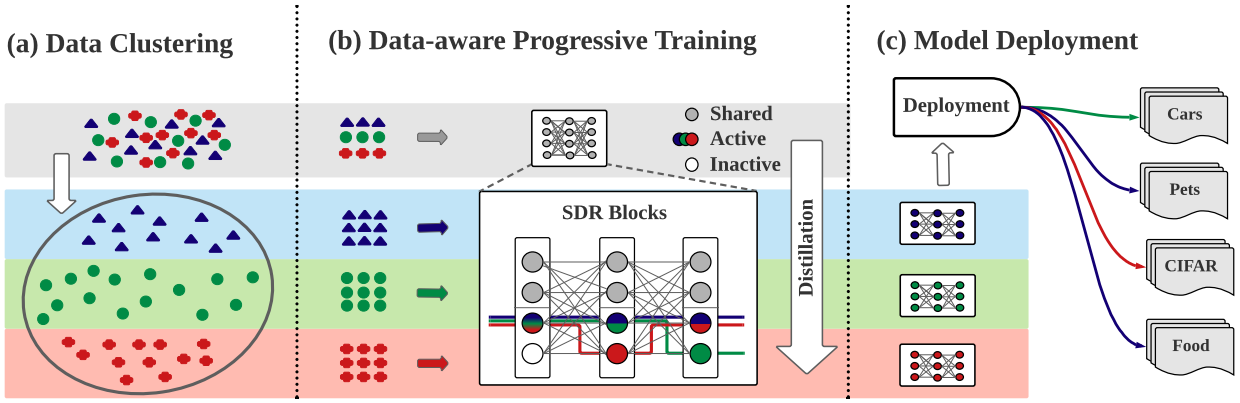
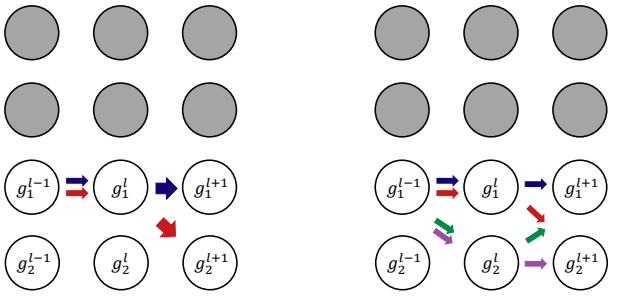


Figure 2: An overview of our proposed SDRnet. (a) We first separate unlabeled images into different subsets by clustering; (b) SDRnet is then constructed with various sub-nets and each sub-net is trained with only one subset of the data by data-aware progressive training; (c) When a downstream task arrives, we route among all the sub-nets to get the best pre-trained model.



(a) Data-aware progressive training at phase 1

(b) Data-aware progressive training at phase 2

Figure 3: Design of SDR block and data-aware progressive training. (a) Illustration of progressive training at phase 1. Each column represents the design of SDR block, which consists of a shared group (2 grey nodes) and several individual groups (2 white nodes). Path is defined as the connections of any individual groups in the consecutive blocks. In phase 1, we add sub-nets containing blue and red paths. (b) Illustration of progressive training at phase 2. In phase 2, we enlarge the space with sub-nets containing green and purple paths.

each block contains 2 share groups(grey nodes) and 2 individual groups(white nodes). $[g_1^{l-1}, g_1^l, g_1^{l+1}]$, $[g_1^{l-1}, g_1^l, g_2^{l+1}]$ are two example paths showed as the blue and red paths, where g_i^l denotes the i-th individual group of the l-th block. The total number of paths can be computed from the number of individual groups and the number of blocks, that is $2^3 = 8$ in the figure. This design makes the model size grow log-linearly with the number of paths, which is extremely space-saving than training a model for one sub-dataset. In general, each D_i will be mapped to a path in advance. When data in D_i comes, it will inference the block with the concatenation of the shared group and the individual group defined in the path, thus W_i is defined as the union of parameters in the corresponding path and all shared groups.

Data-aware progressive training. It is challenging to train a large number of sub-nets simultaneously due to the instability of the training process. Naively sampling a sub-net and training it with its corresponding dataset always leads to instability, which finally results in feature collapse in self-supervised learning. We therefore propose the data-aware progressive training by block to stabilize the optimization process. A network space is defined and enlarged progressively after each phase. At each phase, we only sample and train the networks inside the space. Specifically, the network space only contains the largest network at first. We start adding sub-nets whose paths only differ in the last block (i.e. the blue and red paths in Fig. 3(a)). In the next phase, we continue to add sub-nets with path green and purple, thus paths of all sub-nets in the space now differ in the last two blocks, and go on. With such progressive training, we are able to train the largest network and many sub-nets simultaneously.

Task-customized knowledge distillation. Besides progressive training by blocks, we further propose a task-customized distillation method called SiamKD to balance the model discrepancy and the possible performance drop resulted from training with fewer data and less time. Specifically, features provided by sub-nets are also applied to predict the features of the largest network. The loss function is represented as:

$$\mathcal{L}_{SiamKD}(D_i; W_i) = \mathop{\mathbb{E}}_{\substack{x_1, x_2 \sim \tau(x) \\ x \sim D_i}} \frac{h(f_{W_i}(x_2))}{\|h(f_{W_i}(x_2))\|_2} \frac{f_{W_0}(x_1)}{\|f_{W_0}(x_1)\|_2}. \tag{4}$$

Note that the stop gradient operation is performed on the SDRnet when calculating $\mathcal{L}_{SiamKD}$, as we distill the SDRnet to each sub-net unilaterally. Experiments show that SiamKD significantly outperforms the L2 distillation loss. See the ablation study in Sec. 5.3 for more details.

4.3 Deployment

When a downstream task comes, one can route among all the sub-nets to find the best pre-trained model for the task. As for classification task, one practical implementation is to adopt the k-nearest-neighbor (kNN) (Wu et al. 2018) classifier for fast performance evaluation. For detection task, early stopping can be applied to choose the best pre-trained model. Our experimental results in Sec. 5 verify the effectiveness and efficiency of the above model selection procedures.

Algorithm 1: Training Procedure of SDR.

Require: k: the number of sub-nets, D_0: the entire training dataset

Split D_0 into k subsets $D_1, D_2, ..., D_k$.
Initialize SDRnet with random parameters.
Initialize network space $N = \{W_0\}$.
repeat
  repeat
    Randomly sample a network W_i in N.
    Find the corresponding sub-dataset D_i by W_i.
    Randomly sample a batch of data in D_i.
    Calculate $\mathcal{L}_{SSL}(D_i; W_i) + \mathcal{L}_{SiamKD}(D_i; W_i)$.
    Optimize W_i.
  until Converge.
  Update N with more sub-nets according to data-aware progressive training.
until All sub-nets are trained to converge.

5 Experiment

In this section, we apply the proposed SDR to train SDRnet and a series of sub-nets. We demonstrate the effectiveness of SDR by evaluating the resulting pre-trained models on various downstream tasks including classification and detection. We also take ablation studies on the number of sub-nets, training time and the distillation method as shown in Sec. 5.3.

5.1 Implementation Details

Model configuration. We apply the SDR block in all four stages of ResNet. In each stage, all blocks have four individual groups and one shared group. The size of shared groups is half of all groups. All blocks in same stage perform identically. So we can generate $4^4 = 256$ different sub-nets. For comparison, we enlarge our model so that the size of each sub-net is close to that of ResNet-50 (He et al. 2016), the most commonly used backbone in SSL. For deployment, we reset each sub-net with the corresponding batch normalization (BN) statistics in pre-training following (Cai et al. 2019). We adopt ImageNet as the dataset for self-supervised pre-training without using labels. We use SimSiam (Chen and He 2021) and BYOL (Grill et al. 2020) as our baseline models. Considering the simplicity and effectiveness, we perform most ablations on SimSiam.

Downstream tasks. We validate our method on both classification and detection. For **classification tasks**, we adopt the benchmark proposed in (Ericsson, Gouk, and Hospedales 2021), which considers 11 datasets including both coarse-grained (*e.g.*, CIFAR100 and VOC2007) and fine-grained ones (*e.g.*, Standard Cars and FGVC Aircraft), as detailed in Appendix A. The quality of the pre-trained representations is evaluated by training a supervised linear classifier upon the **frozen** representations in the training set, and then testing it in the validation set. For **detection task**, we evaluate the pre-trained models on PASCAL VOC detection dataset with Faster-RCNN, following the transfer protocol of MoCo (Chen et al. 2020c). Specifically, the pre-trained model is fine-tuned on the VOC `trainval07+12` set and evaluated on the VOC `test2007` set. See Appendix A for more experimental details and hyper-parameters.

5.2 Results and Analysis

Classification. The transfer performance of pre-trained models on classification tasks are summarized in Table 1. As can be seen, SDR improves the performance on all the downstream datasets, compared with the model pre-trained on the full ImageNet, i.e., the SimSiam and BYOL baselines. SDR achieves 2.38% and 2.23% improvement of accuracy respectively over eleven downstream tasks, demonstrating the effectiveness of task-customized self-supervised pre-training to alleviate negative transfer. Especially, SDR(BYOL) reaches the best performance on 7 tasks and second best on 3 tasks, whose average accuracy also outperforms other state-of-the-art methods.

Note that the baseline SimSiam and BYOL uses ResNet-50 as the backbone, whose parameter count is 23.5 million, while the size of each SDR sub-net is 22.6 million. With a smaller model, we achieve better results. Besides, under the same time consumption, we are able to train 256 sub-nets, showing the scalability of our method. In terms of efficient deployment, it only takes several minutes to route among all sub-nets using the kNN classifier to find the best model. Compared with the total training time of SDR, which usually takes hundreds of hours, the searching time is negligible. On Food-101 (Bossard, Guillaumin, and Van Gool 2014) dataset, for example, it takes 20 minutes with 8*V100 to pick up the best route.

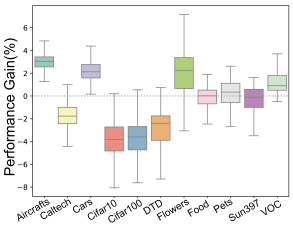
Analysis on downstream tasks. We notice that the performance of the sub-nets varies significantly for different downstream datasets. As showed in Fig. 4(a), we provide the performance gains on kNN accuracy of the 256 sub-nets compared with baseline trained on full ImageNet. The downstream tasks including Aircraft, Cars and Flowers have significant average performance improvement when using SDR. That might be because these datasets are fine-grained datasets sensitive to the negative transfer. Therefore, a subset of ImageNet tends to provide a better pre-trained model. On the other hand, downstream tasks like CIFAR10, CIFAR100 and DTD show limited improvement when using SDR. That might be because these datasets contain classes similar to those in ImageNet, so that effects of negative transfer are negligible. As a result, the full ImageNet may provide more applicable pre-trained models. These observations are also consistent with the preliminary experiments (see Sec. 3).

For illustration purpose, we plot the histogram of kNN accuracy on the Aircraft dataset over the 256 sub-nets. The results are summarized in Fig. 4(b). We further investigate the distribution of classes for the subset that results in the best kNN accuracy on Aircraft. As can be seen, the best pre-trained model for Aircraft is actually pre-trained on a subset of ImageNet mainly containing images of flying objects, such as kite, albatross and stork. The results indicate the effectiveness of the data clustering and data-aware progressive training process.

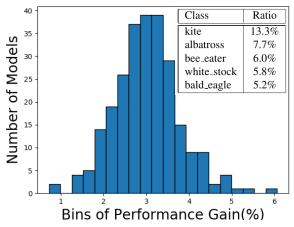
Detection. The transfer results for detection task are provided in Table 2. For detection task, SDR also improves the baselines by 1.48% and 0.78% in AP with smaller model size, compared with the models pre-trained on the full ImageNet,

Table 1: Transfer performance(%) of self-supervised pre-training models on various classification downstream tasks (Bold: best, underline: second best). Supervised baseline is also provided in the first row. SDR improves the baselines significantly by 2.38% and 2.28%. Especially, SDR(BYOL) performs best on seven tasks and second-best on three tasks, achieving state-of-the-art averaged accuracy. *: we take the officially released pre-trained weights and report the transfer performance. **: denotes our re-implementation under the same training epochs with SDR for a fair comparison.

	Epochs	Aircraft	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Flowers	Food	Pets	SUN397	VOC2007	Avg.
Supervised	90	43.59	90.18	44.92	91.42	73.90	72.23	89.93	69.49	91.45	60.49	83.60	73.75
InsDis (Wu et al. 2018)	200	36.87	71.12	28.98	80.28	59.97	68.46	83.44	63.39	68.78	49.47	74.37	62.29
MoCo-v1 (He et al. 2020)	200	35.55	75.33	27.99	80.16	57.71	68.83	82.10	62.10	69.84	51.02	75.93	62.41
PIRL (Misra and Maaten 2020)	200	37.08	74.48	28.72	82.53	61.26	68.99	83.60	64.65	71.36	53.89	76.61	63.92
PCL-v1 (Li et al. 2021)	200	21.61	76.90	12.93	81.84	55.74	62.87	64.73	48.02	75.34	45.70	78.31	56.73
PCL-v2 (Li et al. 2021)	200	37.03	86.42	30.51	91.91	73.54	70.59	85.34	64.88	82.76	56.25	81.14	69.12
MoCo-v2 (Chen et al. 2020c)	800	41.79	87.92	39.31	92.28	74.90	73.88	90.07	68.95	83.30	60.32	82.69	72.31
SimCLR-v1 (Chen et al. 2020a)	1000	44.90	90.05	43.73	91.18	72.73	74.20	90.87	67.47	83.33	59.21	80.77	72.59
SimCLR-v2 (Chen et al. 2020b)	800	46.38	89.63	50.37	92.53	76.78	76.38	92.90	73.08	84.72	61.47	81.57	75.07
InfoMin (Tian et al. 2020)	800	38.58	87.84	41.04	91.49	73.43	74.73	87.18	69.53	86.24	61.00	83.24	72.21
SeLa-v2 (Asano et.al. 2019)	400	37.29	87.20	36.86	92.73	74.81	74.15	90.22	71.08	83.22	62.71	82.73	72.09
DeepCluster-v2* (Caron et al. 2018)	400	48.75	90.52	50.94	94.15	79.33	76.70	93.98	75.90	86.78	**65.41**	84.30	76.98
SwAV* (Caron et al. 2020)	400	51.37	89.65	52.59	93.39	78.72	**78.09**	93.94	75.92	86.81	63.55	83.92	77.09
SimSiam** (Chen and He 2021)	200	51.30	87.02	53.80	89.12	68.43	72.99	91.83	67.35	83.64	52.97	83.40	72.90
SDR (SimSiam)	200	**55.84**	87.55	**61.06**	90.27	71.39	74.47	92.61	68.93	85.03	55.89	85.02	$75.28^{+2.38}$
BYOL** (Grill et al. 2020)	200	45.46	87.82	45.91	91.42	74.37	73.14	90.95	73.13	84.62	56.43	81.99	73.20
BYOL** (Grill et al. 2020)	400	48.93	90.39	54.43	92.12	75.97	76.65	94.50	74.13	87.81	57.99	82.48	75.95
SDR (BYOL)	400	52.51	**91.12**	56.09	**94.27**	**79.90**	76.33	**94.75**	**76.98**	**89.86**	63.62	**85.12**	$\mathbf{78.23}^{+2.28}$



(a) Performance gains of sub-nets



(b) Histogram of performance gains on Aircraft

Figure 4: (a) Performance gain on kNN accuracy of the 256 sub-nets pre-trained by SDR compared with the baseline trained on the full ImageNet. (b) Histogram of the performance gains on Aircraft dataset. The x-axis is the performance gain on kNN accuracy compared with the baseline, and the y-axis is the number of models.

which further verifies the necessity of using task-customized pre-trained models. In detection, we adopt fast deployment through early stopping. We train the model that performs best at iteration 1000, which takes about 15 minutes on 8*V100 for each model. Compared with the six-hours' fine-tuning with 8*V100, the routing procedure takes much less time to produce a reasonable model.

5.3 Ablation Study

Effects of clustering. Here we analyze the importance of clustering through two controlled experiments. We first train each sub-net with the **total ImageNet(IN)**, with all other modules unchanged, including progressive training and knowledge distillation. We call this model SDR-IN-22.6M(line 2 of Table 3) and name our original model as SDR-Cluster-22.6M(last line of the table). SDR-Cluster-22.6M outperforms SDR-IN-22.6M consistently among all tasks. We achieve a nearly 2.7% mean accuracy improvement, which indicates that training with separate subsets contributes a lot to our SDR. Note that without using clustered

Table 2: Detection transfer results(%) from pre-trained models using Faster R-CNN FPN on PASCAL VOC. The models are trained with all layers fine-tuned. Metrics including AP, AP50 and AP75 are reported.

	AP	AP50	AP75
Supervised	53.26	81.51	59.07
InsDis (Wu et al. 2018)	48.82	76.43	52.40
MoCo-v1 (He et al. 2020)	50.51	78.06	54.55
PIRL (Misra and Maaten 2020)	45.08	72.50	47.80
PCL-v1 (Li et al. 2021)	53.93	81.69	59.33
PCL-v2 (Li et al. 2021)	53.92	81.89	59.35
MoCo-v2 (Chen et al. 2020c)	44.74	72.82	47.01
SimCLR-v1 (Chen et al. 2020a)	52.19	81.36	56.92
SimCLR-v2 (Chen et al. 2020b)	51.42	79.40	55.89
InfoMin (Tian et al. 2020)	44.92	72.72	47.41
SeLa-v2 (Asano et.al 2019)	50.41	80.55	54.35
DeepCluster-v2 (Caron et al. 2018)	51.03	80.93	55.51
SwAV (Caron et al. 2020)	52.07	81.50	56.03
SimSiam (Chen and He 2021)	54.17	80.09	59.58
SDR (SimSiam)	$\mathbf{55.65}^{+1.48}$	$81.16^{+1.07}$	$\mathbf{60.89}^{+1.31}$
BYOL (Grill et al. 2020)	52.75	81.83	58.35
SDR (BYOL)	$53.53^{+0.78}$	$\mathbf{82.69}^{+0.86}$	$59.25^{+0.90}$

subsets, SDR-IN-22.6M is even exceeded by the SimSiam baseline, suggesting that the usage of clustered subsets is the most crucial component in the SDR framework. The other experiment is to train the model with random clusters. Specifically, we split the ImageNet **randomly** into 256 sub-datasets and perform our training procedure subsequently, named as SDR-Random-22.6M(line 3), which shows a degenerated performance over all tasks, indicating the great importance of a reasonable clustering method.

Visualization of clustering. To visualize the clustering procedure, we single out the four clusters that provide the best transfer performance on Standard Cars, FGVC Aircraft, Food-101 and DTD, respectively. We randomly plot four samples of each cluster, as shown in each row of Fig. 5. As can be seen, images in a cluster have similar semantics, suggesting the effectiveness of data clustering. These images

Table 3: Effects of clustering and progressive training. The second column(Dataset) for all SDR models means the dataset used for training each sub-net. SimSiam is always trained by total ImageNet. The third column(Param #) means the parameter count of the model during testing time. The last line SDR-Cluster-22.6M is our proposed model. (1) The first section is the comparison of the models trained by the total ImageNet, random splits and the clusters. (2) Comparison of the lottery ticket theorem (Frankle and Carbin 2018) is provided in section 2. SimSiam is pre-trained under 56.3M and then pruned to 22.6M, which are the exact sizes of the super-net and sub-net in our SDR framework.

	Dataset	Param #	Aircraft	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Flowers	Food	Pets	SUN397	VOC2007	Avg.
SimSiam	IN	23.5M	51.30	87.02	53.80	89.12	68.43	72.99	91.83	67.35	83.64	52.97	83.40	72.90
SDR	IN	22.6M	51.95	86.79	55.62	88.60	67.54	72.09	91.83	67.66	82.44	51.58	81.42	72.50
SDR	Random	22.6M	48.21	86.23	50.25	86.62	64.41	72.29	92.47	64.45	82.21	50.39	80.13	70.70
SimSiam	IN	56.3M	52.31	87.46	54.50	90.05	68.75	73.19	93.11	68.56	83.34	53.27	83.90	73.49
SimSiam	IN	22.6M	43.82	63.00	37.72	80.51	50.10	62.81	73.64	50.24	61.43	32.72	68.62	56.78
SDR	Cluster	22.6M	55.84	87.55	61.06	90.27	71.39	74.47	92.61	68.93	85.03	55.89	85.02	**75.28**

Table 4: Results of kNN accuracy and training time (i.e., GPU hours) for SDR with different number of sub-nets.

# of sub-nets	1	4	16	64	256
kNN accuracy	53.42	55.77	56.83	57.83	58.13
Training time (GPU hours)	260	370	400	460	500

also have semantics similar to the corresponding downstream tasks, which brings improvement of the transfer performance.

Effects of progressive training v.s. lottery ticket theorem (Frankle and Carbin 2018), which suggests that neural networks might rely on internal sub-nets that are significantly smaller than the full parameter count for the majority of their predictive accuracy. In this experiment, we try to make lottery ticket theorem explicit to see how exactly the usage of sub-nets may contribute to the success of SDR. We first train a SimSiam with the same amount of parameters with our SDR super-net whose size is 56.3M, denoted as SimSiam-IN-56.3M. Then we perform pruning, the method to get effective sub-nets used in (Frankle and Carbin 2018), to get SimSiam-IN-22.6M, the 'winning ticket' of SimSiam-IN-56.3M. Here 22.6M is the exact size of SDR sub-net used for each downstream task. As shown in Table 3, SDR-Cluster-22.6M outperforms SimSiam-IN-22.6M dramatically and consistently among all tasks, which indicates that choosing the proper sub-dataset is more crucial than getting the 'winning ticket' of the large model. Furthermore, we notice a large performance drop of SimSiam after pruning, while SDR performs even better than the large model SimSiam-IN-56.3M, demonstrating that simply pruning is not enough to get a better sub-net while SDR is more effective to get better performance. Based on the experiments above, we tend to believe that SDR indeed benefits from sub-datasets other than merely making the lottery ticket theorem explicit.

Number of sub-nets. We analyze how different numbers of sub-nets affect the final results by evaluating the kNN accuracy averaged over 11 downstream tasks. The model with one sub-net is actually the SimSiam baseline. The kNN accuracy and the corresponding training time are reported in Table 4. As can be seen, with a larger number of sub-nets, the kNN accuracy increases significantly. Intuitively, a larger number of sub-nets tends to have a higher probability of providing proper sub-sets for various downstream tasks, yet inevitably requiring a longer training time. The proposed SDR, however, only introduces moderate extra training time with the increasing number of sub-nets, which is applicable for real applications.

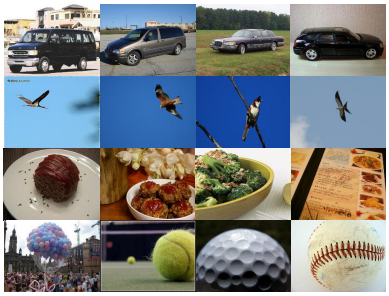


Figure 5: Image samples of different data clusters.

Effects of distillation. We compare our task-customized knowledge distillation method, SiamKD, with the vanilla L2 distillation loss (Hinton, Vinyals, and Dean 2015). We train SDRnet with the L2 distillation loss and SiamKD in Eqn. (4), respectively, following the implementation in Sec. 5.1. For the 11 downstream classification tasks, we compute the average kNN accuracy of the best sub-net, as well as the average standard deviation of the 256 sub-nets for each downstream task. The accuracy of SiamKD is 58.13 ± 2.38, while L2 loss only gets 54.66 ± 0.07. In addition to the inferior performance, the L2 distillation results in a small standard deviation of kNN accuracy and homogenized sub-nets, while SiamKD maintains the feature diversity of sub-nets, which is essential for providing a task-customized model. In practice, we also find SiamKD helps to provide better feature representations and stabilize the training process.

6 Conclusion

In this work, we first identify the negative transfer phenomenon in SSL that involving semantic-irrelevant data in pre-training may degenerate the downstream performance. To address this issue, we propose a novel task-customized SSL paradigm called Scalable Dynamic Routing (SDR). SDR first cluster the training data and then train each sub-net with a different cluster through a data-aware progressive training framework. Finally, customized sub-nets are deployed to different downstream tasks efficiently. In the experiments, we succeed in training 256 sub-nets simultaneously, with a total training cost less than twice of the SSL baseline that provides only one pre-trained model, achieving SOTA results on average accuracy among 11 downstream classification tasks and AP on PASCAL VOC detection task.

References

Asano, Y.; Rupprecht, C.; and Vedaldi, A. 2019. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*.

Bachman, P.; Hjelm, R. D.; and Buchwalter, W. 2019. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101 – Mining Discriminative Components with Random Forests. In *European Conference on Computer Vision*, 446–461.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Campos, V.; Jou, B.; Giró-i Nieto, X.; Torres, J.; and Chang, S.-F. 2017. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

Cao, S.; Ma, L.; Xiao, W.; Zhang, C.; Liu, Y.; Zhang, L.; Nie, L.; and Yang, Z. 2019. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *Computer Vision and Pattern Recognition*, 11216–11225.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 132–149.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 9912–9924.

Chen, K.; Hong, L.; Xu, H.; Li, Z.; and Yeung, D.-Y. 2021. Multisiam: Self-supervised multi-instance siamese representation learning for autonomous driving. In *International Conference on Computer Vision*, 7546–7554.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *International conference on Machine Learning*, 1597–1607.

Chen, T.; Kornblith, S.; Swersky, K.; Norouzi, M.; and Hinton, G. 2020b. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, 22243–22255.

Chen, X.; Fan, H.; Girshick, R.; and He, K. 2020c. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Chen, X.; and He, K. 2021. Exploring simple siamese representation learning. In *Computer Vision and Pattern Recognition*, 15750–15758.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; and Vedaldi, A. 2014. Describing textures in the wild. In *Computer Vision and Pattern Recognition*, 3606–3613.

Cole, E.; Yang, X.; Wilber, K.; Mac Aodha, O.; and Belongie, S. 2021. When Does Contrastive Visual Representation Learning Work? *arXiv preprint arXiv:2105.05837*.

Cuturi, M. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, 2292–2300.

Ericsson, L.; Gouk, H.; and Hospedales, T. M. 2021. How well do self-supervised models transfer? In *Computer Vision and Pattern Recognition*, 5414–5423.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 303–338.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition Workshop*, 178–178.

Figurnov, M.; Collins, M. D.; Zhu, Y.; Zhang, L.; Huang, J.; Vetrov, D.; and Salakhutdinov, R. 2017. Spatially adaptive computation time for residual networks. In *Computer Vision and Pattern Recognition*, 1039–1048.

Frankle, J.; and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Gao, D.; Yang, W.; Zhou, H.; Wei, Y.; Hu, Y.; and Wang, H. 2021. Network Clustering for Multi-task Learning. *arXiv preprint arXiv:2101.09018*.

Goyal, P.; Caron, M.; Lefaudeux, B.; Xu, M.; Wang, P.; Pai, V.; Singh, M.; Liptchinsky, V.; Misra, I.; Joulin, A.; et al. 2021. Self-supervised pre-training of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*.

Han, Y.; Huang, G.; Song, S.; Yang, L.; Wang, H.; and Wang, Y. 2021. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*.

Hansen, C.; Hansen, C.; Alstrup, S.; Simonsen, J. G.; and Lioma, C. 2019. Neural speed reading with structural-jump-lstm. *arXiv preprint arXiv:1904.00761*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Computer Vision and Pattern Recognition*, 9729–9738.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Huh, M.; Agrawal, P.; and Efros, A. A. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614*.

Kim, D.; Cho, D.; Yoo, D.; and Kweon, I. S. 2018. Learning image representations by completing damaged jigsaw puzzles. In *IEEE Winter Conference on Applications of Computer Vision*, 793–802.

Krause, J.; Deng, J.; Stark, M.; and Fei-Fei, L. 2013. Collecting a large-scale dataset of fine-grained cars. In *Workshop on Fine-Grained Visual Categorization*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*.

Li, J.; Zhou, P.; Xiong, C.; and Hoi, S. C. H. 2021. Prototypical Contrastive Learning of Unsupervised Representations. In *International Conference on Learning Representations*.

Li, X.; Liu, Z.; Luo, P.; Change Loy, C.; and Tang, X. 2017. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Computer Vision and Pattern Recognition*, 3193–3202.

Liu, L.; and Deng, J. 2018. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *AAAI Conference on Artificial Intelligence*, 3675–3682.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

McDermott, M.; Nestor, B.; Kim, E.; Zhang, W.; Goldenberg, A.; Szolovits, P.; and Ghassemi, M. 2021. A comprehensive EHR time-series pre-training benchmark. In *Proceedings of the Conference on Health, Inference, and Learning*, 257–278.

Miller, G. A. 1998. *WordNet: An electronic lexical database*. MIT press.

Misra, I.; and Maaten, L. v. d. 2020. Self-supervised learning of pretext-invariant representations. In *Computer Vision and Pattern Recognition*, 6707–6717.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729.

Odena, A.; Lawson, D.; and Olah, C. 2017. Changing model behavior at test-time using reinforcement learning. *arXiv preprint arXiv:1702.07780*.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *Computer Vision and Pattern Recognition*, 3498–3505.

Tao, J.; Thakker, U.; Dasika, G.; and Beu, J. 2019. Skipping rnn state updates without retraining the original model. In *Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems*, 31–36.

Tian, Y.; Henaff, O. J.; and Oord, A. v. d. 2021. Divide and contrast: self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*.

Tian, Y.; Sun, C.; Poole, B.; Krishnan, D.; Schmid, C.; and Isola, P. 2020. What Makes for Good Views for Contrastive Learning? In *Advances in Neural Information Processing Systems*.

Wang, H.; Kembhavi, A.; Farhadi, A.; Yuille, A. L.; and Rastegari, M. 2019. Elastic: Improving cnns with dynamic scaling policies. In *Computer Vision and Pattern Recognition*, 2258–2267.

Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; and Girshick, R. 2019. Detectron2.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Computer Vision and Pattern Recognition*, 3733–3742.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition*, 3485–3492.

Xu, H.; Zhang, X.; Li, H.; Xie, L.; Xiong, H.; and Tian, Q. 2020. Hierarchical Semantic Aggregation for Contrastive Representation Learning. *arXiv preprint arXiv:2012.02733*.