Slimmable Networks for Contrastive Self-supervised Learning

Shuai Zhao · Linchao Zhu · Xiaohan Wang · Yi Yang

Received: date / Accepted: date

Abstract Self-supervised learning makes significant progress in pre-training large models, but struggles with small models. Mainstream solutions to this problem rely mainly on knowledge distillation, which involves a two-stage procedure: first training a large teacher model and then distilling it to improve the generalization ability of smaller ones. In this work, we introduce another one-stage solution to obtain pre-trained small models without the need for extra teachers, namely, slimmable networks for contrastive self-supervised learning (SlimCLR). A slimmable network consists of a full network and several weight-sharing sub-networks, which can be pre-trained once to obtain various networks, including small ones with low computation costs. However, interference between weight-sharing networks leads to severe performance degradation in self-supervised cases, as evidenced by gradient magnitude imbalance and gradient direction divergence. The former indicates that a small proportion of parameters produce dominant gradients during backpropagation, while the main parameters may not be fully optimized. The latter shows that the gradient direction is disordered, and the optimization process is unstable. To address these issues, we introduce three techniques to make the main parameters produce dominant gradients and sub-networks have consistent outputs. These techniques include slow start training of sub-networks, online distillation, and loss re-weighting according to model sizes. Furthermore, theoretical results are presented to demonstrate that a single slimmable linear layer is sub-optimal during linear evaluation. Thus a switchable linear probe layer is applied during linear evaluation. We instantiate SlimCLR with typical contrastive learning frameworks and achieve better performance than previous arts with fewer parameters and FLOPs. The code is available at https://github.com/mzhaoshuai/SlimCLR.

1 Introduction

Over the last decade, deep learning achieves significant success in various fields of artificial intelligence, primarily due to a significant amount of manually labeled data. However, manually labeled data is expensive and far less available than unlabeled data in practice. To overcome the constraint of costly annotations, self-supervised learning (Dosovitskiy et al., 2016; Wu et al., 2018; van den Oord et al., 2018; He et al., 2020; Chen et al., 2020a) aims to learn transferable representations for downstream tasks by training networks on unlabeled data. There has been significant progress in large models, which are larger than ResNet-50 (He et al., 2016) that has roughly 25M parameters. For example, ReLICv2 (Tomasev et al., 2022) achieves an accuracy of 77.1% on ImageNet (Russakovsky et al., 2015) under a linear evaluation protocol with ResNet-50, outperforming the supervised baseline 76.5%.
An interesting question arises: can we obtain multiple small models through one-time pre-training to meet various computation scenarios without the need for additional teachers? Inspired by the success of slimmable networks in supervised learning (Yu et al., 2019b, a), we develop a novel one-stage method for obtaining pre-trained small models without extra large models: slimmable networks for contrastive self-supervised learning, referred to as SlimCLR. A slimmable network comprises a full network and several weight-sharing sub-networks with different widths, where the width represents the number of channels in a network. The slimmable network can operate at various widths, allowing flexible deployment on different computing devices. Therefore, we can obtain multiple networks, including small ones for low computing scenarios, through one-time pre-training. Weight-sharing networks can also inherit knowledge from large ones via shared parameters, resulting in better generalization performance.

In contrast to the success of the large model pre-training, self-supervised learning with small models lags behind. For instance, supervised ResNet-18 with 12M parameters achieves an accuracy of 72.1% on ImageNet, but its self-supervised result with MoCov2 (Chen et al., 2020c) is only 52.5% (Fang et al., 2021). The gap is nearly 20%. To bridge the significant performance gap between supervised and self-supervised small models, popular methods (Koohpayegani et al., 2020; Fang et al., 2021; Gao et al., 2022; Xu et al., 2022) mainly focus on knowledge distillation, where they attempt to transfer the knowledge of a self-supervised large model into small ones. Nevertheless, such methodology has a two-stage procedure: first train an additional large model, and then train a small model to mimic the large one. Shi et al. (2022) investigate the impact of various components in contrastive self-supervised learning with small models to enhance performance without distillation signals. They attain significant progress but are not superior to distillation methods. Moreover, these methods only yield a single small model during pre-training, restricting the versatility of the pre-trained model in different resource-constrained scenarios.

Fig. 1: A slimmable ResNet-50 in supervised (left) and self-supervised (right) manners. ResNet-50 \( [1.0, 0.75, 0.5, 0.25] \) means this slimmable network can switch at width \( [1.0, 0.75, 0.5, 0.25] \). The width 0.25 represents that the number of channels is scaled by 0.25 of the full network.
ate dominant gradients during backpropagation. To avoid conflicts in gradient directions of weight-sharing networks, they should have consistent outputs. To achieve these goals, we propose three simple yet effective techniques during pre-training to alleviate interference among networks. 1) We adopt a slow start strategy for sub-networks. The networks and pseudo supervision of contrastive learning are both unstable and fast updating at the start of training. To avoid interference making the situation worse, we only train the full model initially. After the full model becomes relatively stable, sub-networks can inherit learned knowledge via shared parameters and begin with better initialization. 2) We apply online distillation to make all sub-networks consistent with the full model, thereby eliminating the gradient divergence of networks. The predictions of the full model serve as consistent guidance for all sub-networks. 3) We re-weight the losses of networks according to their widths to ensure that the full model dominates the optimization process. Additional, theoretical results are provided to demonstrate that a single slimmable linear layer is sub-optimal during linear evaluation. Therefore, we adopt a switchable linear probe layer to avoid the interference caused by parameter-sharing.

We instantiate two algorithms for SlimCLR with typical contrastive learning frameworks, i.e., MoCov2 (Chen et al., 2020c) and MoCov3 (Chen et al., 2021). Extensive experiments on ImageNet (Russakovsky et al., 2015) show that SlimCLR achieves significant performance improvements compared to previous arts with fewer parameters and FLOPs.

2 Related Works

**Self-supervised learning** Self-supervised learning aims to learn transferable representations for downstream tasks from the input data itself. According to Liu et al. (2020), self-supervised methods can be summarized into three main categories according to their objectives: generative, contrastive, and generative-contrastive (adversarial). Methods belonging to the same category can be further classified by the difference between pretext tasks. Given input $x$, generative methods encode $x$ into an explicit vector $z$ and decode $z$ to reconstruct $x$ from $z$, e.g., auto-regressive (van den Oord et al., 2016a,b), auto-encoding models (Ballard, 1987; Kingma and Welling, 2014; Devlin et al., 2019; He et al., 2022). Contrastive learning methods encoder input $x$ into an explicit vector $z$ to measure similarity. The two mainstream methods below this category are context-instance contrast (infoMax Hjelm et al. (2019), CPC van den Oord et al. (2018), AMDIM Bachman et al. (2019)) and instance-instance contrast (DeepCluster Caron et al. (2018), MoCo He et al. (2020); Chen et al. (2021), SimCLR Chen et al. (2020a,b), SimSiam Chen and He (2021)). Generative-contrastive methods generate a fake sample $x'$ from $x$ and try to distinguish $x'$ from real samples, e.g., DCGANs Radford et al. (2016), inpainting Pathak et al. (2016), and colorization Zhang et al. (2016).

**Self-supervised small models** While self-supervised learning has made significant progress with large models like ResNet-50 (Chen et al., 2021; Tomasev et al., 2022), small models struggle with common self-supervised pretext tasks. There exists a notable performance gap between self-supervised small models and their supervised counterparts in downstream tasks. To address this, typical solutions employ knowledge distillation (Hinton et al., 2015), which involves incorporating an additional self-supervised large model to guide the pre-training of the small model. The key lies in aligning the predicted distributions of the student and the teacher, which arise from different data views of the same sample. Alignment is generally achieved by minimizing their Kullback–Leibler (KL) divergence (Koolpayegani et al., 2020; Fang et al., 2021). SimDis (Gu et al., 2021) and DisCo (Gao et al., 2022) substitute the KL divergence with $\ell_2$ distance. Additionally, to enhance generalization, BINGO (Xu et al., 2022) transfers relationships among similar samples produced by the teacher to the predictions of the student. Beyond distillation methods, Shi et al. (2022) improve the performance of self-supervised small models by carefully tuning various components of contrastive self-supervised learning. While they achieve considerable progress, their results does not surpass those of distillation methods.

**Slimmable networks** Slimmable networks are first proposed to achieve instant and adaptive accuracy-efficiency trade-offs on different devices (Yu et al., 2019). A slimmable network can execute at different widths during runtime. Following the pioneering work, universally slimmable networks (Yu and Huang, 2019b) develop systematic training approaches to allow slimmable networks to run at arbitrary widths. AutoSlim (Yu and Huang, 2019a) further achieves one-shot architecture search for channel numbers under a certain computation budget. MutualNet (Yang et al., 2020) trains slimmable networks using different input resolutions to learn multi-scale representations. Dynamic slimmable networks (Li et al., 2022, 2021) change the number of channels of each layer on the fly according to the input. In contrast to weight-sharing sub-networks in
slimmable networks, some methods aim to train multiple sub-networks with independent parameters (Zhao et al., 2022b). A relevant concept of slimmable networks in network pruning is network slimming (Liu et al., 2017; Chavan et al., 2022; Wang et al., 2021), which aims to achieve channel-level sparsity for better computation efficiency.

**Knowledge distillation** Knowledge distillation (Hinton et al., 2015) aims to transfer the knowledge of a large network to a small one, and can be roughly categorized into two types: logits distillation (Hinton et al., 2015; Zhang et al., 2018; Guo, 2022; Zhao et al., 2022a; Mirzadeh et al., 2020) and intermediate feature distillation (Park et al., 2019; Tian et al., 2020; Peng et al., 2019). The former only requires the student to mimic the output of the teacher, while the latter also aligns the intermediate features of the teacher and student. Intermediate feature distillation methods generally achieve better performance than logits distillation methods. Recently, several methods (Guo, 2022; Zhao et al., 2022a) try to find out the limit of logits distillation and improve its performance. In this paper, online distillation is a kind of logits distillation. Besides, in a slimmable network, distillation also occurs between a large network and a small network via the shared parameters.

3 Method

3.1 Description of SlimCLR

We develop two instantial algorithms for SlimCLR with typical contrastive learning frameworks MoCov2 and MoCov3 (Chen et al., 2020c, 2021). As shown in Figure 2a (right), a slimmable network with n widths $w_1, \ldots, w_n$ contains multiple weight-sharing networks $f_{\theta_{w_1}}, \ldots, f_{\theta_{w_n}}$, which are parameterized by learnable weights $\theta_{w_1}, \ldots, \theta_{w_n}$, respectively. Each network $f_{\theta_{w_i}}$ in the slimmable network has its own set of weights $\Theta_{w_i}$ and $\theta_{w_i} \in \Theta_{w_i}$. A network with a small width shares the weights with large ones, namely, $\Theta_{w_j} \subset \Theta_{w_i}$ if $w_j < w_i$. Generally, we assume $w_j < w_i$ if $j > i$, i.e., $w_1, \ldots, w_n$ arrange in descending order, and $\theta_{w_i}$ represent the parameters of the full model.

We first illustrate the learning process of SlimCLR-MoCov2 in Figure 2a. Given a set of images $D$, an image $x$ sampled uniformly from $D$, and one distribution of image augmentation $T$, SlimCLR produces two data views $	ilde{x}_1 = t(x)$ and $	ilde{x}_2 = t'(x)$ from $x$ by applying augmentations $t \sim T$ and $t' \sim T$, respectively. For the first view, SlimCLR outputs multiple representations $h_{\theta_{w_1}}, \ldots, h_{\theta_{w_n}}$ and predictions $z_{a_{w_1}}, \ldots, z_{a_{w_n}}$, where $h_{\theta_{w_i}} = f_{\theta_{w_i}}(\tilde{x}_1)$ and $z_{a_{w_i}} = g_{w_i}(h_{\theta_{w_i}})$. $g$ is a stack of slimmable linear transformation layers, i.e., a slimmable version of the MLP head in MoCov2 and SimCLR (Chen et al., 2020a). For the second view, SlimCLR only outputs a single representation from the full model $h_{\xi_{w_1}} = f_{\xi_{w_1}}(\tilde{x}_2)$ and prediction $z_{\xi_{w_1}} = g_{w_1}(h_{\xi_{w_1}})$. We minimize the InfoNCE (van den Oord et al., 2018) loss with respect to $\theta_{w_i}$ to maximize

1 In contrast to MoCov2 and SimCLR, where the output of the model $z_a$ is referred to as the projection, in this work, we refer to the final output of the model as the prediction. This is to maintain consistency with the notation used in SlimCLR-MoCov3 and to simplify the formulas.
the similarity of positive pairs $z_{θ_{u_1}}$ and $z_{ξ_{u_1}}$:

$$L_{θ_{u_1}} = -\log \frac{e^{z_{θ_{u_1}} \cdot w_{i} / τ_1}}{e^{z_{θ_{u_1}} \cdot w_{i} / τ_1} + \sum_{z} e^{z \cdot w_{i} / τ_1}},$$  \quad (1)

where $z_{θ_{u_1}} = z_{θ_{u_1}} / \|z_{θ_{u_1}}\|_2$, $z_{ξ_{u_1}} = z_{ξ_{u_1}} / \|z_{ξ_{u_1}}\|_2$, $τ_1$ is a temperature hyper-parameter, and $\{z\}$ are features of negative samples. In SlimCLR-MoCov2, $\{z\}$ are obtained from a queue which is updated every iteration during training by $z_{ξ_{u_1}}$, following the approach used in MoCov2. Without any regularization during training, the overall objective is the sum of losses of all networks with various widths:

$$L_θ = \sum_{i=1}^{n} L_{θ_{u_i}}.$$  \quad (2)

Here $ξ$ is updated by $θ$ every iteration using a momentum coefficient $m ∈ [0, 1)$, as follows: $ξ ← mξ + (1 − m)θ$.

Compared to SlimCLR-MoCov2, SlimCLR-MoCov3 has an additional projection process. Firstly, it projects the representation to another high dimensional space, and then makes predictions. The projector $q$ is a stack of slimmable linear transformation layers. SlimCLR-MoCov3 also adopts the InfoNCE loss, but the negative samples come from other samples in the mini-batch.

After contrastive learning, we only keep $f_{θ_{u_1}}, \ldots, f_{θ_{w_n}}$ and abandon other components.

3.2 Interference of networks and solutions

In Figure 1, it can be observed that a vanilla implementation of SlimCLR experiences significant performance degradation due to interference of weight-sharing networks. In this section, we will discuss the consequences of this interference and present solutions to address it.

3.2.1 Gradient magnitude imbalance

Gradient magnitude imbalance refers to the phenomenon where a small fraction of parameters receives dominant gradients during backpropagation. For example, in a slimmable network with widths $[1.0, 0.5, 0.25]$, $\ell_2$ norm of $θ_{0.25} = \|\nabla_{θ_{0.25}} L_θ\|_2$ maybe larger than that of the remaining parameters $\|\nabla_{θ_{1,0.05,0.25}} L_θ\|_2$, even though $θ_{0.25}$ only counts for approximately 6% of the total parameters. This is because the gradients of different losses accumulate as follows:

$$\nabla_{θ_{w_n}} L_θ = \frac{∂L_θ}{∂θ_{w_n}} = \sum_{i=1}^{n} \frac{∂L_{θ_{u_i}}}{∂θ_{w_n}}.$$  \quad (3)

Assuming that the gradients of a single loss have a similar magnitude distribution, the accumulation of gradients from different losses increases the gradient norm of the shared parameters.

We evaluate the gradient magnitude imbalance by calculating the ratios of gradient norms during training. Figure 3 shows the ratios of $\|\nabla_{θ_{0.0}} L_θ\|_2$ and $\|\nabla_{θ_{0.05,0.25}} L_θ\|_2$ versus $\|\nabla_{θ_{0.25}} L_θ\|_2$, separately. The ratio of their numbers of parameters is $\frac{θ_{0.0} \setminus θ_{0.25}}{θ_{0.05,0.25}} ≈ 15$, where $θ_{0.05,0.25} ∈ θ_{0.0} \setminus θ_{0.25}$. In Figure 3a, both ratios of gradient norms are around 3.5, indicating that the majority of parameters obtain a large gradient norm and dominate the optimization process. However, in Figure 3b & 3c, when training a slimmable network with widths $[1.0, 0.5, 0.25]$, $\|\nabla_{θ_{0.25}} L_θ\|_2$ becomes close or larger than $\|\nabla_{θ_{0.05,0.25}} L_θ\|_2$ despite having much fewer parameters.

Gradient magnitude imbalance is more pronounced in self-supervised cases. In the case of supervised learning shown in Figure 3b, $\|\nabla_{θ_{0.05,0.25}} L_θ\|_2$ is initially close to $\|\nabla_{θ_{0.25}} L_θ\|_2$, but the former increases as training progresses. By contrast, for vanilla SlimCLR-MoCov2 in Figure 3c, $\|\nabla_{θ_{0.05,0.25}} L_θ\|_2$ is smaller than the other most time. A conjecture is that the pretext task — instance discrimination is more challenging than supervised classification. Consequently, small networks with limited capacity face difficulty in converging, leading to large losses and dominant gradients.

3.2.2 Gradient direction divergence

In addition to the imbalance in gradient magnitudes, the gradient directions of different weight-sharing networks can also conflict with each other. These conflicts lead to disordered gradient directions of the full model, which we refer to as gradient direction divergence.

We visualize the principal gradient directions in Figure 4. Specifically, we collect the gradients with respect to parameters in the last linear layer during training; after training, we perform PCA on these gradients and calculate their projections on the first two principal components (Li et al., 2018). In Figure 4b, before the slow start point, we only train the full model. In this case, the gradient directions are stable and consistent during training. By contrast, after the slow start point, gradient directions become disordered due to conflicts in weight-sharing networks. We also show the gradient directions when training a slimmable network in a supervised case in Figure 4a. In the supervised case, gradient direction divergence is also inevitable. Nevertheless, the gradient direction divergence in the supervised case is less significant compared to the self-supervised case in Figure 4b. In supervised cases, all networks have consis-
3.2.3 Solutions to network interference

To address the issue of gradient magnitude imbalance, it is necessary for the majority of parameters to dominate the optimization process, namely, the ratios of gradient norms in Figure 3 should be large. To alleviate the problem of gradient direction divergence, constraints can be placed on the similarity scores in Eq. (1) to prevent predictions from varying significantly from each other. To achieve these goals, we develop three simple yet effective pre-training techniques: slow start, online distillation, and loss reweighting. Furthermore, we provide theoretical evidence demonstrating that a weight-sharing linear layer is suboptimal during linear evaluation, and a switchable linear probe layer is a better alternative.

Slow start At the start of training, the model and contrastive similarity scores in Eq. (1) update quickly, resulting in an unstable optimization procedure. To prevent interference between networks making the situation harder, we employ a slow start technique in which we only train the full model, updating \( \theta_{1,0} \) by \( \nabla_{\theta_{1,0}} \mathcal{L}_\theta \), for the first \( S \) epochs. As shown in Figure 3d, the ratios of gradient norms are large prior to the \( S \)-th epoch, but dramatically decrease after the slow start point. During the first \( S \) epochs, the full model can learn knowledge from the data without disturbances, and sub-networks can inherit this knowledge via the shared parameters and begin training with a better initialization. Similar approaches are also adopted in some one-shot NAS methods (Cai et al., 2020; Yu et al., 2020).

Online distillation The full network has the highest capacity to acquire knowledge from the data, and its predictions can provide guidance to all sub-networks in resolving gradient direction conflicts among weight-sharing networks. Namely, sub-networks should learn from the full network. Following Yu and Huang (2019b), we minimize the KL divergence between the estimated
Contrastive self-supervised learning, a single slimmable linear probe layer cannot achieve several complex mappings from different representations to the same object classes simultaneously. The failure is because the learned representations in Figure 2 do not meet the requirement discussed in Appendix A. In this case, we propose to use a switchable linear probe layer during linear evaluation, where each network in the slimmable network has its own linear probe layer. This allows each network to learn its unique mapping from its representation to the object classes.

4 Experiments

4.1 Experimental details

Dataset We train SlimCLR on ImageNet (Russakovsky et al., 2015), which contains 1.28M training and 50K validation images. During pre-training, we use training images without labels.

Pre-training of SlimCLR-MoCov2 By default, we use a total batch size 1024, an initial learning rate 0.2, and weight decay $1 \times 10^{-4}$. We adopt the SGD optimizer with a momentum 0.9. A linear warm-up and cosine decay policy (Goyal et al., 2017; He et al., 2019) for learning rate is applied, and the warm-up epoch is 10. The temperatures are $\tau_1 = 0.2$ for InfoNCE and $\tau_2 = 5.0$ for online distillation. Without special mentions, other settings including data augmentations, queue size (65,536), and feature dimension (128) are the same as the counterparts of MoCov2 (Chen et al., 2020c). The slow start epoch $S$ of sub-networks is set to be half of the number of total epochs.

Pre-training of SlimCLR-MoCov3 We use a total batch size 1024, an initial learning rate 1.2, and weight decay $1 \times 10^{-6}$. We adopt the LARS (You et al., 2017) optimizer with momentum 0.9 and weight decay $1 \times 10^{-6}$.
4.2 Linear evaluation on ImageNet

The results of SlimCLR on ImageNet are presented in Table 1. Despite our efforts to mitigate the interference of weight-sharing networks, as described in Section 3.2, slimmable training unavoidably causes a drop in performance for the full model. Moreover, when training for more epochs, the performance degradation becomes more pronounced. However, it is important to note that such degradation is not unique in the self-supervised learning rate policy with warm-up epoch 10. The temperatures are $\tau_1 = 1.0$ and $\tau_2 = 1.0$. The slow start epoch $S$ is half of the total epochs. One different thing is that we increase the initial learning rate to 3.2 after $S$ epochs. Pre-training is all done with mixed precision (Miceviceicu et al., 2018).

**Linear evaluation** Following the general linear evaluation protocol (Chen et al., 2020a; He et al., 2020), we add new linear layers on the backbone and freeze the backbone during evaluation. We also apply online distillation with a temperature $\tau_2 = 1.0$ when training these linear layers. For evaluation of SlimCLR-MoCov2, we use a total batch size 1024, epochs 100, and an initial learning rate 60, which is decayed by 0.4 with cosine decay policy.

| Method | Backbone | Teacher | Top-1 | Top-5 | Epochs | #Params | #FLOPs |
|--------|----------|---------|-------|-------|--------|---------|--------|
| Supervised | R-50 | x | 76.6 | 93.2 | 100 | 25.6 M | 4.1 G |
| R-34 | | | 75.0 | - | - | 21.8 M | 3.7 G |
| R-18 | | | 72.1 | - | - | 11.9 M | 1.8 G |
| R-50 | | | 76.3 (5.4) | 92.9 | 100 | 21.8 M | 4.1 G |
| R-50, 75 | | | 74.9 | 92.3 | 100 | 14.7 M | 2.3 G |
| R-50, 5 | | | 72.2 | 90.8 | 100 | 6.9 M | 1.1 G |
| R-50, 25 | | | 64.4 | 86.0 | 100 | 2.0 M | 278 M |

| Baseline (individual networks trained with MoCov2) | | | | | | | |
|--------|---------|-------|-------|--------|---------|--------|
| R-50 | x | 67.5 | 87.8 | 200 | 25.6 M | 4.1 G |
| R-50, 75 | | | 67.2 | 87.5 | 200 | 14.7 M | 2.3 G |
| R-50, 5 | | | 64.3 | 85.8 | 200 | 6.9 M | 1.1 G |
| R-50, 25 | | | 47.9 | 72.8 | 200 | 2.0 M | 278 M |

| MoCov2 (2020, preprint) | | | | | | | |
|--------|---------|-------|-------|--------|---------|--------|
| R-50 | | | 71.1 | - | 800 | 14.7 M | 2.3 G |
| R-50, 75 | | | 70.4 | 89.6 | 800 | 6.9 M | 1.1 G |
| R-50, 5 | | | 71.3 | 90.8 | 300 | |
| R-50, 25 | | | 69.7 | 89.4 | 300 | |

Table 1: Linear evaluation results of SlimCLR with ResNet-50 trained with MoCov2 on ImageNet. Through only one-time pre-training, SlimCLR produces multiple different small models without extra large teacher models. Plus, SlimCLR outperforms previous methods with fewer parameters and FLOPs. The performance degradation when training a slimmable network is shown in cyan. Eff-B0 is short for EfficientNet-B0 (Tan and Le, 2019).
Table 2: Transfer learning results of SlimCLR pre-trained models on COCO val2017 set. Bounding-box AP ($\text{AP}^{\text{bb}}$) for object detection and mask AP ($\text{AP}^{\text{mk}}$) for instance segmentation. The parameters of backbones during pre-training are also presented.

| pre-train          | backbone | #Params | AP$^{\text{bb}}$ | AP$^{\text{bb}}_{50}$ | AP$^{\text{bb}}_{75}$ | AP$^{\text{mk}}$ | AP$^{\text{mk}}_{50}$ | AP$^{\text{mk}}_{75}$ |
|--------------------|----------|---------|-----------------|------------------------|------------------------|-----------------|------------------------|------------------------|
| supervised (Yu et al., 2019) | R-50 | 25.5 M | 37.4 | 59.6 | 40.5 | 34.9 | 56.5 | 37.3 |
|                     | R-50, 0.75 | 14.7 M | 36.7 | 58.7 | 39.3 | 34.3 | 55.4 | 36.1 |
|                     | R-50, 0.5 | 6.9 M | 34.7 | 56.3 | 36.8 | 32.6 | 53.1 | 34.1 |
|                     | R-50, 0.25 | 2.0 M | 30.2 | 50.3 | 31.5 | 28.6 | 47.5 | 29.9 |
| MoCo (He et al., 2020) | R-50 | 25.6 M | 38.5 | 58.9 | 42.0 | 35.1 | 55.9 | 37.7 |
|                     | R-34 | 21.8 M | 38.4 | 57.0 | 41.0 | 33.3 | 53.6 | 35.4 |
|                     | R-18 | 11.9 M | 32.0 | 51.0 | 34.7 | 29.6 | 48.2 | 31.5 |
| BINGO (Xu et al., 2022) | R-50, 0 | 25.6 M | 38.6 | 60.1 | 42.0 | 35.7 | 57.2 | 38.0 |
|                     | R-50, 0.75 | 14.7 M | 37.7 | 59.3 | 40.9 | 34.9 | 56.3 | 37.4 |
|                     | R-50, 0.5 | 6.9 M | 35.8 | 56.9 | 38.6 | 33.2 | 54.2 | 35.3 |
|                     | R-50, 0.25 | 2.0 M | 31.1 | 51.0 | 33.3 | 29.1 | 48.1 | 30.9 |

4.3 Transfer learning

In this section, we assess the transfer learning ability of SlimCLR on object detection and instance segmentation, using the Mask R-CNN (He et al., 2017) with FPN (Lin et al., 2017) architectures as the supervised slimmable network (Yu et al., 2019). We fine-tune all parameters, including batch normalization (Ioffe and Szegedy, 2015), end-to-end on the COCO 2017 dataset (Lin et al., 2014) using a default 1x training schedule from MMDetection (Chen et al., 2019). During training, we apply synchronized batch normalization across different GPUs. The backbone is a ResNet-50 [1,0.0,0.75,0.5,0.25] pre-trained via SlimCLR-MoCov2 for 800 epochs. Our results, presented in Table 2, demonstrate that SlimCLR-MoCov2 outperforms the supervised baseline in terms of transfer learning ability. Notably, by training only once, SlimCLR produces multiple networks that surpass previous methods pre-trained with large teachers while requiring fewer parameters. Particularly, SlimCLR outperforms previous distillation-based pre-training methods significantly in the instance segmentation task. These findings demonstrate the effectiveness of pre-training with SlimCLR.

4.4 Discussion

In this section, we will discuss the influences of different components in SlimCLR.

**switchable linear probe layer** Table 3a demonstrates the notable impact of the switchable linear probe layer compared to a slimmable linear probe layer during linear evaluation. The introduction of a switchable linear probe layer results in significant improvements in accuracy. When only one slimmable layer is used, the interference between weight-sharing linear layers is unavoidable as discussed in Appendix A. The learned representations of pre-trained models do not satisfy the requirements in Appendix A.

**slow start and training time** Table 3b presents experiments conducted with and without the slow start technique. The use of slow start prevents interference between weight-sharing networks during the initial stages of training, allowing the system to quickly reach a stable point during optimization. Consequently, sub-networks start with better initialization and achieve...
Table 3: Ablation experiments with SlimCLR-MoCov2 on ImageNet. The experiment in a former table serves as a baseline for the consequent table.

| Model | slimmable | switchable |
|-------|-----------|------------|
|       | Top-1     | Top-5      | Top-1     | Top-5      |
| R-50x1 | 64.8      | 80.1      | 65.6      | 86.8      |
| R-50x75 | 63.4      | 85.3      | 64.3      | 80.0      |
| R-50x5 | 59.6      | 82.9      | 61.3      | 84.1      |
| R-50x25 | 53.0      | 77.8      | 54.5      | 79.1      |

We compared four different distillation losses, including the classical mean-square-error (MSE) and KL divergence (KD), as well as two recent approaches: ATKD (Guo, 2022) and DKD (Zhao et al., 2022a). ATKD reduces the difference in sharpness between distributions of teacher and student to help the student better mimic the teacher. DKD decouples the classical knowledge distillation objective function into target class and non-target class knowledge distillation to achieve more effective and flexible distillation. In Table 3c, these four distillation losses make trivial differences in our context.

Upon combining the outcomes of distillation with ResNet-50[1.0,0.0,5.0,2.5] in Figure 3e, we observe that the primary advantage of distillation lies in improving the performance of the full model, while the enhancements in sub-networks are relatively marginal. This contradicts the original objective of knowledge distillation, which aims to transfer knowledge from larger models to smaller ones and enhance their performance. One plausible explanation for this outcome is that the sub-networks in a slimmable network already inherit the knowledge from the full network through shared parameters, rendering logit distillation less useful. In our context, the primary function of online distillation is to alleviate the interference among weight-sharing sub-networks, as demonstrated in Figure 3e & 4c.

We also test the influence of different temperatures in online distillation, i.e., $\tau_2$ in Eq. (4). Following classical KD (Hinton et al., 2015), we choose $\tau_2 \in \{3.0, 4.0, 5.0, 6.0\}$. The results are presented in Table 3d. The choices of temperatures make trivial differences. By contrast, SEED (Fang et al., 2021) uses a small temperature 0.01 for the teacher to get sharp distribution and a relatively large temperature 0.2 for the student. BINGO (Xu et al., 2022) adopts a single temperature 0.2. These choices are different from ours, but we observed that SlimCLR is more robust to the choice of temperature.

Table 4: The pre-training time for 200 epochs. #V100 is the number of Tesla V100 GPUs. The slow start epoch $S = 100$. We adopt NVIDIA DALI (NVIDIA, 2021) to accelerate the data augmentation pipeline.

| Method             | Backbone | #V100 | Time (h) |
|--------------------|----------|-------|----------|
| SwAV (Caron et al.) | R-50x2   | 128   | 36.0     |
| SwAV (Caron et al.) | R-50     | 64    | 12.5     |
| MoCov2 (Shi et al.) | R-50     | 8     | 42.4     |
| MoCov2 (Shi et al.) | R-18     | 8     | 40.9     |
| SlimCLR-MoCov2     | R-50[0.0] | 8     | 19.1     |
| SlimCLR-MoCov2     | R-50[0.0,0.5] | 8 | 23.0     |
| SlimCLR-MoCov2     | R-50[0.0,0.5,0.25] | 8 | 26.6     |
| SlimCLR-MoCov2     | R-50[0.0,0.75,0.5,0.25] | 8 | 33.4     |

Table 4: The pre-training time for 200 epochs. #V100 is the number of Tesla V100 GPUs. The slow start epoch $S = 100$. We adopt NVIDIA DALI (NVIDIA, 2021) to accelerate the data augmentation pipeline.

The results are presented in Table 4. The choices of temperatures make trivial differences. By contrast, SEED (Fang et al., 2021) uses a small temperature 0.01 for the teacher to get sharp distribution and a relatively large temperature 0.2 for the student. BINGO (Xu et al., 2022) adopts a single temperature 0.2. These choices are different from ours, but we observed that SlimCLR is more robust to the choice of temperature.

Table 3d. The choices of temperatures make trivial differences. By contrast, SEED (Fang et al., 2021) uses a small temperature 0.01 for the teacher to get sharp distribution and a relatively large temperature 0.2 for the student. BINGO (Xu et al., 2022) adopts a single temperature 0.2. These choices are different from ours, but we observed that SlimCLR is more robust to the choice of temperature.

loss reweighting We compared four loss reweighting
matters in Table 3e. They are

1. \( \lambda_i = 1.0 + I\{w_i = w_1\} \times \sum_{j=2}^{n} w_j \),
2. \( \lambda_i = 1.0 + I\{w_i = w_1\} \times \max\{w_2, \ldots, w_n\} \),
3. \( \lambda_i = n \times \frac{w_i}{\sum_{j=1}^{n} w_j} \),
4. \( \lambda_i = n \times \frac{1.0 + \sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} (1.0 + \sum_{k=j}^{n} w_k)} \),

where \( I\{\cdot\} \) equals to 1 if the inner condition is true, 0 otherwise. The corresponding widths of networks with widths \([1.0, 0.75, 0.5, 0.25]\) are \([2.5, 1.0, 1.0, 1.0]\), \([1.75, 1.0, 1.0, 1.0]\), \([1.6, 1.2, 0.8, 0.4]\), and \([1.54, 1.08, 0.77, 0.62]\). It is clear that a larger weight for the full model helps the system achieve better performance. This demonstrates again that it is important for the full model to lead the training process. The differences between the above four loss reweighting strategies are mainly reflected in the sub-networks with small sizes. To ensure the performance of the smallest network, we adopt the reweighting manner (1) in practice.

Self-supervised and supervised slimmable networks Training slimmable networks is generally more challenging in self-supervised learning than in supervised learning. In Figure 3 and Figure 4, gradient imbalance and gradient direction divergence are more pronounced in self-supervised cases, leading to more severe performance degradation in Figure 1. To gain a deeper understanding of the difficulty in training slimmable networks in self-supervised learning, we visualize the error surface and optimization trajectory (Li et al., 2018) of slimmable networks during training. Specifically, we train slimmable networks in both supervised and self-supervised (MoCo He et al. (2020)) manners on CIFAR-10 (Krizhevsky and Hinton, 2009) for 100 epochs, with a ResNet-20×4 base network containing 4.3M parameters. In self-supervised cases, we use a k-NN predictor (Wu et al., 2018) to obtain the accuracy. After training, we visualize the error surface and optimization trajectory in Figure 5 following Li et al. (2018).

The visualizations reveal that self-supervised learning is more challenging than supervised learning. In the left error surface of Figure 5a and Figure 5c, it can be observed that the terrain surrounding the valley is relatively flat in supervised cases, while it is more complex in self-supervised cases. Moreover, from the trajectory in the left of Figure 5b and Figure 5d, the contours in supervised cases are denser, indicating that the model in self-supervised cases requires more time to achieve the same accuracy improvement compared to the model in supervised cases.

The visualization indicates that weight-sharing networks have a greater impact in self-supervised cases. Firstly, weight-sharing networks induce significant changes to the error surface in Figure 5c, while the change is less apparent in supervised cases. Secondly, in self-supervised cases, interference between weight-sharing networks causes the model to deviate further from the global minima (i.e., the origin in the visualization) as illustrated in Figure 5d. In Figure 5b, the maximal offsets from the global minima along the 2nd PCA component are 21.75 and 28.49 for ResNet-20×4 and ResNet-20×4[1,0,0,5], respectively. This represents a 31.0% increase in offset. Conversely, for self-supervised
cases in Figure 5d, the maximal offsets from the global minima along the 2nd PCA component are 13.26 and 18.75 for ResNet-20×4 and ResNet- 20×4[1,0,0,5], respectively. This increase in offset is 41.4%. Therefore, it is evident that weight-sharing networks have a more pronounced impact in self-supervised cases.

By combining Figure 3 (gradient magnitude imbalance), Figure 4 (gradient direction divergence), and Figure 5 (optimization visualization), we posit a causal relationship between gradient magnitude imbalance, gradient direction divergence, and the challenge in training self-supervised slimmable networks. The gradient magnitude imbalance and gradient direction divergence are evidences of interference between weight-sharing networks, making the training of slimmable networks more difficult than that of a normal network. In the supervised case, consistence global ground truth alleviates the interference (Figure 3b&4a&5a&5b). However, in the self-supervise case, the lack of consistent global ground truth makes the optimization of self-supervised slimmable networks harder than that of supervised slimmable networks (Figure 3c&4b&5c&5d).

During the pre-training stage of SlimCLR, we introduce online distillation, slow start and loss reweighting to mitigate the gradient magnitude imbalance and gradient direction divergence (Figure 3f&4c). Additionally, to meet the conditions of inputs (Appendix A), we propose the switchable linear probe layer for improved linear evaluation of a self-supervised slimmable network.

5 Conclusion

In this work, we adapt slimmable networks for contrastive learning to obtain pre-trained small models in a self-supervised manner. By using slimmable networks, we can pre-train once and obtain several models of varying sizes, making them suitable for use across different devices. Additionally, our approach does not require the additional training process of large teacher models, as seen in previous distillation-based methods. However, weight-sharing networks in a slimmable network cause interference during self-supervised learning, resulting in gradient magnitude imbalance and gradient direction divergence. We develop several techniques to relieve the interference among networks during pre-training and linear evaluation. Two specific algorithms are instantiated in this work, i.e., SlimCLR-MoCoV2 and SlimCLR-MoCoV3. We take extensive experiments on ImageNet and achieve better performance than previous arts with fewer network parameters and FLOPs.

Conflict of interest. The authors declare that they have no conflict of interest.

Data availability. The datasets analysed during the current study are available in https://www.image-net.org/, https://www.cs.toronto.edu/~kriz/cifar.html, and https://cocodataset.org/. No new datasets were generated.

Acknowledgements. This work is supported by Major program of the National Natural Science Foundation of China (T2293720/T2293723). This work is also partially supported by Fundamental Research Funds for the Central Universities (Grant Number: 226-2022-00051).

Appendix

A Conditions of inputs

We consider the conditions of inputs when only using one slimmable linear layer during evaluation, i.e., consider solving multiple multi-class linear regression problems with shared weights. The parameters of the linear layer are $\theta \in \mathbb{R}^{d \times C}$, $C$ is the number of classes, where $\theta = [\theta_{11}, \theta_{12}]$, $\theta_{11} \in \mathbb{R}^{d_1 \times C}$, $\theta_{21} \in \mathbb{R}^{d_2 \times C}$, $d_1 + d_2 = d$. The first input for the full model is $X \in \mathbb{R}^{N \times d}$, where $N$ is the number of samples, $X = [X_{11}, X_{12}]$, $X_{11} \in \mathbb{R}^{N \times d_1}$, $X_{12} \in \mathbb{R}^{N \times d_2}$. The second input $X_1 \in \mathbb{R}^{N \times d_1}$ is the input feature for the sub-model parameterized by $\theta_{11}$. Generally, we have $N \geq d > d_1$. We assume that both $X$ and $X_1$ have independent columns, i.e., $X^T X$ and $X_1^T X_1$ are invertible. The ground truth is $T \in \mathbb{R}^{N \times C}$. The prediction of the full model is $Y = X \theta$, to minimize the sum-of-least-squares loss between prediction and ground truth, we get

$$\theta = \arg \min_{\theta} \|X \theta - T\|_2^2.$$  

(8)

By setting the derivative w.r.t. $\theta$ to 0, we get

$$\theta = (X^T X)^{-1} X^T T.$$  

(9)

In the same way, we can get

$$\theta_{11} = (X_{11}^T X_{11})^{-1} X_{11}^T T.$$  

(10)

For $X^T X$, we have

$$X^T X = \begin{bmatrix} X_{11}^T X_{11} & X_{11}^T X_{12} \\ X_{12}^T X_{11} & X_{12}^T X_{12} \end{bmatrix}.$$  

(11)

We denote the inverse of $X^T X$ as $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$, where $B_{12} = B_{21}^T$ as $X^T X$ is a symmetric matrix. For $X^T X B = I$, we have

$$X^T X \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} I_{d_1} & 0_{d_1 \times d_2} \\ 0_{d_2 \times d_1} & I_{d_2} \end{bmatrix}.$$  

(12)

Then we can get

$$X_{11}^T X_{11} + X_{11}^T X_{12} B_{21} = I_{d_1},$$  

(13)

$$X_{11}^T X_{11} B_{12} + X_{11}^T X_{12} B_{21} = 0_{d_1 \times d_2},$$  

(14)

$$X_{11}^T X_{12} B_{11} + X_{12}^T X_{12} B_{21} = 0_{d_1 \times d_2},$$  

(15)

$$X_{12}^T X_{11} B_{12} + X_{12}^T X_{12} B_{22} = I_{d_2}.$$  

(16)
At the same time
\[ \theta = (X^T X)^{-1} X^T T = BX^T T \]
\begin{align*}
B_{11} & = B_{12} = \frac{1}{2} [X_{11} X_{12}]^T T \\
B_{21} & = B_{22} = \frac{1}{2} [X_{21} X_{22}]^T T \\
\text{and} & \\
\theta_{11} & = (B_{11} X_{11}^T + B_{12} X_{12}^T) T \\
& = (X_{11}^T X_{11})^{-1} X_{11}^T T. \tag{17}
\end{align*}

From Eq. (15), we get
\begin{align*}
B_{11} &= - (X_{12} X_{12})^{-1} X_{12} X_{11} B_{11}, \tag{18} \\
B_{12} &= -B_{11} X_{11} X_{12} (X_{12} X_{12})^{-1}. \tag{19} \\
\text{Substitute Eq. (19) into Eq. (13), we get} & \\
B_{11} &= (X_{11}^T X_{11} - X_{11}^T X_{12} (X_{12} X_{12})^{-1} X_{12}^T X_{11})^{-1}. \tag{20} \\
\text{At the same time} & \\
\theta_{11} &= (B_{11} X_{11}^T + B_{12} X_{12}^T) T \\
& = B_{11} (X_{11}^T - X_{11}^T X_{12} (X_{12} X_{12})^{-1} X_{12}^T X_{11}) T. \tag{21} \\
\text{Combining Eq. (18) and Eq. (22), we get the condition of the input} & \\
B_{11} (X_{11}^T - X_{11}^T X_{12} (X_{12} X_{12})^{-1} X_{12}^T X_{11}) T = (X_{11}^T X_{11})^{-1} X_{11}^T T. \tag{23}
\end{align*}

In order to verify whether the input condition in Eq. (23) is met in practice, we randomly sampled 2048 images from the training set of ImageNet and used a ResNet-50 [1, 0.0, 0.75, 0.5, 0.25] pre-trained by SimCLR-Mocov2 (800 epochs) to extract their features. The features extracted by ResNet-50 are denoted as \( X \in \mathbb{R}^{2048 \times 1024} \), and the features extracted by ResNet-50a are denoted as \( X_1 \in \mathbb{R}^{2048 \times 512} \). We also conducted similar experiments on the validation set of ImageNet and found that the average value of entries in \([L - R]\) is 1.07, which indicates a total difference of 1099665.50. We also conducted similar experiments on the validation set of ImageNet and found that the average value of entries in \([L - R]\) is 0.88, which indicates a total difference of 903094.9.

These results demonstrate that the features of a slimmable network learned by contrastive self-supervised learning cannot meet the input conditions in Eq. (23) when using a single slimmable linear probe layer. This provides an explanation for why using a switchable linear probe layer achieves much better performance in Table 3a.

References

Bachman P, Hjelm RD, Buchwalter W (2019) Learning representations by maximizing mutual information across views. In: NeurIPS

Ballard DH (1987) Modular learning in neural networks. In: AAAI

Cai H, Gan C, Wang T, Zhang Z, Han S (2020) Once-for-all: Train one network and specialize it for efficient deployment. In: ICLR

Caron M, Bojanowski P, Joulin A, Douze M (2018) Deep clustering for unsupervised learning of visual features. In: ECCV

Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, Joulin A (2020) Unsupervised learning of visual features by contrasting cluster assignments. Advances in neural information processing systems 33:9912–9924

Chavan A, Shen Z, Liu Z, Liu Z, Cheng KT, Xing EP (2022) Vision transformer slimming: Multi-dimension searching in continuous optimization space. In: CVPR

Chen K, Wang Y, Pang J, Cao Y, Xiong Y, Li X, Sun S, Feng W, Liu Z, Xu J, Zhang Z, Cheng D, Zhu C, Cheng T, Zhao Q, Li B, Lu X, Zhu R, Wu Y, Dai J, Wang J, Shi J, Ouyang W, Loy CC, Lin D (2019) MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:190607155

Chen T, Kornblith S, Norouzi M, Hinton GE (2020a) A simple framework for contrastive learning of visual representations. In: ICML

Chen T, Kornblith S, Swersky K, Norouzi M, Hinton GE (2020b) Big self-supervised models are strong semi-supervised learners. In: NeurIPS

Chen X, He K (2021) Exploring simple siamese representation learning. In: CVPR

Chen X, Fan H, Girshick RB, He K (2020c) Improved baselines with momentum contrastive learning. CoRR abs/2003.04297, 2003.04297

Chen X, Xie S, He K (2021) An empirical study of training self-supervised vision transformers. In: ICCV

Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT

Dosovitskiy A, Fischer P, Springenberg JT, Riedmiller MA, Brox T (2016) Discriminative unsupervised feature learning with exemplar convolutional neural networks. TPAMI

Fang Z, Wang J, Wang L, Zhang L, Yang Y, Liu Z (2021) SEED: self-supervised distillation for visual representation. In: ICLR

Gao Y, Zhuang J, Lin S, Cheng H, Sun X, Li K, Shen C (2022) Disco: Remedy self-supervised learning on lightweight models with distilled contrastive learning. In: ECCV

Goyal P, Dollâr P, Girshick RB, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K (2017) Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR abs/1706.02677, 1706.02677

Gu J, Liu W, Tian Y (2021) Simple distillation baselines for improving small self-supervised models. arXiv preprint arXiv:210611304

Guo J (2022) Reducing the teacher-student gap via adaptive temperatures

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: CVPR

He K, Gkioxari G, Dollâr P, Girshick RB (2017) Mask R-CNN. In: ICCV, pp 2980–2988

He K, Fan H, Wu Y, Xie S, Girshick RB (2020) Momentum contrast for unsupervised visual representation learning. In: CVPR

He K, Chen X, Xie S, Li Y, Dollâr P, Girshick R (2022) Masked autoencoders are scalable vision learners. In: CVPR
He T, Zhang Z, Zhang H, Zhang Z, Xie J, Li M (2019) Bag of tricks for image classification with convolutional neural networks. In: CVPR

Hinton GE, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. CoRR abs/1503.02531

Hjelm RD, Fedorov A, Lavioie-Marchildon S, Grewal K, Bachman P, Trischler A, Bengio Y (2019) Learning deep representations by mutual information estimation and maximization. In: ICLR

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML

Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: ICLR

Koohpayegani SA, Tejankar A, Pirsiavash H (2020) Contrastive feature learning via teacher assistant. In: AAAI

Ghasemzadeh H (2020) Improved knowledge distillation. In: CVPR

García D, Ginsburg B, Houston M, Kuchaiev O, Venkatesh S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Li F (2015) Imagenet large scale visual recognition challenge. IJCV

Shi H, Zhang Y, Yang T, Zhou L, Li Y, Guo Y, Zhuang Y (2022) On the efficacy of small self-supervised contrastive models without distillation signals. In: AAAI, vol 36, pp 2225-2234

Tan M, Le Q (2019) EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICLR, PMLR, pp 6105–6114

Tian Y, Krishnan D, Isola P (2020) Contrastive representation distillation. In: ICLR

Tomasev N, Bica I, McWilliams B, Buesing L, Pascanu R, Blundell C, Mitrovic J (2022) Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? CoRR abs/2201.05119

Wang W, Chen M, Zhao S, Chen L, Hu J, Liu H, Cai D, He X, Liu W (2021) Accelerate cnns from three dimensions: A comprehensive pruning framework. In: CVPR

Xu H, Fang J, Zhang X, Xie L, Wang X, Dai W, Xiong H, Tian Q (2022) Bag of instances aggregation boosts self-supervised distillation. In: ICLR

Yang T, Zhu S, Chen C, Yan S, Zhang M, Willis AR (2020) Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In: ECCV

You Y, Gitman I, Ginsburg B (2017) Large batch training of convolutional networks. arXiv preprint arXiv:170803888

Yu J, Huang T (2019a) Autoslim: Towards one-shot architecture search for channel numbers. arXiv preprint arXiv:190311728

Yu J, Huang TS (2019b) Universally slimmable networks and improved training techniques. In: ICCV

Yu J, Yang L, Xu N, Yang J, Huang TS (2019) Slimmable neural networks. In: ICLR

Yu J, Jin P, Liu H, Bender G, Kindermans P, Tan M, Huang TS, Song X, Pang R, Le Q (2020) Bignas: Scaling up neural architecture search with big single-stage models. In: ECCV, Springer, pp 702–717

Zhang R, Isola P, Efros AA (2016) Colorful image colorization. In: ECCV

Zhang Y, Xiang T, Hospedales TM, Lu H (2018) Deep mutual learning. In: CVPR, DOI 10.1109/CVPR.2018.00454

Zhao B, Cui Q, Song R, Qiu Y, Liang J (2022a) Decoupled knowledge distillation. In: CVPR

Zhao S, Zhou L, Wang W, Cai D, Lam TL, Xu Y (2022b) Towards better accuracy-efficiency trade-offs: Divide and co-training. TIP 31:5869–5880

versarial networks. In: ICLR

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Li F (2015) Imagenet large scale visual recognition challenge. IJCV