On Improving the Algorithm-, Model-, and Data- Efficiency of Self-Supervised Learning

Yun-Hao Cao, Jianxin Wu*
National Key Laboratory for Novel Software Technology, Nanjing University, China
caoyh@lamda.nju.edu.cn, wujx2001@nju.edu.cn

Abstract

Self-supervised learning (SSL) has developed rapidly in recent years. However, most of the mainstream methods are computationally expensive and rely on two (or more) augmentations for each image to construct positive pairs. Moreover, they mainly focus on large models and large-scale datasets, which lack flexibility and feasibility in many practical applications. In this paper, we propose an efficient single-branch SSL method based on non-parametric instance discrimination, aiming to improve the algorithm, model, and data efficiency of SSL. By analyzing the gradient formula, we correct the update rule of the memory bank with improved performance. We further propose a novel self-distillation loss that minimizes the KL divergence between the probability distribution and its square root version. We show that this alleviates the infrequent updating problem in instance discrimination and greatly accelerates convergence. We systematically compare the training overhead and performance of different methods in different scales of data, and under different backbones. Experimental results show that our method outperforms various baselines with significantly less overhead, and is especially effective for limited amounts of data and small models.

1. Introduction

Deep supervised learning has achieved great success in the last decade. However, traditional supervised learning approaches rely heavily on a large set of annotated training data. Self-supervised learning (SSL) has gained popularity because of its ability to avoid the cost of annotating large-scale datasets as well as the ability to obtain task-agnostic representations. After the emergence of the contrastive learning (CL) paradigm [42, 9], SSL has clearly gained momentum and several recent works [10, 21, 7] have achieved comparable or even better accuracy than the supervised pertaining when transferring to downstream tasks.

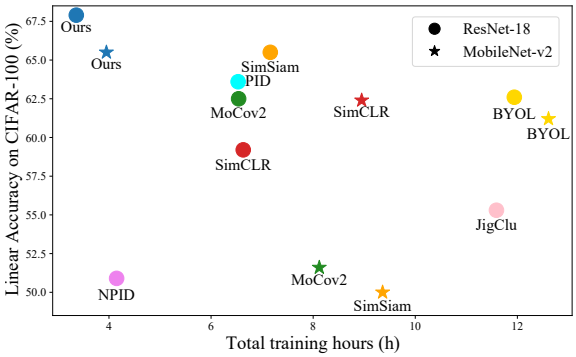


Figure 1: Linear probing accuracy and training cost (in hours) of different SSL methods on CIFAR-100 [30].

However, these methods are almost all dual-branched, that is, the network needs to generate at least two views for each image during learning. What's worse, the combination of a time-consuming algorithm (dual-branched), a large-scale dataset (e.g., ImageNet), a complex backbone (e.g., ResNet-50), and a large number of epochs (800 or more) means that SSL methods are computationally extremely expensive. This phenomenon makes SSL a privilege for researchers at few institutions. In this paper, we propose to improve the efficiency of SSL methods from three aspects: algorithm (training) efficiency, model efficiency, and data efficiency.

As an alternative to dual-branch SSL, single-branch methods [6, 20, 36] only require a single crop for each image in each iteration, which naturally reduces the training overhead per iteration. As a representative of them, parametric instance discrimination methods [16, 34, 5] learn to classify every example into its own category. However, the final parametrized classification layer will bring an intolerable increase in computation and GPU memory usage as the number of training data increases. As a solution, NPID [44] transforms instance discrimination into a non-parametric version by maintaining a memory bank but its accuracy is far behind mainstream contrastive learning methods. MoCo [23] improves NPID using a momentum

*J. Wu is the corresponding author.

encoder at the cost of turning to dual-branch again. Chen *et al*. [8] proposed a jigsaw clustering task to improve single-branch SSL but the complicated pipeline makes its training overhead even larger than many dual-branch methods. Therefore, how to design an efficient and effective single-branch self-supervised method is challenging.

In this paper, we aim to bridge the accuracy gap between single- and dual-branch methods while maintaining the training efficiency of single-branch methods. Our method is based on NPID [44], but with the following three important improvements. First, we perform a forward pass on the untrained network to obtain features as the initialization of the memory bank, which was randomly initialized in both NPID and MoCo. Inspired by [21, 34], we know that a randomly initialized network also has representation ability, and experiments show that our initialization can speed up the convergence with negligible cost. Second, we revise the update rule of the memory bank based on gradient formulation. In [44], the feature of the i-th instance will only be used to update the weights of the i-th class. By analyzing the weights' gradient, we know that the feature of an instance will also be passed back to update the weights corresponding to other instances using our update rule. Third, we design an effective self-distillation loss that minimizes the KL divergence of the probability distribution and the distribution after taking the square root. Theoretical and empirical results demonstrate that this loss can effectively solve the problem of infrequent updating [5] in instance discrimination and greatly accelerate convergence, achieving better performance with less overhead, as shown in Fig. 1.

In addition to improving algorithm efficiency, we also try to improve the model and data efficiency in self-supervised learning. In practical applications, many models need to be deployed on terminal devices with limited memory, computation, and storage capabilities. Hence, self-supervised learning with small models is an important problem. Fang *et al*. [18] found that small models perform poorly under the paradigm of self-supervised contrastive learning and smaller models with fewer parameters cannot effectively learn instance-level discriminative representation with a large amount of data. SEED [18] and DisCo [19] adopt knowledge distillation to address this problem and Shi *et al*. [40] tweaked hyperparameters and image augmentations to improve performance on small models. In this paper, we show that our method can effectively improve the performance of small models and speed up the convergence of instance discrimination tasks for them.

From the perspective of data efficiency, many realistic scenarios require that we cannot always rely on large-scale training data. For example, it is difficult to collect large-scale training data in some fields (e.g., medical images). Also, fast model iteration (e.g., update a model in 10 minutes) forbids us from using large-scale data for training. Therefore, in this paper, we study the performance of different SSL methods under different scales of training data. Experimental results demonstrate the data efficiency of our method, and our improvements will increase as the amount of data decreases. In summary, our contributions are:

- We propose a single-branch method, which improves the training efficiency, model efficiency, and data efficiency of self-supervised learning.
- We propose the initialization method of the memory bank, and revise the update rule based on the gradient formula.
- We propose a self-distillation KL loss to alleviate the infrequent updating problem for instance discrimination, which greatly accelerates the convergence.
- We systematically compare the efficiency of different SSL methods, and exhaustive experiments show that our method achieves better performance on various benchmarks with less training overhead. Moreover, our method is extremely effective for lightweight models and small data, and our advantages will be further amplified as the amount of data decreases.

2. Related Works

Self-supervised learning (SSL) has emerged as a powerful method to learn visual representations without labels. Many recent works follow the contrastive learning paradigm [42]. For instance, SimCLR [9] and MoCo [23] train networks to identify a pair of views originating from the same image when contrasted with many views from other images. Follow-up works BYOL [21] and SimSiam [11] discard negative sampling in contrastive learning but achieve even better results using siamese networks. Unlike the siamese structure in contrastive methods, single-branch methods [20, 36, 6, 16] propose different pretext tasks to train unsupervised models. Pretext-based approaches mainly explore the context features of images or videos such as context similarity [36, 14], spatial structure [20], clustering property [6], temporal structure [31], etc. Parametric instance discrimination [16, 2, 34] learns to discriminate between a set of surrogate classes, where each class represents different transformed patches of a single image. NPID [44] employs non-parametric instance discrimination by maintaining a memory bank but its performance is far behind the mainstream contrastive learning methods. JigClu [8] improves the performance of single-branch methods at the cost of greater training overhead.

There are also some recent works trying to improve the efficiency of SSL in different dimensions. SEED [18] and DisCo [19] study self-supervised learning with small models. SSQL [3] proposes to pretrain quantization-friendly self-supervised models to facilitate downstream deployment. Cao *et al*. [4] and Cole *et al*. [13] investigated the data efficiency of self-supervised methods. Fast-MoCo [12]

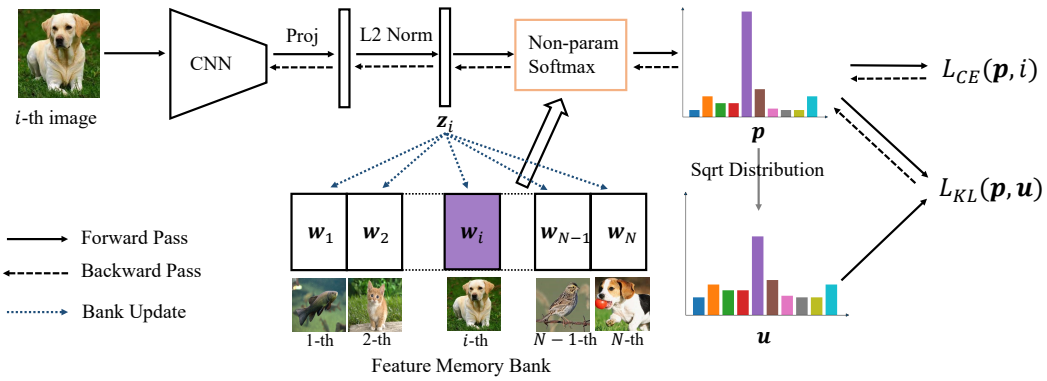


Figure 2: The general framework of our method.

tries to accelerate the training of MoCov2 [10], which is still a dual-branch method. These previous methods try to improve the SSL efficiency from a single dimension, but we study the efficiency of SSL from three dimensions for the first time in this work.

3. The Proposed Method

We begin with the basic notation and a brief introduction of our framework, followed by our algorithm and analysis.

3.1. Preliminaries

An input image $\boldsymbol{x}_i$ $(i = 1, \cdots, N)$ is sent to a network $f(\cdot)$ and get output representation $\boldsymbol{z}_i = f(\boldsymbol{x}_i) \in \mathbb{R}^d$, where N denotes the total number of instances. Then, a fully connected (FC) layer $\boldsymbol{w}$ is used for classification and the number of classes equals the total number of training images N for parametric instance discrimination. Let us denote the FC's weights as $\boldsymbol{w}_i$ $(i = 1, 2, \ldots, N)$, then the prediction for the i-th instance is

$$p_i = \frac{\exp(\boldsymbol{w}_i^T \boldsymbol{z}_i)}{\sum_{j=1}^{N} \exp(\boldsymbol{w}_j^T \boldsymbol{z}_i)} . \tag{1}$$

The loss function for the i-th instance is

$$L_{\text{CE}} = -\log(p_i) , \tag{2}$$

because every instance is a class and the label for $\boldsymbol{x}_i$ is i.

As shown in Fig. 2, we use a non-parametric variant following [44], where each $\boldsymbol{w}_i$ is stored in a feature memory bank without using gradient back-propagation. This eliminates the need for computing and storing the gradients for $\boldsymbol{w}_i$, improving the storage and training efficiency.

3.2. Feature Bank

Now we describe how we initialize and update the feature memory bank.

Feature Calibrate. NPID [44] and MoCo [23] randomly initialize the memory bank while we perform a forward pass on the untrained network to obtain features for initialization, i.e, $\boldsymbol{w}_i = \boldsymbol{z}_i^{(0)}$. This brings negligible overhead, but as we show later in Sec. 4.4, speeds up convergence and improves performance.

Grad Update. A naive way to update the weights in the feature bank is to use the current output feature [44]:

$$\boldsymbol{w}_i \leftarrow m\boldsymbol{w}_i + (1-m)\boldsymbol{z}_i^{(t)} , \tag{3}$$

where $\boldsymbol{z}_i^{(t)}$ is the output representation for the i-th instance at the t-th iteration and m is a hyper-parameter.

However, if we calculate the gradient w.r.t $\boldsymbol{w}_k$:

$$\frac{\partial L_{\text{CE}}}{\partial \boldsymbol{w}_k} = -\delta_{\{k=i\}}\boldsymbol{z}_i + \frac{e^{\boldsymbol{w}_k^T \boldsymbol{z}_i}}{\sum_{j=1}^{N} e^{\boldsymbol{w}_j^T \boldsymbol{z}_i}}\boldsymbol{z}_i = (p_k - \delta_{\{k=i\}})\boldsymbol{z}_i , \tag{4}$$

where δ is an indicator function, equals 1 iff $k = i$.

According to (4), when we sum the loss from all instances, the update direction (i.e., negative gradient) for $\boldsymbol{w}_i$ will be affected by the output of other instances. Specifically, the corrected update direction will be:

$$\hat{\boldsymbol{z}}_i^{(t)} = (1-p_i)\boldsymbol{z}_i^{(t)} - \sum_{j \neq i} p_j \boldsymbol{z}_j^{(t)} . \tag{5}$$

Then we use this corrected direction to update the bank:

$$\boldsymbol{w}_i \leftarrow m\boldsymbol{w}_i + (1-m)\hat{\boldsymbol{z}}_i^{(t)} . \tag{6}$$

3.3. SqrtKL

When we do instance discrimination, one important issue is that the updates to FCs are very rare: the gradient with respect to $\boldsymbol{w}_j$ $(j \neq i)$ has to be calculated from $-\frac{1}{p_i}$, which is *mostly* related to p_i. Now if we back propagate from p_i to $\boldsymbol{w}$, it is *mostly* focused only on updating $\boldsymbol{w}_i$, but *not* other FC weights $\boldsymbol{w}_j$ $(j \neq i)$. Although the $\sum_{j=1}^{C} \exp(\boldsymbol{w}_j^T \boldsymbol{z}_i)$ term involves $\boldsymbol{w}_j$ for $j \neq i$, its impact is negligible in most cases. To be more precise, from (4) we know that when

$j \neq i$, then the gradient with respect to $\boldsymbol{w}_j$ is $p_j \boldsymbol{z}_i$ — clearly negligible when $p_j \approx 0$. Or, $\boldsymbol{w}_i$ is updated roughly only once per epoch, thus we need many epochs to converge.

Now we define a square root probability distribution

$$u_i = \frac{\sqrt{p_i}}{\sum_{j=1}^{N} \sqrt{p_j}}$$

for $i = 1, 2, \ldots, N$. $\mathbf{u} = \{u_1, \ldots, u_N\}$ will be clearly more balanced than $\mathbf{p} = \{p_1, \ldots, p_N\}$, as shown in Fig. 2. In addition to the cross entropy loss, we can add a KL divergence loss:

$$L_{\text{SqrtKL}} = \text{KL}(\mathbf{p}, \mathbf{u}) \,. \tag{7}$$

Because $\mathbf{u}$ is generated out of $\mathbf{p}$, one network is enough and it is a self-distillation. Note that "more balanced" means even though the prediction $\mathbf{p}$ is very sharp (hence $p_j \approx 0$ if $j \neq i$), $\mathbf{u}$ will be less sharp. One example: let $N = 10$, $\mathbf{p} = \{0.91, 0.01, \ldots, 0.01\}$. Then $\mathbf{u} = \{0.5145, 0.0539, \ldots, 0.0539\}$ is much flatter and hence more $\boldsymbol{w}_j$ for $j \neq i$ will be updated in every epoch.

3.3.1 Alleviate the Infrequent Updating Problem

We only consider the gradient of $\text{KL}(\mathbf{p}, \mathbf{u})$ with respect to $\boldsymbol{w}_j$ ($j \neq i$). Note that $\mathbf{u}$ is not involved in gradient computation (in knowledge distillation [27] the teacher predictions are not involved in gradient computation, either). Now we can get (see appendix for derivations)

$$\frac{\partial \text{KL}(\mathbf{p}, \mathbf{u})}{\partial p_k} = 0.5 \log p_k + (1 + \log c) \,, \tag{8}$$

where we define $c = \sum_s \sqrt{p_s}$. Using the above example where $p_j = 0.01$, from (4) we can get:

$$\left\| \frac{\partial L_{\text{CE}}}{\partial \boldsymbol{w}_j} \right\|_2 = p_j \left\| \boldsymbol{z}_i \right\|_2 = 0.01 \left\| \boldsymbol{z}_i \right\|_2 \,. \tag{9}$$

For L_{SqrtKL}, we can also calculate the gradient w.r.t. $\boldsymbol{w}_j$ (see appendix for detailed derivations):

$$\left\| \frac{\partial L_{\text{SqrtKL}}}{\partial \boldsymbol{w}_j} \right\|_2 \approx 0.021 \left\| \boldsymbol{z}_i \right\|_2 \,, \tag{10}$$

where the update range of $\boldsymbol{w}_j$ is doubled, hence mitigating the infrequent updating problem and it will be further alleviated by increasing the coefficient λ introduced later.

Note that NPID [44] uses proximal optimization to accelerate convergence of instance discrimination:

$$L_p = \left\| \boldsymbol{z}_i - \boldsymbol{w}_i \right\|_2^2 . \tag{11}$$

However, we can find that $\frac{\partial L_p}{\partial \boldsymbol{w}_j} = 0$ for $j \neq i$, which means this loss does not solve the infrequent updating issue. The difference between L_{SqrtKL} and L_p in gradient calculation explains why our method is significantly better than NPID in the following experimental results.

3.3.2 From an Optimization Perspective

L_{SqrtKL} can be decomposed into two components:

$$L_{\text{SqrtKL}} = \underbrace{\sum_k p_k \log p_k}_{L_1} \underbrace{- \sum_k p_k \log u_k}_{L_2} \tag{12}$$

To minimize L_1 amounts to maximize $-\sum_k p_k \log p_k$, or max entropy [29]. L_1 achieves its minimum when $p_k = \frac{1}{N}$ for all k. Obviously, L_2 achieves its minimum when

$$p_j = \begin{cases} 1 & j = \arg\max_k u_k \\ 0 & \text{otherwise} \end{cases} .$$

Note that L_2 is determined by the largest value in the distribution, hence minimizing the cross entropy loss will in effect minimize L_2, too.

While L_2 makes the distribution sharper, L_1 makes it flatter. In the appendix, we show that combining L_1 and L_2 gives the best results, and L_1 is more important in L_{SqrtKL}.

The overall loss function of our method is:

$$L = L_{\text{CE}} + \lambda L_{\text{SqrtKL}} \,, \tag{13}$$

where λ is a hyper-parameter.

4. Experimental Results

We introduce the implementation details in Sec. 4.1. We experiment on CIFAR-10 [30], CIFAR-100 [30], and Tiny-ImageNet in Sec. 4.2. We experiment on ImageNet [38] and study the transfer performance of ImageNet pretrained models on downstream recognition, object detection, and instance segmentation benchmarks in Sec. 4.3. Finally, we investigate the effects of different components and hyper-parameters in our method in Sec. 4.4. All our experiments were conducted using PyTorch with Tesla K80 and 3090 GPUs. Codes will be publicly available upon acceptance.

4.1. Implementation Details

Datasets. The main experiments are conducted on four benchmark datasets, i.e., CIFAR-10, CIFAR-100, Tiny-ImageNet and ImageNet. Tiny-ImageNet contains 100,000 training and 10,000 validation images from 200 classes at 64×64 resolution. We also conduct transfer experiments on 2 recognition benchmarks as well as 2 detection benchmarks Pascal VOC 07&12 [17] and COCO2017 [33].

Backbones. In addition to the commonly used ResNet-50 [25] in recent SSL papers, we also adopt 4 smaller networks to study model efficiency, i.e., ResNet-18 [25], MobileNetv2 [39], MobileNetv3 [28], and EfficientNet [41] for our experiments. Sometimes we abbreviate ResNet-18/50 to R-18/50, and MobileNetv3 to Mobv3.

Table 1: Linear evaluation results on three benchmark datasets. All pretrained for 400 epochs and we report the total pretraining cost (in hours) using 4 Tesla K80 cards on CIFAR-10 as an example.

Backbone	Method	Single Crop	Single Network	Training Cost (h)	GPU Memory (MB)	Accuracy (%)		
						CIFAR-10	CIFAR-100	Tiny-ImageNet
ResNet-18	BYOL [21]	×	×	11.94	2897	89.3	62.6	32.6
	JigClu [8]	✓	✓	11.59	2344	88.7	55.3	33.4
	SimSiam [11]	×	✓	7.16	2501	90.7	65.5	37.1
	SimCLR [9]	×	✓	6.63	2185	89.4	59.2	37.6
	MoCov2 [10]	×	×	6.54	1757	88.9	62.5	35.8
	PID [5]	✓	✓	6.53	3639	89.8	63.6	36.8
	NPID [44]	✓	✓	4.15	1879	80.8	50.9	27.3
	Ours	✓	✓	**3.36**	**1715**	**91.1**	**67.9**	**39.7**
MobileNetv2	BYOL [21]	×	×	12.61	4503	88.1	61.2	28.7
	SimSiam [11]	×	✓	9.36	4275	86.1	50.0	20.5
	SimCLR [9]	×	✓	8.95	4061	**88.9**	62.4	23.6
	MoCov2 [10]	×	×	8.12	2599	83.3	51.6	21.3
	Ours	✓	✓	**3.95**	**2181**	88.7	**65.5**	**36.2**
ResNet-50	BYOL [21]	×	×	31.08	9435	90.3	66.7	41.1
	SimSiam [11]	×	✓	22.32	9139	90.9	64.3	39.3
	SimCLR [9]	×	✓	21.94	8951	91.5	66.2	42.8
	MoCov2 [10]	×	×	14.72	5373	90.2	66.5	42.2
	Ours	✓	✓	**10.75**	**5095**	**92.0**	**71.6**	**44.9**

Training details. We use SGD for pretraining, with a batch size of 512 and a base lr=0.1. The learning rate has a cosine decay schedule. The weight decay is 0.0001 and the SGD momentum is 0.9. We set $m = 0.5$ and $\lambda = 20$ and we pretrain for 400 epochs on CIFAR-10, CIFAR-100, and Tiny-ImageNet, and 200 epochs on ImageNet by default.

4.2. Experiments on CIFAR and Tiny ImageNet

We first compare our method with 4 popular dual-branch SSL methods (BYOL [21], SimSiam [11], SimCLR [9], MoCov2 [10]) and 3 single-branch methods (PID [16], NPID [44], Jigclu [8]) on CIFAR-10, CIFAR-100 and Tiny-ImageNet using three CNN backbones in Table 1. All methods are pretrained for 400 epochs for fair comparisons and we report the total training hours on CIFAR-10 using 4 K80 GPUs. We also report the GPU memory usage of each method during training and here we use the same batch size 512 for fair comparisons. We report the linear probing accuracy on each dataset, following the practice in [3].

Comparison with Dual-Branch Methods. As shown in Table 1, our method only requires a single network branch and a single crop, thus achieving much lower memory usage and training time than mainstream dual-branch SSL methods. When compared with SimSiam [11], our method only needs 46.9% of the training time and 68.6% of the GPU memory usage, but achieves 0.4%, 2.4% and 2.6% higher accuracy on CIFAR-10, CIFAR-100 and Tiny-ImageNet under R-18, respectively. When compared with BYOL [21], our method achieves significantly higher accuracy, using only one-third of the training time and nearly half of the GPU memory usage. We can reach similar conclusions by comparing with other methods and backbones.

Note that current self-supervised methods such as MoCov2 [10] and SimSiam perform poorly on small architectures such as MobileNetv2, as mentioned in [18]. In contrast, our method can also achieve very good results together with small models, especially on CIFAR-100 and Tiny-ImageNet. We think the reason for this is that the capacity of the small model is not enough to learn difficult self-supervised tasks. In contrast, our single-branch classification method is simple to learn and our proposed method makes the model easier to converge.

Comparison with Single-Branch Methods. Although both our method and PID [16, 5] are single-branch ones, PID requires a parameterized classification layer, which brings additional training (gradient back-propagation) and storage overhead, and will inevitably deteriorate with more training data. In contrast, our method is non-parametric and the training time and storage are less affected by the amount of training data. At the same time, our corrected update rule and SqrtKL loss also enable us to achieve much better results on all three datasets than NPID [44] and PID, which are also based on instance discrimination. When compared with the state-of-the-art single-branch method JigClu [8], the training time of our method is reduced by 71% for ResNet-18 (from 11.59 to 3.36 hours), because we do not need complex patch-level augmentations.

In short, our method greatly improves the training efficiency of the SSL method, achieves the best results with the least training overhead, and has a greater improvement in small models. It can be seen that among all comparison methods, MoCov2 is the strongest opponent in the tradeoff between accuracy and efficiency, so the main comparison method in our subsequent experiments will be MoCov2.

Table 2: Downstream object detection performance on VOC 07&12 and linear evaluation accuracy on Tiny-ImageNet when pretrained on ImageNet subsets using R-18 and R-50. Improvements compared to MoCov2 are listed in parentheses.

Backbone	Pretraining				VOC 07&12			Tiny-ImageNet
	Method	#Images	Epochs	Cost (h)	AP_{50}	AP	AP_{75}	
ResNet-18	random init.	0	0	0	59.2	32.5	31.5	0.5
	MoCov2 [10]	10,000	200	0.43	61.8	34.3	33.4	9.7
	Ours			**0.28**	**67.1** (+5.3)	**38.5** (+4.2)	**37.8** (+4.4)	**19.4** (+9.7)
	MoCov2 [10]	10,000	800	1.72	65.0	37.2	37.0	13.7
	Ours			**1.12**	**68.5** (+3.5)	**39.8** (+2.6)	**39.8** (+2.8)	**20.5** (+6.8)
	MoCov2 [10]	100,000	200	4.33	70.6	41.6	42.7	23.6
	SimSiam [11]			4.47	71.1	42.5	44.3	24.3
	Ours			**2.81**	**71.8** (+1.2)	**43.1** (+1.5)	**44.7** (+2.0)	**29.5** (+5.9)
	MoCov2 [10]	100,000	800	17.32	72.7	43.6	45.3	27.4
	Ours			**11.24**	**73.4** (+0.7)	**44.8** (+1.2)	**47.0** (+1.7)	**32.4** (+5.0)
ResNet-50	random init.	0	0	0	63.0	36.7	36.9	0.5
	MoCov2 [10]	10,000	800	1.88	71.6	43.9	45.9	23.6
	Ours			**1.64**	**76.8** (+5.2)	**49.3** (+5.4)	**53.6** (+7.7)	**26.3** (+2.7)
	MoCov2 [10]	100,000	200	4.65	76.2	48.0	51.6	35.3
	SimSiam [11]			5.42	76.4	49.8	54.2	30.5
	Ours			**4.09**	**78.2** (+2.0)	**51.1** (+3.1)	**55.7** (+4.1)	**36.3** (+1.0)
	MoCov2 [10]	100,000	800	18.62	78.7	51.5	56.3	43.7
	Ours			**16.36**	**79.7** (+1.0)	**53.3** (+1.8)	**58.8** (+2.2)	**44.3** (+0.6)

Table 3: ImageNet (subsets) pretraining results on small architectures. All pretrained for 200 epochs and we report the linear evaluation accuracy (%) when transferring to CIFAR-100 and the pretraining hours using 8 3090 cards. †: Results from [40].

Backbone	# Images	10,000		100,000		1,281,167	
	Method	Linear (%) ↑	Cost (h) ↓	Linear (%) ↑	Cost (h) ↓	Linear (%) ↑	Cost (h) ↓
Mobv3-small (2.5M)	MoCov2	21.8	0.42	33.0	4.18	40.4†	53.55
	Ours	**34.0**	**0.34**	**39.9**	**3.43**	**44.3**	**43.94**
Mobv3-large (5.4M)	MoCov2	28.1	0.42	32.5	4.23	42.4†	54.19
	Ours	**31.5**	**0.38**	**36.1**	**3.79**	**50.1**	**48.56**
EfficientNet-b0 (5.3M)	MoCov2	26.0	0.43	34.8	4.31	43.2†	55.22
	Ours	**38.1**	**0.39**	**39.9**	**3.87**	**47.8**	**49.56**
ResNet-18 (11.7M)	MoCov2	39.9	0.43	51.7	4.33	54.0†	55.47
	Ours	**48.8**	**0.28**	**55.3**	**2.81**	**60.4**	**36.05**

4.3. ImageNet and Transferring Experiments

In this subsection, we first perform unsupervised pretraining on the large-scale ImageNet training set without using labels, then investigate the downstream object detection performance on COCO2017 [33] and Pascal VOC 07&12 [17]. The detector is Faster R-CNN [37] for Pascal VOC, and Mask R-CNN [24] for COCO, both with the C4 backbone [37], following [10, 11].

Data Efficiency. In order to study the data efficiency of different methods, we first compare the performance under different data volumes by sampling the original ImageNet to smaller subsets. We randomly sample (without using any image label) 10 thousand (10k) and 100 thousand (100k) images to construct IN-10k and IN-100k, respectively. We only change the amount of data here and other training settings remain the same as before.

We experiment with ResNet-18 and ResNet-50 on ImageNet subsets in Table 2 and transfer the pretrained weights to Pascal VOC 07&12 for object detection and to Tiny-ImageNet for linear evaluation. As Table 2 shows, our method achieves significant improvements on both downstream tasks. Take R-18 as an example, when both are trained for 200 epochs on IN-100k (100,000 images), our method is significantly better than the baseline counterpart MoCov2: up to +1.2 AP_{50}, +1.5 AP, +2.0 AP_{75} on VOC 07&12 and +5.9% accuracy on Tiny-ImageNet, with 35.1% reduction in training time. When the amount of training data is further reduced to 10,000, our advantages will be further expanded: up to +5.3 AP_{50} on VOC and +9.7% accuracy on Tiny-ImageNet. Note that the results of our method trained for 200 epochs on IN-10k even surpass the results of MoCov2 trained for 800 epochs on IN-100k for R-50. Moreover, when comparing the results of R-18 and R-50, we find that our method will have a greater relative improvement on the smaller model R-18, especially on the linear evaluation metric of Tiny-ImageNet. These results

Table 4: Transfer Learning. All unsupervised methods are based on 200-epoch pretraining in ImageNet. We use Faster R-CNN for VOC and Mask R-CNN for COCO under the C4-backbone. Bold entries are the best two results following the style of [11]. †: Results from [11].

Method	Single Branch	VOC 07 detection			VOC 07+12 detection			COCO detection			COCO instance seg.		
		AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP^{bb}_{50}	AP^{bb}	AP^{bb}_{75}	AP^{mask}_{50}	AP^{mask}	AP^{mask}_{75}
scratch†	-	35.9	16.8	13.0	60.2	33.8	33.1	44.0	26.4	27.8	46.9	29.3	30.8
ImageNet supervised†	✓	74.4	42.4	42.7	81.3	53.5	58.8	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR† [9]	×	75.9	46.8	50.1	81.8	55.5	61.4	57.7	37.9	40.9	54.6	33.3	35.3
MoCov2† [10]	×	**77.1**	**48.5**	**52.5**	**82.3**	**57.0**	**63.3**	**58.8**	**39.2**	**42.5**	**55.5**	**34.3**	**36.6**
BYOL† [21]	×	**77.1**	47.0	49.9	81.4	55.3	61.1	57.8	37.9	40.9	54.3	33.2	35.0
SwAV† [7]	×	75.5	46.5	49.6	81.5	55.4	61.4	57.6	37.6	40.3	54.2	33.1	35.1
SimSiam† [11]	×	75.5	47.0	50.2	**82.0**	56.4	**62.8**	57.5	37.9	40.9	54.2	33.2	35.2
Ours	✓	75.5	**47.5**	**51.4**	**82.0**	**56.5**	62.6	**58.1**	**38.4**	**41.3**	**54.8**	**33.6**	**35.9**

Table 5: Object detection and instance segmentation results using ResNet-18 C4. †: Results from [18]. 'T' and 'AP^{mk}' abbreviate for pretrained teacher and 'AP^{mask}', respectively.

Method	T	COCO detection			COCO instance seg.		
		AP^{bb}_{50}	AP^{bb}	AP^{bb}_{75}	AP^{mk}_{50}	AP^{mk}	AP^{mk}_{75}
MoCov2† [10]	×	53.9	35.0	37.7	51.1	31.0	33.1
SEED† [18]	R-50	54.2	**35.3**	37.8	51.1	31.1	33.2
SEED† [18]	R-101	**54.3**	**35.3**	**37.9**	**51.3**	31.3	33.4
Ours	×	54.2	35.2	**37.9**	51.2	**31.4**	**33.5**

demonstrate the training, model, and data efficiency of our method, which improves performance while reducing the training time, and has greater advantages for small data and small models (i.e., resource-constrained scenarios).

Model Efficiency. In order to further study the performance of small models, we conduct experiments with different small models on ImageNet subsets (including the entire ImageNet) in Table 3. We transfer the learned representations to CIFAR-100 and conduct linear probing for comparison. Our method achieves higher accuracy than MoCov2 consistently under different lightweight backbones using training images at different scales, with less training costs. Moreover, we can see that our method's advantages are more obvious when the amount of data is reduced. Take MobileNetv3-small as an example, the improvement of our method is 3.9% when trained on ImageNet, and it increases to 6.9% on IN-100k and 12.2% on IN-10k.

Then, we present ImageNet and transferring results and we use ResNet-18 and Resnet-50 as the backbone to compare with mainstream methods. We will discuss the results of linear evaluation on ImageNet later in Sec. 5 and here we present the results of transferring to detection.

In Table 4, we compare the learned representations of ResNet-50 on ImageNet by transferring them to other tasks, including VOC object detection and COCO object detection and instance segmentation. All methods are based on 200-epoch pretraining on ImageNet using the reproduction of SimSiam [11]. Table 4 shows that our method's representations are transferable beyond the ImageNet task and it is competitive among these leading methods. SimSiam [11] conjectures that the common siamese structure is a core factor for the general success of these methods while our method achieves comparable results without using a siamese network. In Table 5, we compare the learned representations of ResNet-18 on ImageNet by transferring them to detection and segmentation tasks. Our method achieves better results than MoCov2 and is even comparable to SEED [18] (which uses extra knowledge distillation).

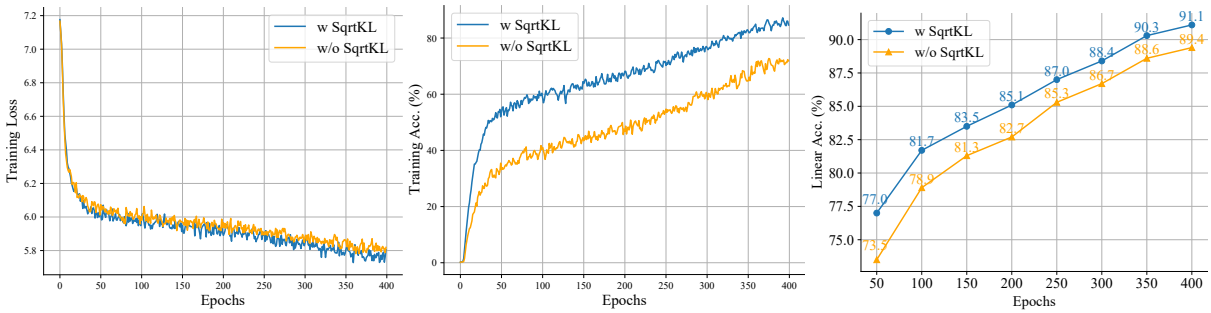
4.4. Ablation Study

Effect of Feature Calibrate. From Table 6 we can see that this initialization brings 1.1% gains on Tiny-ImageNet, with negligible cost (less than a minute).

Effect of SqrtKL. In Fig. 3 we plot the training curve of our method with and without using SqrtKL. As seen from Fig. 3b, SqrtKL can greatly speed up the convergence of the instance classification task and our method achieves much higher instance discrimination accuracy. Moreover, from the linear accuracy comparison of each epoch in Fig. 3c (also Table 6), we can see that our SqrtKL can also improve the representation ability of self-supervised models.

Effect of Grad Update. From Table 6 we can see that our corrected rule (6) brings 0.6% and 0.8% accuracy gains on Tiny-ImageNet without and with SqrtKL, respectively. Note that all our three strategies are beneficial and combining the three strategies achieves the best performance.

Effect of Hyper-parameter m. Now we study the effect of the hyper-parameter m, i.e., the momentum coefficient of (6). We train on CIFAR-10 for 400 epochs for all settings and the results are shown in Figure 4. We can observe that $m = 0.5$ and $m = 0.7$ achieve the highest accuracy. Notice that when $m = 0.0$, (6) is equivalent to directly updating with the results of the current iteration, that is, forgetting the previous results, so the effect is not good. It is worth mentioning that m in MoCov2 and BYOL is usually set to 0.99 or 0.999, which is larger than $m = 0.5$ in our paper. It is because they act on the model weights while we only act on the output features, and the update frequency of



(a) Training Loss (b) Training Accuracy (c) Linear Evaluation

Figure 3: Our method with vs. without SqrtKL on CIFAR-10.

Feature Calibrate	SqrtKL	Grad Update	CIFAR-10	Tiny-IN
×	×	×	88.8	35.8
✓	×	×	89.4	36.9
✓	×	✓	90.0	37.5
✓	✓	×	**91.1**	38.9
✓	✓	✓	**91.1**	**39.7**

Table 6: Ablation study under ResNet-18.

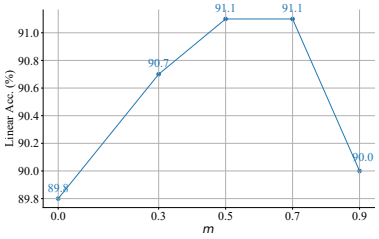


Figure 4: Effect of m on CIFAR-10 under ResNet-18.

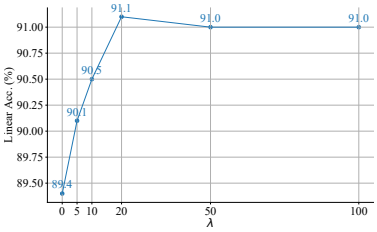


Figure 5: Effect of λ on CIFAR-10 under ResNet-18.

the model weights is much more frequent than features (the model weights will update in multiple iterations per epoch while each instance's representation only updates once).

Effect of Hyper-parameter λ. Now we study the effect of the hyper-parameter λ, i.e., the coefficient of L_{SqrtKL}. We train on CIFAR-10 for 400 epochs and the results are shown in Figure 5. We can observe that as λ grows, the accuracy steadily improves and will not continue to improve when it grows beyond 20. Notice that we directly set λ to 20 for all our experiments throughout this paper and did not tune it under different datasets or backbones. It also indicates that we can get better results with more carefully tuned λ.

5. Conclusions

In this paper, we proposed to improve the efficiency of self-supervised learning from three aspects: algorithm, model, and data. As a solution, we proposed an efficient single-branch method based on non-parametric instance discrimination, with enhanced update rule and self-distillation loss. Various experiments show that our method obtained a significant edge over baseline counterparts with much less training cost. Moreover, we achieved impressive results with limited amounts of training data and lightweight models, which demonstrates the model and data efficiency of our method. In the future, we will try to optimize the performance of our method on larger-scale datasets, which is a limitation of the current method.

Table 7: ImageNet linear evaluation accuracy (%) of different methods under ResNet-50.

Method	Single Branch	Accuracy (%)
Colorization [45]		39.6
JigPuz [36]		45.7
DeepCluster [6]		48.4
NPID [44]		54.0
BigBiGan [15]	✓	56.6
LA [46]		58.8
SeLa [1]		61.5
CPCv2 [26]		63.8
JigClu [8]		66.4
Ours		64.5
MoCo [23]		60.6
PIRL [35]		63.6
SimCLR [9]	×	64.3
PCL [32]		65.9
MoCov2 [10]		67.7

Despite performing well on detection, our metrics on ImageNet linear evaluation are not as good as the current mainstream dual-branch methods for ResNet-50, as shown in Table 7. This is partly because linear evaluation sometimes does not accurately measure the performance of SSL methods, as noted in [22]. More importantly, we conjecture the capacity of our method is not enough to model larger-scale data, such as ImageNet-21k. Therefore, in this paper, we mainly focused on the efficiency improvement, especially on small model and small data. Making our method suit large-scale data is an interesting future work.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, pages 1–13, 2020. 8

[2] Yue Cao, Zhenda Xie, Bin Liu, Yutong Lin, Zheng Zhang, and Han Hu. Parametric instance classification for unsupervised visual feature learning. *arXiv preprint arXiv:2006.14618*, 2020. 2

[3] Yun-Hao Cao, Peiqin Sun, Yechang Huang, Jianxin Wu, and Shuchang Zhou. Synergistic self-supervised and quantization learning. In *The European Conference on Computer Vision*, volume 13690 of *LNCS*, page 587–604. Springer, 2022. 2, 5

[4] Yun-Hao Cao and Jianxin Wu. Rethinking self-supervised learning: Small is beautiful. *arXiv preprint arXiv:2103.13559*, 2021. 2

[5] Yun-Hao Cao, Hao Yu, and Jianxin Wu. Training vision transformers with only 2040 images. In *The European Conference on Computer Vision*, volume 13685 of *LNCS*, pages 220–237. Springer, 2022. 1, 2, 5

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *The European Conference on Computer Vision*, volume 11218 of *LNCS*, pages 132–149. Springer, 2018. 1, 2, 8

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in neural information processing systems*, pages 9912–9924, 2020. 1, 7

[8] Pengguang Chen, Shu Liu, and Jiaya Jia. Jigsaw clustering for unsupervised visual representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 11526–11535, 2021. 2, 5, 8

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *The International Conference on Machine Learning*, pages 1597–1607, 2020. 1, 2, 5, 7, 8

[10] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 3, 5, 6, 7, 8

[11] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2, 5, 6, 7

[12] Yuanzheng Ci, Chen Lin, Lei Bai, and Wanli Ouyang. Fast-MoCo: Boost momentum-based contrastive learning with combinatorial patches. In *The European Conference on Computer Vision*, volume 13686 of *LNCS*, pages 290–306. Springer, 2022. 2

[13] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When does contrastive visual representation learning work? In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 14755–14764, 2022. 2

[14] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representations learning by context prediction. In *The IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 2

[15] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, page 10542–10552, 2019. 8

[16] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014. 1, 2, 5

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 4, 6

[18] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. SEED: Self-supervised distillation for visual representation. In *The International Conference on Learning Representations*, pages 1–12, 2021. 2, 5, 7

[19] Yuting Gao, Jia-Xin Zhuang, Shaohui Lin, Hao Cheng, Xing Sun, Ke Li, and Chunhua Shen. Disco: Remedying self-supervised learning on lightweight models with distilled contrastive learning. In *The European Conference on Computer Vision*, volume 13686 of *LNCS*, pages 237–253. Springer, 2022. 2

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *The International Conference on Learning Representations*, pages 1–14, 2015. 1, 2

[21] Jean-Bastien Grill, Florian Strub, Florent Altche, Corentin Tallec, Pierre H.Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammand Gheshlaghi Azar, Bial Piot, Koray Kavukcuoglu, Remi Munos, and Michal Valko. Boostrap your own latent: A new approach to self-supervised learning. In *Advances in neural information processing systems*, pages 21271–21284, 2020. 1, 2, 5, 7

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 8

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 3, 8

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *The IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 6

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4

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

[27] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 4

[28] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *The IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 4

[29] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957. 4

[30] Alex Krizhevsky and Geoffrey E. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1, 4

[31] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *The IEEE International Conference on Computer Vision*, pages 667–676, 2017. 2

[32] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, pages 1–12, 2021. 8

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *The European Conference on Computer Vision*, volume 8693 of *LNCS*, pages 740–755. Springer, 2014. 4, 6

[34] Yu Liu, Lianghua Huang, Pan Pan, Bin Wang, Yinghui Xu, and Rong Jin. Train a one-million-way instance classifier for unsupervised visual representation learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8706–8714, 2021. 1, 2

[35] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020. 8

[36] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *The European Conference on Computer Vision*, volume 9910 of *LNCS*, pages 69–84. Springer, 2016. 1, 2, 8

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 6

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 4

[40] Haizhou Shi, Youcai Zhang, Siliang Tang, Wenjie Zhu, Yaqian Li, Yandong Guo, and Yueting Zhuang. On the efficacy of small self-supervised contrastive models without distillation signals. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2):2225–2234, 2022. 2, 6, 13

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *The International Conference on Machine Learning*, pages 6105–6114, 2019. 4

[42] Aarin van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 2

[43] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 11, 12

[44] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 1, 2, 3, 4, 5, 8

[45] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *The European Conference on Computer Vision*, volume 9907 of *LNCS*, pages 649–666. Springer, 2016. 8

[46] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 6002–6012, 2019. 8

A. More Discussions about L_{SqrtKL}

A.1. Detailed Derivations

First, we present the derivation of (8) as below:

$$\frac{\partial \mathrm{KL}(\mathbf{p},\mathbf{u})}{\partial p_k} = \frac{\partial \sum_s p_s \log \frac{p_s}{u_s}}{\partial p_k} \tag{14}$$

$$= \frac{\partial \sum_s p_s \log p_s}{\partial p_k} - \frac{\partial \sum_s p_s \log u_s}{\partial p_k} \tag{15}$$

$$= 1 + \log p_k - \log u_k \tag{16}$$

$$= 1 + \log p_k - \log \sqrt{p_k} + \log \left(\sum_s \sqrt{p_s} \right) \tag{17}$$

$$= 0.5 \log p_k + (1 + \log c)\,, \tag{18}$$

where we define $c = \sum_s \sqrt{p_s}$ and use the fact that $\mathbf{u}$ is not involved in gradient computation from (15) to (16). It is obvious that $c \geq 1$ because $c^2 \geq \sum_k p_k = 1$. Equally obvious is that $c \leq \sqrt{N}$ — hence $1 \leq c \leq \sqrt{N}$.

Then, we denote $O_k = \frac{\partial \mathrm{KL}(\mathbf{p},\mathbf{u})}{\partial p_k}$ for simplicity and calculate the gradient of L_{SqrtKL} with respect to $\boldsymbol{w}_j$ $(j \neq i)$:

$$\frac{\partial \mathrm{KL}(\mathbf{p},\mathbf{u})}{\partial \boldsymbol{w}_j} = \sum_k \frac{\partial \mathrm{KL}(\mathbf{p},\mathbf{u})}{\partial p_k} \cdot \frac{\partial p_k}{\partial \boldsymbol{w}_j} \tag{19}$$

$$= \sum_{k \neq j} O_k \cdot \frac{\partial p_k}{\partial \boldsymbol{w}_j} + O_j \cdot \frac{\partial p_j}{\partial \boldsymbol{w}_j} \tag{20}$$

$$= \left(-\sum_{k \neq j} O_k p_k p_j + O_j (p_j - p_j^2) \right) \boldsymbol{z}_i\,, \tag{21}$$

where we use the equation below from (20) to (21)

$$\frac{\partial p_k}{\partial \boldsymbol{w}_j} = p_k (\delta_{\{k=j\}} - p_j) \boldsymbol{z}_i\,. \tag{22}$$

Then we continue to use the example in the paper, i.e., $N = 10$ and $\mathbf{p} = \{0.91, 0.01, \ldots, 0.01\}$. For L_{CE}, from (4) we can get:

$$\frac{\partial L_{\text{CE}}}{\partial \boldsymbol{w}_j} = p_j \boldsymbol{z}_i = 0.01 \boldsymbol{z}_i\,. \tag{23}$$

For L_{SqrtKL}, we can also calculate the gradient w.r.t. $\boldsymbol{w}_j$ from (21) after numerical substitution:

$$\frac{\partial L_{\text{SqrtKL}}}{\partial \boldsymbol{w}_j} = \approx -0.021 \boldsymbol{z}_i\,, \tag{24}$$

where the update range of $\boldsymbol{w}_j$ has been expanded by over two times. Hence, we can see how L_{SqrtKL} alleviate the infrequent updating problem by giving more gradients to $\boldsymbol{w}_j$ $(j \neq i)$ and it will be further alleviated as we increase the coefficient λ.

Table 8: Ablation study on L_{SqrtKL}.

Loss Formulation	CIFAR-10
-	88.8
$L_1 = \sum_k p_k \log p_k$	90.7
$L_2 = -\sum_k p_k \log u_k$	89.8
$L_{\text{SqrtKL}} = L_1 + L_2$	**91.1**

A.2. Ablation Study on L_{SqrtKL}

In Sec. 3.3.2 we analyzed that our proposed L_{SqrtKL} can be decomposed into two components:

$$L_{\text{SqrtKL}} = \underbrace{\sum_k p_k \log p_k}_{L_1} \underbrace{- \sum_k p_k \log u_k}_{L_2}\,, \tag{25}$$

where L_2 makes the distribution sharper while L_1 makes the distribution flatter. To further demonstrate the effectiveness of our method, we experiment with only L_1 or L_2, noting that all these variants use L_{CE}. As shown in Table 8, we can find that only using L_1 (i.e., maximizing entropy) can achieve good results. Note that L_1 can also alleviate the infrequent updating problem and make the distribution flatter. We can see that combining L_1 and L_2 can get better results, and L_1 plays a more important role in L_{SqrtKL}.

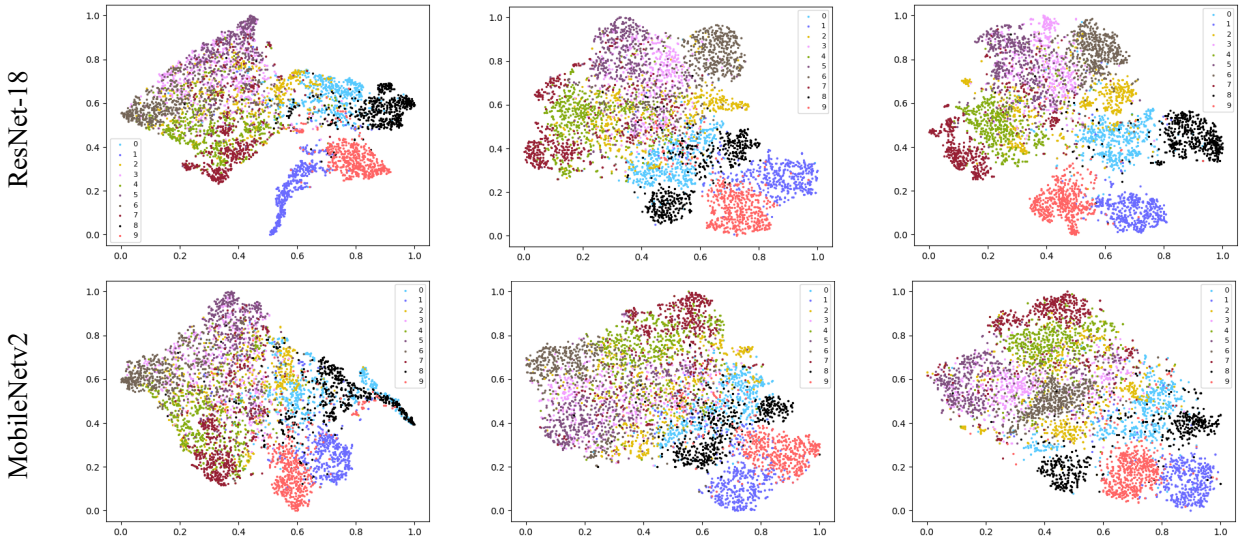
B. t-SNE Visualization

To demonstrate the effectiveness of the proposed method in a more intuitive way, we visualize the feature spaces learned by different methods in Fig. 6. First, three models are trained on the CIFAR-10 dataset by using SimCLR, SimSiam and our method, respectively. After that, 5,000 samples in CIFAR-10 are represented accordingly and then are reduced to a two-dimensional space by t-SNE [43]. As seen, the samples are more separable in the feature space learned by our method than both MoCov2 and SimSiam (especially under MobileNetv2).

C. ImageNet Subsets Experiments

As a supplement to Table 3 in Sec. 4.3, we transfer the learned representations on ImageNet subsets to CIFAR-10 and we report the linear probing accuracy on CIFAR-10 for comparison in Table 9. For better illustration, we also visualize these results in Fig. 7. We can reach similar conclusions as in the paper:

- Our method outperforms baseline counterpart MoCov2 consistently using different backbones and different scales of training images, with less training cost.
- Our method's advantages are more obvious when the amount of data is reduced. Take MobileNetv3-small as an example, the improvement of our method is 0.7% when

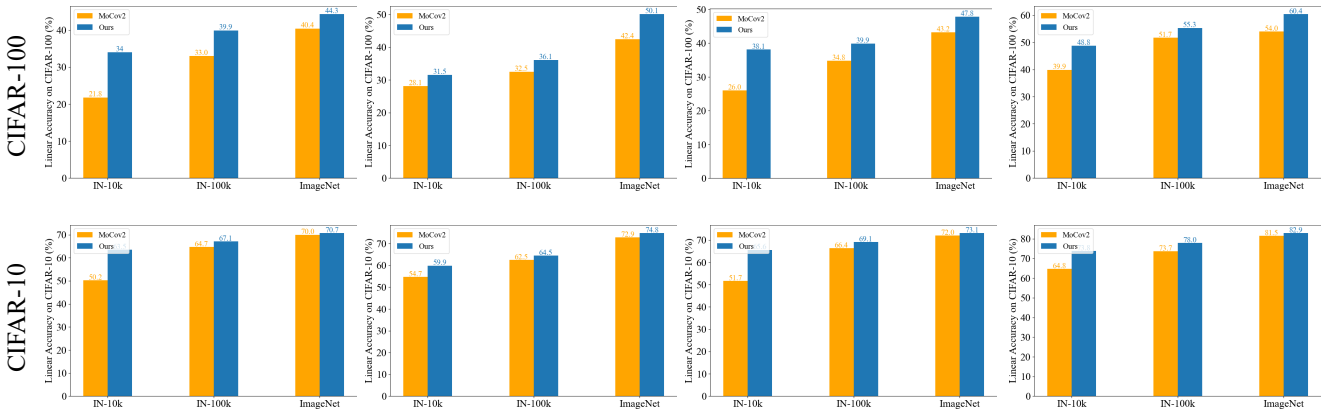


(a) SimSiam (b) MoCov2 (c) Ours

Figure 6: t-SNE [43] visualization of CIFAR-10 using ResNet-18. The column (a), (b) and (c) show the results of SimSiam, MoCov2 and our method, respectively. This figure is best viewed in color.

trained on ImageNet, and it increases to 2.4% on IN-100k and 13.3% on IN-10k.

- The amount of data required is positively correlated with the capacity of the model. Take Mobv3-small and Mobv3-large as an example, we can see that Mobv3-small even achieves better performance than Mobv3-large on IN-10k and IN-100k. It indicates that when the capacity of the model is small (i.e., has fewer parameters), a small amount of training data is enough, and the benefits brought by increasing the amount of data will become smaller and smaller. On the contrary, when the capacity of the model is large, the benefit of increasing the amount of data will be greater than that of the small model.



(a) MobileNetv3-Small (b) MobileNetv3-Large (c) EfficientNet-b0 (d) ResNet-18

Figure 7: Comparison of our method and MoCov2 when pretrained on ImageNet subsets and then transferred to downstream recognition datasets. Upper row: Transferring to CIFAR-100. Bottom row: Transferring to CIFAR-10.

Table 9: ImageNet (subsets) pretraining results on small architectures. All pretrained for 200 epochs and we report the linear evaluation accuracy (%) when transferring to CIFAR-10 and the pretraining hours using 8 3090 cards. †: Results from [40].

Backbone	# Images	10,000		100,000		1,281,167	
	Method	Linear (%) ↑	Cost (h) ↓	Linear (%) ↑	Cost (h) ↓	Linear (%) ↑	Cost (h) ↓
Mobv3-small (2.5M)	MoCov2	50.2	0.42	64.7	4.18	70.0†	53.55
	Ours	**63.5**	**0.34**	**67.1**	**3.43**	**70.7**	**43.94**
Mobv3-large (5.4M)	MoCov2	54.7	0.42	62.5	4.23	72.9†	54.19
	Ours	**59.9**	**0.38**	**64.5**	**3.79**	**74.8**	**48.56**
EfficientNet-b0 (5.3M)	MoCov2	51.7	0.43	66.4	4.31	72.0†	55.22
	Ours	**65.6**	**0.39**	**69.1**	**3.87**	**73.1**	**49.56**
ResNet-18 (11.7M)	MoCov2	64.8	0.43	73.7	4.33	81.5†	55.47
	Ours	**73.8**	**0.28**	**78.0**	**2.81**	**82.9**	**36.05**