

## SimCLIC: A Simple Framework for Contrastive Learning of Image Classification

Han YANG

*College of Computer Science & Technology, Qingdao University, Qingdao 266000, China*  
*E-mail: 2020020644@qdu.edu.cn*

Jun LI\*

*College of Computer Science & Technology, Qingdao University, Qingdao 266000, China*  
*E-mail: lijun@qdu.edu.cn*

**Abstract** Contrastive learning, a self-supervised learning method, is widely used in image representation learning. The core idea is to close the distance between positive sample pairs and increase the distance between negative sample pairs in the representation space. Siamese networks are the most common structure among various current contrastive learning models. However, contrastive learning using positive and negative sample pairs on large datasets is computationally expensive. In addition, there are cases where positive samples are mislabeled as negative samples. Contrastive learning without negative sample pairs can still learn good representations. In this paper, we propose a simple framework for contrastive learning of image classification (SimCLIC). SimCLIC simplifies the Siamese network and is able to learn the representation of an image without negative sample pairs and momentum encoders. It is mainly by perturbing the image representation generated by the encoder to generate different contrastive views. We apply three representation perturbation methods, namely, history representation, representation dropout, and representation noise. We conducted experiments on several benchmark datasets to compare with current popular models, using image classification accuracy as a measure, and the results show that our SimCLIC is competitive. Finally, we did ablation experiments to verify the effect of different hyperparameters and structures on the model effectiveness.

**Keywords** contrastive learning; representation learning; image classification

## 1 Introduction

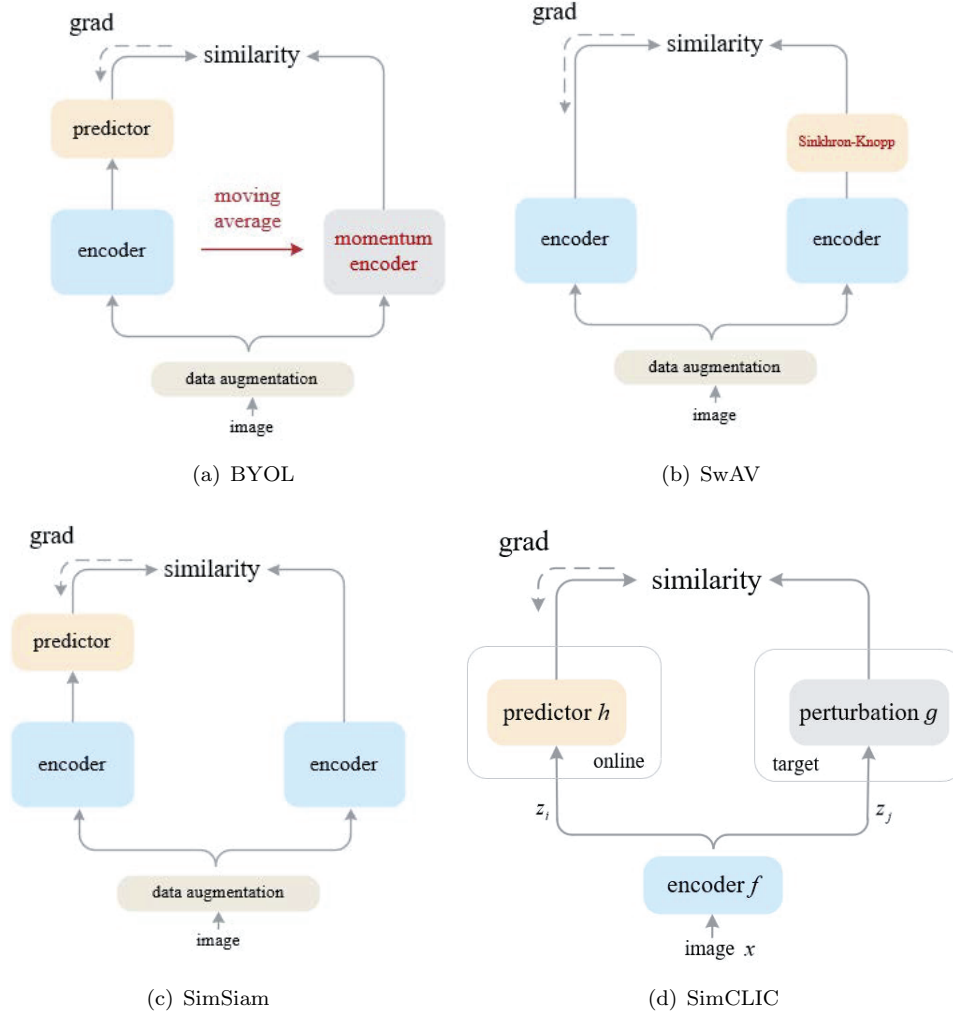
Image classification is one of the most important tasks in computer vision. Deep learning is currently the most effective method to address image classification. Supervised learning methods<sup>[1–5]</sup> are limited by the scarcity of labeled data and the high cost of manually labeled data. Self-supervised learning methods<sup>[6–17]</sup> have been applied to image classification tasks, and research results have shown that self-supervised learning methods can achieve competitive or even better results compared to supervised learning methods. The basic structure of self-supervised learning is the use of a Siamese network (shown in Figure 1), which is a network containing two symmetric networks (online network and target network) for obtaining the representation of the input image after two augmentations, maximizing the mutual information

---

Received August 18, 2022, accepted January 31, 2023

\*Corresponding author

between them. The core of self-supervised learning requires learning a representation learning model, by which similar instances are made closer in the representation space, while dissimilar instances are further away. The key to self-supervised learning is how to construct similar instances, and dissimilar instances, and how to prevent model collapse. Contrastive learning methods in general can be divided into two categories: Contrast-based methods<sup>[6–8, 11–14, 16]</sup> and clustering-based methods<sup>[9, 10, 15, 17]</sup>.



**Figure 1** Comparison on Siamese architectures. For simplicity, we only box out the on-line network and the target network in SimCLIC. The dashed line indicates the gradient propagation flow, the lack of dashed line means stop-gradient

Currently, self-supervised learning in computer vision usually uses data augmentation, and large changes in the image are likely to invert image distortion, and there is no standard solution for how to ensure that the representation of an image remains unchanged during data augmentation. In this paper, we propose a simple framework for contrastive learning of image classification (SimCLIC) (shown in Figure 1(d)), which perturbs the image representation posteriorly to obtain a contrastive view. Structurally, our Siamese network is slightly different

from SimSiam<sup>[18]</sup> and BYOL<sup>[14]</sup> in that the online network and the target network share the same encoder. As with SimSiam<sup>[18]</sup>, SimCLIC uses neither negative pairs nor a momentum encoder. We applied three different post perturbation methods to generate different but related contrastive views. We conducted experiments on several benchmark datasets and the results demonstrate that SimCLIC’s image classification performance are highly competitive. We summarize our contributions as follows:

- We propose a novel contrastive learning framework: SimCLIC, which constructs contrastive views for contrastive learning by perturbing the representation of images.
- We employ three different perturbation methods: History representation, representation dropout and representation noise.
- We conducted experiments on several benchmark datasets and the performance compared with other contrastive learning methods show that SimCLIC is highly competitive.
- We verified the effect of different hyperparameters and structures on the effect of the model through ablation experiments.

The remainder of this paper is organized as follows. Section 2 provides an overview of the literature related to image classification, and contrastive learning. Section 3 provides a comprehensive and systematic introduction to the various parts of SimCLIC and the implementation details. Section 4 describes the experiments we performed and the related conclusions. Section 6 concludes this paper.

## 2 Related Works

In this section, we first give a brief introduction to image classification, and then detail the contrastive learning.

**Image classification.** Image classification mainly solves the problem of whether an image contains a certain type of object, and feature description of the image is the main research content of image classification. Image classification methods describe the image globally by manual features or feature learning methods, and then use classifiers to determine whether there is a certain class of objects. Early image classification methods<sup>[19–21]</sup> mainly extracted image features manually, such as color features, texture features, and local features, etc. There is still a large “semantic gap” between the underlying features of images obtained by shallow learning and the high-level themes of images. Deep learning methods<sup>[1–5, 22–25]</sup> use a network structure to learn the hierarchical structural features of images, which can extract abstract features closer to the high-level semantics of images and outperform traditional methods in image classification. Subject to the scarcity of labeled data and the high cost of labeling, self-supervised contrast learning methods have started to be widely used in vision tasks.

**Contrastive learning.** Self-supervised learning methods in visual representations are divided into two categories: Generative self-supervised learning and discriminative self-supervised learning. Generative learning is represented by adversarial learning<sup>[26]</sup> and automatic coding<sup>[27–29]</sup>. These types of methods usually operate in pixel space, leading to expensive computational overhead. Among the discriminative methods, contrastive learning<sup>[6–17]</sup> is by far the most widely

used and also the best performing. Contrastive learning methods learn the representation of an image by constructing a contrastive view that pulls together positive pairs and pushes away negative pairs.

As the pioneer of contrastive learning, InstDisc<sup>[6]</sup>, treats each sample as a separate class, allowing the model to learn features that distinguish these individual instances so that the final learned features capture the similarity between samples. Due to the large number of samples in the dataset, InstDisc<sup>[6]</sup> uses a memory bank to store the features of all samples in order to reduce computation. Based on the property that different human sensory perspectives share some important information, CMC<sup>[30]</sup> proposes multi-view contrastive learning to maximize the mutual information between different sensory perspectives, so that the features learned by the model satisfy the view-invariant. MoCo and SimCLR are two series of methods that draw on each other in contrastive learning. MoCo v1<sup>[12]</sup> also exploits the idea of memory bank, which treats contrastive learning as a dictionary look-up task. To improve the quality of the learned representation, SimCLR v1<sup>[11]</sup> introduces a learnable nonlinear transformation projection head between representation and contrastive loss. MoCo v2<sup>[13]</sup> incorporates some techniques from SimCLR v1<sup>[11]</sup> into MoCo v1<sup>[12]</sup>, including projection head, more data augmentations and longer training time. Compared to SimCLR v1<sup>[11]</sup>, SimCLR v2<sup>[7]</sup> uses a larger network model for image representation, a deeper projection head, and uses MoCo v1’s queueing dictionary and momentum encoder.

Individual discriminant-based contrastive learning treats each sample as a separate class. However, even if 60,000 negative samples are taken for contrastive learning as in MoCo v1<sup>[12]</sup>, it is still only an approximation, and the sampled negative samples may be unbalanced, and some may even be positive, making model learning difficult. SwAV<sup>[15]</sup> believes that it is not necessary to compare with all negative samples, but can use the idea of clustering to find  $K$  cluster centers to represent the whole dataset, and then the cluster centers can generate low-dimensional soft labels for each sample, and finally use “swapped” prediction to perform contrastive learning. SwAV<sup>[15]</sup> also proposes multi-crop, which randomly crops and resizes 2 global views and  $V$  local views. It can capture the local features of the images well without increasing the memory and computational load excessively at a smaller resolution. Negative samples play an important role in contrastive learning to avoid model collapse. However, BYOL<sup>[14]</sup> achieves good results by using only positive samples, converting the original matching problem into a prediction problem, and training the model through a prediction-based task. SimSiam<sup>[18]</sup> demonstrates that simple Siamese networks can learn meaningful representations even without using negative sample pairs, large batches, and momentum encoders.

Transformer<sup>[31]</sup> is a great achievement in deep learning, and Vision Transformer (ViT)<sup>[32]</sup> shows that Transformer<sup>[31]</sup> can be used in computer vision. MoCo v3<sup>[33]</sup> and DINO<sup>[34]</sup> both use ViT<sup>[32]</sup> as a backbone, and the former solves the training instability problem by directly freezing the patch projection layer, while the latter adds centering to avoid model collapse.

### 3 Method

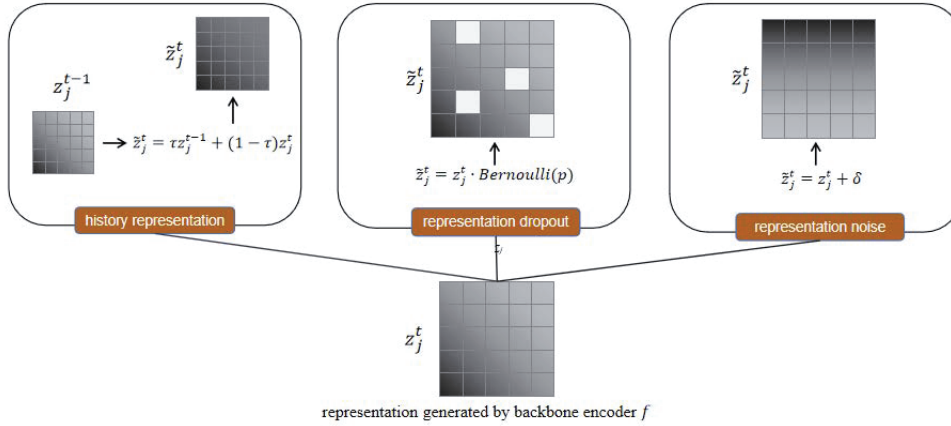
In this section we will describe in detail the various parts of SimCLIC and their implementation.

### 3.1 Framework

Our framework, shown in Figure 1d, is partially similar to SimSiam’s Siamese network architecture. It does not use either negative pairs or momentum encoders. In SimCLIC, the encoder  $f$ , the online network and the target network are included. Since data augmentation is discarded, we construct a different but related view of the online network in the target network by representing perturbations.

### 3.2 Representation Perturbation

Currently, contrastive learning methods use image augmentation to generate two different but related views. In SimCLIC, we generate two contrastive views by perturbing the representation of the image. We apply three methods to introduce representation perturbation, namely history representation, representation dropout and representation noise, as shown in Figure 2.



**Figure 2** The elaboration of three different representation perturbations

**History representation.** Inspired by the momentum update mechanism, we introduce representation perturbations by exploiting historical representations from previous training iterations. Specifically, we use momentum updates to generate representations in the target network. Suppose  $z_j^t$  is the representation generated by the  $t$ th iteration of the encoder  $f$ . The output of the target network (the perturbed representation  $\tilde{z}_j^t$ ) is computed by combining the output embedding  $z_j^t$  with the historical embedding  $z_j^{t-1}$ :

$$\tilde{z}_j^t = \tau z_j^{t-1} + (1 - \tau) z_j^t, \quad (1)$$

where  $\tau \in [0, 1)$  is a parameter that controls the proportion of information from the current and previous iterations.

**Representation dropout.** We apply the representation dropout method to achieve a perturbation of the image representation in the target network to obtain a contrastive view. Similar to dropout in convolutional neural networks, the information in each dimension of the image’s representation is set to 0 with probability  $p$ :

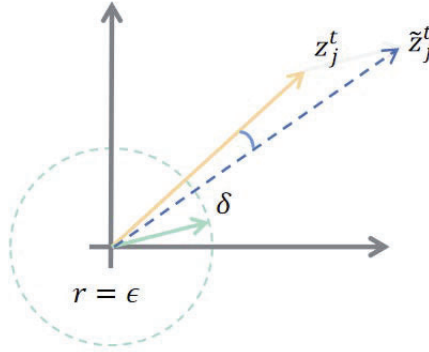
$$\tilde{z}_j^t = z_j^t \cdot \text{Bernoulli}(p). \quad (2)$$

To prevent too much information loss, which can lead to changes in the semantic information of the image, we restrict  $p$ . The effect of the  $p$  value on the model will be described in more detail in the ablation study section.

**Representation noise.** Inspired by [35] and [36], we perturb the image representation by adding noise. Formally, given an image to obtain its representation  $z_j$  in the  $d$ -dimensional representation space generated by the encoder  $f$ , we can achieve the following representation-level augmentation:

$$\tilde{z}_j^t = z_j^t + \delta. \quad (3)$$

To prevent the addition of noise from causing changes in the semantic information of the image, we restrict the size of the noise, where the added noise vector  $\delta$  is subject to  $\|\delta\|_2 = \epsilon$  and  $\delta = \bar{\delta} \odot \text{sign}(z_j)$ ,  $\bar{\delta} \in \mathbb{R}^d \sim U(0, 1)$ . We elaborate on it in two dimensions, as shown in Figure 3. By the above two conditions, we restrict  $\delta$  to be numerically equivalent to a point on a hypersphere of radius  $\epsilon$  and constrain that  $z_j, \delta$  should be in the same hyperoctet.



**Figure 3** The elaboration of three different representation perturbations

### 3.3 Loss Function

Our architecture takes image  $x$  as input. A representation  $z_i$  and  $z_j$  ( $z_i = z_j$ ) of the image generated by the encoder  $f$  (consisting of a backbone network and a projection multi-layer perception (MLP) head) is obtained. The predictor MLP head denoted as  $h$  maps the representation  $z_i$  to a latent space to obtain  $p_i$ , which is matched to the perturbed representation  $\tilde{z}_j$ . The two output vectors are represented as  $p_i = h(f(x))$  and  $\tilde{z}_j = g(f(x))$ , minimizing their negative cosine similarity as follows:

$$\text{sim}(p_i, \tilde{z}_j) = -\frac{p_i}{\|p_i\|_2} \cdot \frac{\tilde{z}_j}{\|\tilde{z}_j\|_2}, \quad (4)$$

where  $\|\cdot\|$  is  $l_2$ -norm. Following [14] and [18], we define a symmetric loss function  $\mathcal{L}$  as follows, which is defined for each image, and the total loss is the average of all images. It has a minimum possible value of  $-1$ .

$$\mathcal{L} = \frac{1}{2}\text{sim}(p_i, \tilde{z}_j) + \frac{1}{2}\text{sim}(p_j, \tilde{z}_i). \quad (5)$$

Following [18], we perform the operation of stopping the gradient on the target network and back-propagating the loss only on the online network, so we update the equation (5) to:

$$\mathcal{L} = \frac{1}{2}\text{sim}(p_i, \text{stopgrad}(\tilde{z}_j)) + \frac{1}{2}\text{sim}(p_j, \text{stopgrad}(\tilde{z}_i)). \quad (6)$$

The pseudo-code of SimCLIC is in Algorithm 1.

---

**Algorithm 1** SimCLIC Pseudocode, PyTorch-like

---

**Require:** image dataset  $\mathbb{D}$

**Require:**  $f, h, g$  #encoder, predictor, perturbation

```

1: for  $\mathbb{D}^t$  in  $\mathbb{D}$  do #load a batch
2:    $z_i = z_j = f(x)$  #output of encoder
3:    $p_i, p_j = h(z_i), h(z_j)$  #output of predictor
4:    $\tilde{z}_i, \tilde{z}_j = g(p_i), g(p_j)$  #output of perturbation
5:    $\mathcal{L} = \frac{1}{2}\text{sim}(p_i, \text{stopgrad}(\tilde{z}_j)) + \frac{1}{2}\text{sim}(p_j, \text{stopgrad}(\tilde{z}_i))$  #loss calculation
6:    $\mathcal{L}.\text{backward}()$  #back-propagate
7:    $\text{update}(f, h)$  #parameters update
8: end for
9: def  $\text{sim}(p, z)$  : #negative cosine similarity
10:    $z = z.\text{detach}()$  #stop gradient
11:    $p = \text{normalize}(p, \text{dim} = 1)$  # $l_2$ -normalize
12:    $z = \text{normalize}(z, \text{dim} = 1)$  #  $l_2$ -normalize
13:   return  $-(p * z).\text{sum}(\text{dim} = 1).\text{mean}()$ 

```

---

### 3.4 Implementation Details

**Architecture.** We choose ResNet-50<sup>[4]</sup> as the backbone network for encoder  $f$ , followed by a projection MLP. The projection MLP applies batch normalization (BN) to each layer, including output layer. The MLP has 3 layers. The predictor MLP in the online network applies batch normalization (BN) to each layer, except for the output layer. The MLP has 2 layers. The representation of image is projected into a latent space in the online network via the predictor MLP, and a new representation is formed in the target network by perturbing  $g$ . The output of the encoder is used for downstream tasks and the outputs of the online and target networks are used to compute the contrastive loss.

**Optimization.** We train 1000 epochs using the stochastic gradient descent (SGD) optimizer with a batch size of 128. The base learning rate = 0.05 and update the learning rate using cosine decay. The SGD momentum is 0.9. The weight decay is 0.001.

## 4 Experiments

We perform unsupervised pretraining on benchmark datasets without labels. The trained representation of the training set is frozen, on which a supervised linear classifier is trained to evaluate the quality of the pre-trained representation, and then tested in the validation set.

#### 4.1 Datasets

We have done experiments on datasets FGVC Aircraft<sup>[37]</sup>, Caltech101<sup>[38]</sup>, Stanford Cars<sup>[39]</sup>, CIFAR10, CIFAR100<sup>[40]</sup>, DTD<sup>[41]</sup>, Flowers<sup>[42]</sup>, Food-101<sup>[43]</sup>, Oxford-IIIT Pets<sup>[44]</sup>, SUN397<sup>[45]</sup> and Pascal VOC2007<sup>[46]</sup>. For the datasets with undefined training and validation sets, we randomly select 80% of them as the training set and the rest as the validation set. The number of classes and the total number of images contained in each dataset are shown in Table 1.

**Table 1** The statistics of datasets

	Aircraft	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Flowers	Food	Pets	SUN397	VOC2007
Classes	100	101	196	10	100	47	102	101	37	397	20
Total number	10000	≈9000	16185	60000	60000	5640	8189	101000	≈5000	108000	9963

#### 4.2 Experimental Results

We evaluate the performance of SimCLIC after self-supervised pre-training on the training set from several benchmark datasets mentioned above. We evaluated it by performing it in linear evaluation and semi-supervised training, as well as a comparison with some other contrastive learning methods (Contrast-based: InsDis<sup>[6]</sup>, MoCo v1<sup>[12]</sup> and its upgrade MoCo v2<sup>[13]</sup>, PIRL<sup>[8]</sup>, SimCLR v1<sup>[11]</sup> and its upgrade SimCLR v2<sup>[7]</sup>, InfoMin<sup>[16]</sup> and BYOL<sup>[14]</sup>; cluster-based: PCL and PCL v2<sup>[9]</sup>, SeLa v2<sup>[10]</sup>, DeepCluster v2<sup>[17]</sup> and SwAV<sup>[15]</sup>). The results are reported in Table 2. Except for the SimCLIC experimental results, the rest of the data are cited from the literature [47]. We use **bold** to denote the best result on each dataset and underline to denote the second best.

**Linear evaluation.** We follow the linear evaluation protocol of [14, 48, 49] to train a regularized multinomial logistic regression classifier on a frozen final representation of the backbone network ResNet-50<sup>[4]</sup> in the encoder  $f$  of SimCLIC. During training and testing, no data augmentation was used. The images were resized to 224 pixels along the shorter side using bicubic resampling. The cross-entropy objective is minimized using L-BFGS<sup>[50]</sup> with  $l_2$  regularization, where we select the regularization parameters over 45 logarithmically spaced values between  $10^{-6}$  and  $10^5$ . For the multi-label classification task (Pascal VOC2007), we fit a binary classifier for each class. The upper part of Table 2 reports the classification error rate of the validation set in each benchmark dataset. We can find that BYOL<sup>[14]</sup>, DeepCluster v2<sup>[17]</sup> and SwAV<sup>[15]</sup> perform well in each dataset, especially DeepCluster v2<sup>[17]</sup>. SimCLIC achieves a state-of-the-art (SOTA) on CIFAR10<sup>[40]</sup>. From the experimental results of SimCLIC for three different perturbations, history representation and representation noise have similar performance, and representation dropout has slightly lower classification accuracy.

**Fine-tuning.** We followed the protocol of [11, 47] to fine-tune the model with slight modifications. We chose SGD with Nesterov momentum and a momentum parameter of 0.9 as the optimizer to optimize the loss in 5000 steps with a batch size of 128. The learning rate was updated using cosine annealing. The learning rate and weight decay were chosen for a grid of seven log-interval learning rates between 0.0001 and 0.1, and for seven log-interval weight decay values between  $10^{-6}$  and  $10^{-3}$ , and no weight decay, respectively. The lower part of



Table 2 reports the classification error rate in the validation set of several benchmark datasets. It can be found that BYOL<sup>[14]</sup>, DeepCluster v2<sup>[17]</sup> and SwAV<sup>[15]</sup> still perform well on each benchmark dataset. DeepCluster v2<sup>[17]</sup> achieves SOTA on six of the datasets and second best on four datasets, SwAV<sup>[15]</sup> achieves SOTA on two datasets and second best on three datasets. SimCLIC achieves second best on CIFAR10<sup>[40]</sup> and Flowers<sup>[42]</sup>.

Since we have no control over the pre-training settings, the architectures and parameters used by the models vary, as shown in Table 3. We can see that under the existing conditions, we can set the maximum batch size to 128, which is the minimum compared to other contrastive learning methods. It can be predicted that the performance of SimCLIC will be further improved as the batch size increases. In section 5.3 we will further investigate the effect of batch size on SimCLIC.

**Table 2** Linear evaluation and fine-tuning error rates (%)

	ImageNet	Aircraft	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Flowers	Food	Pets	SUN397	VOC2007
Linear evaluation:												
InsDis	40.50	63.13	28.88	71.02	19.72	40.03	31.54	16.56	36.61	31.22	50.53	25.63
MoCo v1	39.40	64.45	24.67	72.01	19.84	42.29	31.17	17.90	37.90	30.16	48.98	24.07
PCL	38.50	78.39	23.10	87.07	18.16	44.26	37.13	35.27	51.98	24.66	54.30	21.69
PIRL	38.30	62.92	25.52	71.28	17.47	38.74	31.01	16.40	35.35	28.64	46.11	23.39
PCL v2	32.40	62.97	13.58	69.49	8.09	26.46	29.41	14.66	35.12	17.21	43.75	18.86
SimCLR v1	30.70	55.10	9.95	56.27	8.82	27.27	25.80	9.13	32.53	16.67	40.79	19.23
MoCo v2	28.90	58.21	12.08	60.69	7.72	25.10	26.12	9.93	31.05	16.70	39.68	17.31
SimCLR v2	28.30	53.62	10.37	49.63	7.47	23.22	23.62	7.10	26.92	15.28	38.53	18.43
SeLa v2	28.20	62.71	12.80	63.14	7.27	25.19	25.85	9.78	28.92	16.78	37.29	17.27
InfoMin	27.00	61.42	12.16	58.96	8.51	26.57	25.27	12.82	30.47	13.76	39.00	16.76
BYOL	25.70	46.13	<b>8.54</b>	<u>43.60</u>	6.74	22.14	23.09	5.5	26.99	<u>10.90</u>	40.01	18.86
DeepCluster v2	<u>24.80</u>	<b>45.51</b>	<u>8.67</u>	<b>41.40</b>	5.98	<b>20.39</b>	<b>21.38</b>	<b>5.28</b>	<b>22.06</b>	<b>10.64</b>	<u>34.52</u>	<b>16.06</b>
SwAV	<b>24.70</b>	<u>45.96</u>	9.16	45.94	6.01	<u>20.42</u>	<u>22.98</u>	<u>5.38</u>	<u>23.38</u>	12.40	<b>34.42</b>	<u>16.32</u>
Fine-tuned:												
InsDis		26.62	27.96	38.44	6.68	31.74	36.01	10.49	23.22	23.78	48.16	28.10
MoCo v1		24.39	25.05	34.88	6.11	28.48	34.63	10.55	22.72	23.04	46.65	25.09
PCL		25.03	12.38	26.76	3.65	20.38	30.00	9.17	21.70	13.02	41.60	17.92
PIRL		27.32	29.17	38.98	7.77	33.52	35.74	10.19	25.04	23.74	49.62	30.10
PCL v2		20.63	11.96	28.32	3.50	19.74	28.24	7.05	19.66	14.61	41.18	17.80
SimCLR v1		18.94	<u>9.65</u>	16.22	<b>2.93</b>	<u>15.47</u>	28.46	6.25	17.60	15.90	36.69	17.42
MoCo v2		20.13	15.62	24.80	3.55	28.67	30.53	5.65	23.22	20.20	44.23	28.29
SimCLR v2		21.29	17.06	20.16	3.78	20.95	29.84	5.68	17.78	16.80	38.88	21.81
SeLa v2		18.01	11.01	14.38	3.20	15.63	25.64	<b>4.20</b>	13.76	11.45	34.16	<u>15.15</u>
InfoMin		19.76	16.08	21.24	3.06	28.85	28.88	4.76	21.07	14.72	42.34	23.37
BYOL		20.55	10.60	15.40	2.99	16.05	26.38	5.52	14.45	<b>10.38</b>	36.04	17.30
DeepCluster v2		<u>17.48</u>	<b>9.25</b>	<b>12.73</b>	<u>2.94</u>	<b>14.85</b>	<u>25.16</u>	4.69	<b>12.49</b>	<u>10.57</u>	<b>33.58</b>	<b>15.10</b>
SwAV		<b>16.92</b>	10.15	<u>13.24</u>	3.22	15.63	<b>24.84</b>	4.54	<u>12.78</u>	10.95	<u>33.76</u>	15.34
Fine-tuned:												
InsDis	history	56.15	12.12	54.26	<u>5.97</u>	25.96	26.40	9.52	28.76	18.78	40.18	18.56
SimCLIC (ours)	dropout	57.09	12.35	54.52	6.00	26.18	27.02	9.47	29.82	18.93	40.61	18.89
	noise	55.86	11.92	53.89	<b>5.92</b>	25.65	26.04	9.50	29.51	18.75	40.12	18.55
Fine-tuned:												
InsDis		26.62	27.96	38.44	6.68	31.74	36.01	10.49	23.22	23.78	48.16	28.10
MoCo v1		24.39	25.05	34.88	6.11	28.48	34.63	10.55	22.72	23.04	46.65	25.09
PCL		25.03	12.38	26.76	3.65	20.38	30.00	9.17	21.70	13.02	41.60	17.92
PIRL		27.32	29.17	38.98	7.77	33.52	35.74	10.19	25.04	23.74	49.62	30.10
PCL v2		20.63	11.96	28.32	3.50	19.74	28.24	7.05	19.66	14.61	41.18	17.80
SimCLR v1		18.94	<u>9.65</u>	16.22	<b>2.93</b>	<u>15.47</u>	28.46	6.25	17.60	15.90	36.69	17.42
MoCo v2		20.13	15.62	24.80	3.55	28.67	30.53	5.65	23.22	20.20	44.23	28.29
SimCLR v2		21.29	17.06	20.16	3.78	20.95	29.84	5.68	17.78	16.80	38.88	21.81
SeLa v2		18.01	11.01	14.38	3.20	15.63	25.64	<b>4.20</b>	13.76	11.45	34.16	<u>15.15</u>
InfoMin		19.76	16.08	21.24	3.06	28.85	28.88	4.76	21.07	14.72	42.34	23.37
BYOL		20.55	10.60	15.40	2.99	16.05	26.38	5.52	14.45	<b>10.38</b>	36.04	17.30
DeepCluster v2		<u>17.48</u>	<b>9.25</b>	<b>12.73</b>	<u>2.94</u>	<b>14.85</b>	<u>25.16</u>	4.69	<b>12.49</b>	<u>10.57</u>	<b>33.58</b>	<b>15.10</b>
SwAV		<b>16.92</b>	10.15	<u>13.24</u>	3.22	15.63	<b>24.84</b>	4.54	<u>12.78</u>	10.95	<u>33.76</u>	15.34
Fine-tuned:												
InsDis	history	22.86	11.75	19.56	3.12	19.65	29.58	4.86	20.84	17.46	42.15	21.21
SimCLIC (ours)	dropout	22.24	12.27	20.12	3.11	20.48	29.46	5.68	20.97	17.79	42.68	21.86
	noise	22.46	11.72	19.89	<u>2.94</u>	19.62	28.92	<u>4.51</u>	21.18	17.28	42.38	21.37

### 4.3 Hyper-Parameter Sensitivity

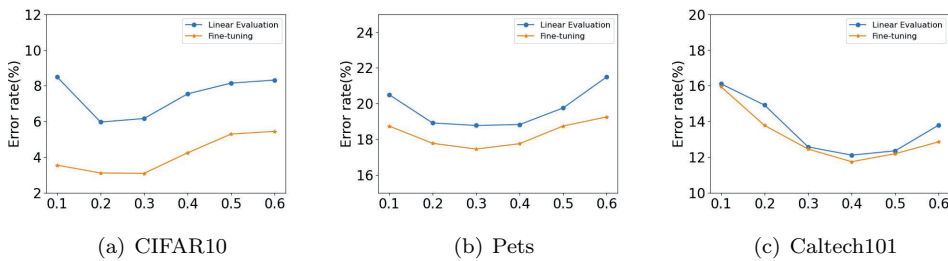
To guide hyperparameter selection, we perform a hyperparameter sensitivity study of SimCLIC's performance on CIFAR10<sup>[40]</sup>, Pets<sup>[44]</sup> and Caltech101<sup>[38]</sup> datasets with relatively large

differences in the number of classes. We investigate the variation in the performance of our framework with respect to the three hyperparameters of historical representation momentum  $\tau$ , representation dropout rate  $p$ , and representation noise size  $\epsilon$ .

**Table 3** Training details reported in the original paper

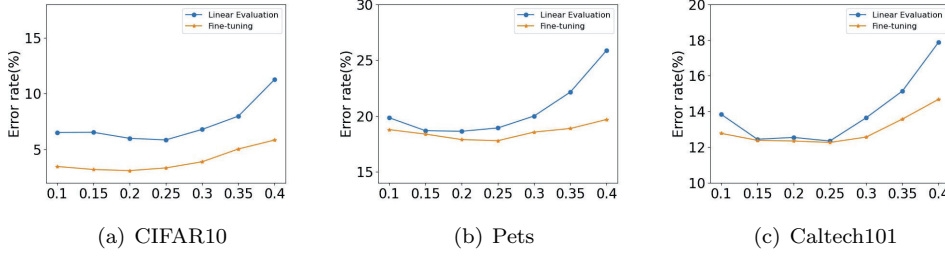
	Epochs	Batch size	Momentum encoder	Momentum bank	Projection head	Data augmentation
InsDis*	200	256		✓		✓
MoCo v1	200	256	✓			✓
PCL	200	256	✓			✓
PIRL*	200	1024		✓		✓
PCL v2	200	256	✓		✓	✓
SimCLR v1	1000	4096			✓	✓
MoCo v2	800	256	✓		✓	✓
SimCLR v2	800	4096	✓		✓	✓
SeLa v2	400	4096		✓	✓	✓
InfoMin	800	256	✓		✓	✓
BYOL	1000	4096			✓	✓
DeepCluster v2	800	4096		✓	✓	✓
SwAV	800	4096			✓	✓
SimCLIC (ours)	1000	128			✓	

**Effect of  $\tau$ .** We set the momentum parameter of SimCLIC  $\tau$  within  $[0.1, 0.6]$  with a step size of 0.1. We plot in Figure 4 the linear evaluation and fine-tuning classification error rates for different momentum parameter values. From Figure 4, we observe that SimCLIC performs consistently on linear evaluation and fine-tuning, with fine-tuning having a relatively lower error rate than linear evaluation. Comparing the results for the three datasets shows that the dataset with a larger number of classes benefits more from a larger value. The performance starts to decrease when  $\tau$  is greater than a certain level. In our analysis, the more the number of classes, the more difficult it is to classify, and more contrastive information is needed to improve the robustness of the model.

**Figure 4** The effect of the magnitude of the momentum parameter  $\tau$ 

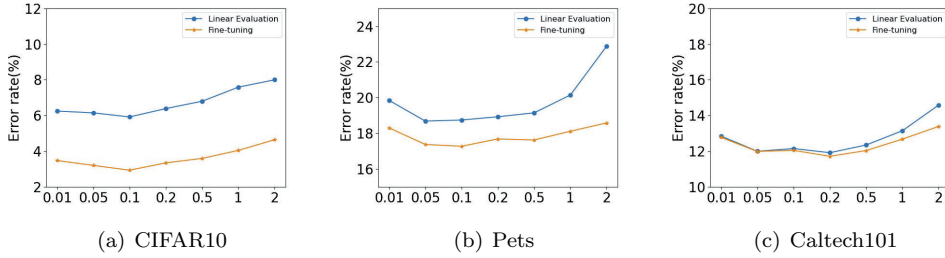
**Effect of  $p$ .** We set the dropout parameter  $p$  of SimCLIC within  $[0.1, 0.4]$  with a step size of 0.05. We plot the linear evaluation and fine-tuning classification error rates for different dropout probabilities  $p$  in Figure 5. We can see that SimCLIC benefits from an appropriately dropout probability  $p$  (e.g.,  $p = 0.25$ ). When  $p$  is greater than a certain value (e.g.,  $p = 0.25$ ), as  $p$  increases, the performance of the model decreases at an accelerated rate. We analyze

that when  $p$  increases to a certain level, the semantic information of the image may change as  $p$  increases again, thus affecting the performance of SimCLIC. It is also observed that the gap between linear evaluation and fine-tuning becomes larger when  $p$  is greater than a certain degree, which proves the greater the role of label.



**Figure 5** The effect of the magnitude of the dropout parameter  $p$

**Effect of  $\epsilon$ .** The size of  $\epsilon$  directly determines the size of the noise, and we plot the effect of different noise sizes on the SimCLIC in Figure 6. We can see that the appropriate size of noise is beneficial to improve the performance of SimCLIC. And the magnitude of the impact of different noise sizes is small compared to the history representation and representation dropout.



**Figure 6** The effect of the magnitude of the noise parameter  $\epsilon$

## 5 Ablation Study

We study each component in SimCLIC to explore its contribution to the classification performance of SimCLIC. We used the representation noise on the CIFAR<sup>[40]</sup> dataset for the ablation study.

### 5.1 Predictor

We investigated the classification performance of several different predictor compositions, and Table 4 summarizes the classification performance. We can see that, as with SimSiam<sup>[18]</sup>, SimCLIC no longer works when the predictor  $h$  is removed. When a fixed random initialization is used, SimCLIC training is difficult to converge resulting in the model not learning a useful representation. We analyze that since the dataset has only 10 classes, the final classification accuracy of both can still be maintained at about 10%. The more MLP layers of projector the better the result, which is consistent with the findings of SimCLR v2<sup>[7]</sup>.

**Table 4** Effect of predictor  $h$ 

	No predictor	Fixed random init	1-layer MLP	2-layer MLP
Error rate (%)	90.17	89.16	8.65	5.92

## 5.2 Stop-Gradient

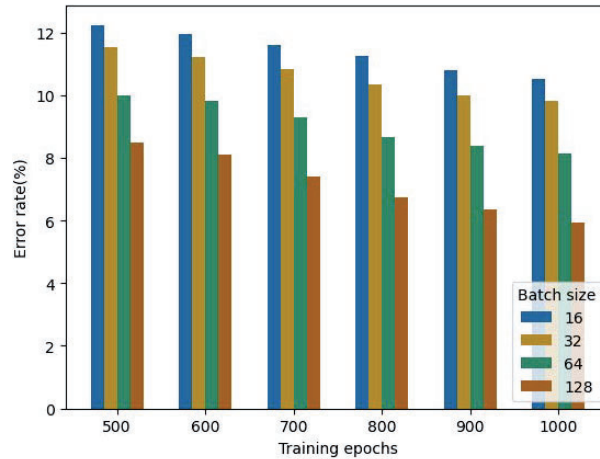
Existing researches<sup>[14, 15, 18]</sup> argues that stop-gradient play a key role in preventing collapse of the contrastive learning model. We kept the architecture and all hyperparameters constant and evaluated SimCLIC by removing or adding the stop-gradient operator. As the results in Table 5 show that the stop-gradient is indispensable in SimCLIC as in other studies<sup>[14, 15, 18]</sup>.

**Table 5** Effect of stop-gradient operator

	With stop-gradient	Without stop-gradient
Error rate (%)	5.92	91.04

## 5.3 Batch Size

We test the performance of SimCLIC by setting the batch size and epoch from 16 to 128 and 500 to 1000, respectively, while keeping the architecture and parameters unchanged, as shown in Figure 7. From this, we can see that the classification error rate of SimCLIC decreases as both batch size and epoch increase when batch size and epoch are set in these two intervals. We predict that the classification error rate of SimCLIC will further decrease as the batch size increases. Limited by the equipment, we cannot do further validation.

**Figure 7** Effect of batch sizes

## 5.4 Loss Function

In addition to cosine similarity, cross-entropy similarity is also one of the most common similarity measurement functions in contrastive learning. We replace the cosine similarity with

the cross-entropy similarity by modifying Equation (4) to:

$$\text{sim}(p_i, \tilde{z}_j) = -\text{softmax}(\tilde{z}_j) \cdot \log(\text{softmax}(p_i)), \quad (7)$$

keeping the other architectures and parameters unchanged. The comparison of the performance is presented in Table 6. The cross-entropy similarity is not as good as the cosine similarity, but it still makes the SimCLIC converge to prevent model collapse.

**Table 6** Effect of loss function

	Cosine	Cross-entropy
Error rate (%)	5.92	10.69

## 6 Conclusion

In this paper, we propose a new contrastive learning framework without negative sample pairs and momentum encoders, SimCLIC, based on Siamese networks. It replaces data augmentation by perturbing the output of the encoder. We apply three different perturbation methods and encapsulate ResNet-50<sup>[4]</sup> into the framework. Experiments on several benchmark datasets show that SimCLIC is competitive and even performs better on individual datasets compared to other contrastive learning frameworks. Currently we have tested SimCLIC’s performance only on image classification tasks, and we will continue to test it on other computer vision tasks (e.g., semantic segmentation, instance segmentation, etc.) in the future.

## References

- [1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012, 25.
- [2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*, 2014.
- [3] Szegedy C, Liu W, Jia Y Q, et al. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015: 1–9.
- [4] He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 770–778.
- [5] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017: 4700–4708.
- [6] Wu Z R, Xiong Y J, Yu S X, et al. Unsupervised feature learning via non-parametric instance discrimination. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018: 3733–3742.
- [7] Chen T, Kornblith S, Swersky K, et al. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 2020, 33: 22243–22255.
- [8] Misra I, Van Der Maaten L. Self-supervised learning of pretext-invariant representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020: 6707–6717.
- [9] Li J N, Zhou P, Xiong C M, et al. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv: 2005.04966*, 2020.
- [10] Asano Y M, Rupprecht C, Vedaldi A. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv: 1911.05371*, 2019.
- [11] Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020: 1597–1607.
- [12] He K M, Fan H Q, Wu Y X, et al. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020: 9729–9738.
- [13] Chen X L, Fan H Q, Girshick R, et al. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv: 2003.04297*, 2020.

- [14] Grill J B, Strub F, Althé F, et al. Bootstrap your own latent—a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 2020, 33: 21271–21284.
- [15] Caron M, Misra I, Mairal J, et al. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 2020, 33: 9912–9924.
- [16] Tian Y L, Sun C, Poole B, et al. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 2020, 33: 6827–6839.
- [17] Caron M, Bojanowski P, Joulin A, et al. Deep clustering for unsupervised learning of visual features. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018: 132–149.
- [18] Chen X L, He K M. Exploring simple siamese representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021: 15750–15758.
- [19] Swain M J, Ballard D H. Color indexing. *International Journal of Computer Vision*, 1991, 7(1): 11–32.
- [20] Huang J, Kumar S R, Mitra M, et al. Image indexing using color correlograms. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997: 762–768.
- [21] Ojala T, Pietikainen M, Harwood D. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. *Proceedings of 12th International Conference on Pattern Recognition*, 1994, 1: 582–585.
- [22] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324.
- [23] Iandola F N, Han S, Moskewicz M W, et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv: 1602.07360*, 2016.
- [24] Qin Z, Zhang Z N, Chen X T, et al. Fd-mobilenet: Improved mobilenet with a fast downsampling strategy. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018: 1363–1367.
- [25] Zhang X Y, Zhou X Y, Lin M X, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018: 6848–6856.
- [26] Goodfellow I, PougeAbadie J, Mirza M, et al. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2014, 27.
- [27] Vincent P, Larochelle H, Bengio Y, et al. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 2008: 1096–1103.
- [28] Kingma D P, Welling M. Auto-encoding variational Bayes. *arXiv preprint arXiv: 1312.6114*, 2013.
- [29] Rezende D J, Mohamed S, Wierstra D. Stochastic backpropagation and variational inference in deep latent gaussian models. *International Conference on Machine Learning*, 2014, 2: 2.
- [30] Tian Y L, Krishnan D, Isola P. Contrastive multiview coding. *European Conference on Computer Vision*, 2020: 776–794.
- [31] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017, 30. Doi: 10.48550/arXiv.1706.03762.
- [32] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020.
- [33] Chen X L, Xie S N, He K M. An empirical study of training self-supervised visual transformers. *arXiv e-prints*, 2021: arXiv–2104.
- [34] Caron M, Touvron H, Misra I, et al. Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021: 9650–9660.
- [35] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv: 1412.6572*, 2014.
- [36] Yu J L, Yin H Z, Xia X, et al. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022: 1294–1303.
- [37] Maji S, Rahtu E, Kannala J, et al. Fine-grained visual classification of aircraft. *arXiv preprint arXiv: 1306.5151*, 2013.
- [38] Fei-Fei L, Fergus R, Perona P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, 2004: 178–178.
- [39] Krause J, Deng J, Stark M, et al. Collecting a large-scale dataset of fine-grained cars. 2013.

- [40] Krizhevsky A. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 2009, 1(4).
- [41] Cimpoi M, Maji S, Kokkinos I, et al. Describing textures in the wild. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014: 3606–3613.
- [42] Nilsback M E, Zisserman A. Automated flower classification over a large number of classes. 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, 2008: 722–729.
- [43] Bossard L, Guillaumin M, Gool L V. Food-101—mining discriminative components with random forests. *European Conference on Computer Vision*, 2014: 446–461.
- [44] Parkhi O M, Vedaldi A, Zisserman A, et al. Cats and dogs. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012: 3498–3505.
- [45] Xiao J X, Hays J, Ehinger K A, et al. Sun database: Large-scale scene recognition from abbey to zoo. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010: 3485–3492.
- [46] Everingham M, Gool L V, Williams C K I, et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010, 88(2): 303–338.
- [47] Ericsson L, Gouk H, Hospedales T M. How well do self-supervised models transfer? *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021: 5414–5423.
- [48] Kolesnikov A, Zhai X H, Beyer L. Revisiting self-supervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019: 1920–1929.
- [49] Kornblith S, Shlens J, Le Q V. Do better imagenet models transfer better? *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019: 2661–2671.
- [50] Liu D C, Nocedal J. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 1989, 45(1): 503–528.