CaCo: Both Positive and Negative Samples are Directly Learnable via Cooperative-Adversarial Contrastive Learning

Xiao Wang ©, Graduate Student Member, IEEE, Yuhang Huang ©, Dan Zeng ©, Senior Member, IEEE, and Guo-Jun Qi ©, Fellow, IEEE

Abstract—As a representative self-supervised method, contrastive learning has achieved great successes in unsupervised training of representations. It trains an encoder by distinguishing positive samples from negative ones given query anchors. These positive and negative samples play critical roles in defining the objective to learn the discriminative encoder, avoiding it from learning trivial features. While existing methods heuristically choose these samples, we present a principled method where both positive and negative samples are directly learnable end-to-end with the encoder. We show that the positive and negative samples can be cooperatively and adversarially learned by minimizing and maximizing the contrastive loss, respectively. This yields cooperative positives and adversarial negatives with respect to the encoder, which are updated to continuously track the learned representation of the query anchors over mini-batches. The proposed method achieves 71.3% and 75.3% in top-1 accuracy respectively over 200 and 800 epochs of pre-training ResNet-50 backbone on ImageNet1K without tricks such as multi-crop or stronger augmentations. With Multi-Crop, it can be further boosted into 75.7%. The source code and pre-trained model are released in https://github.com/maple-research-lab/caco.

Index Terms—Contrastive learning, cooperative-adversarial learning, self-supervised learning, positive-negative samples.

I. INTRODUCTION

Unsupervised representation learning [1], [2], [3] has attracted tremendous attentions recently as model pre-training becomes an essential step for deep neural networks [4]. Among them are a family of instance discrimination-based methods [5], [6], [7], [8], which train the network by distinguishing positive samples from their negative counterparts given query anchors from mini-batches during the learning process. Contrastive learning [1] and many variants [9], [10] are one of the most popular directions that achieved great success in last 3 years by pulling the embeddings of positive pairs together and pushing that of negative pairs away. In these methods, a critical question arises regarding how to find the most critical positives and negatives that can provide discriminative knowledge to self-supervise the network training.

SimCLR [6] and MoCo [7], [11] are two representative contrastive learning methods in this category. Given a query anchor, SimCLR uses the other samples from the same mini-batch as negative examples while an augmented view of the anchor is used as the positive sample. On the contrary, the MoCo maintains a memory bank as in [5], and uses the negative samples from the memory bank. The memory bank is built by queueing the learned representations over past mini-batches in a FIFO (First-In-First-Out) fashion, and a momentum update is applied to the key network whose weights are an exponential moving average of the query networks over time. SimCLR [6] usually needs a very large batch size to involve sufficient negative samples in each iteration, making it very computationally and memory demanding. Although the MoCo also has a very large memory bank, it can be maintained efficiently as the negative samples in it are collected on-the-fly from the past mini-batches.

Both methods show that successful training strongly relies on the construction of negative samples. Usually, harder negatives that are closer and mixed with the query anchors contain more discriminative information, making it harder to learn shallow features providing shortcut solutions to distinguish between positives and negatives. Along this direction, recent works [12] use heuristic sampling by concentrating on the hard negatives around queries.

On the contrary, a recent breakthrough demonstrates that negative samples are directly learnable by treating them as a part of network weights through adversarial contrastive learning [8], naturally resulting in hard negatives pushed towards query anchors. It provides a more principled way to learn rather than heuristically construct discriminative negatives. Specifically, by treating the negatives and the encoder as two adversarial players, the AdCo outperforms the SimCLR and MoCo in terms of both accuracy and efficiency. However, a question still remains — can we learn positive samples as well just like those negative samples? In this paper, we strive to answer it by revealing the relation between positive and negative samples via a cooperative-adversarial principle.
Unlike negative samples, we will show that given a query anchor, its positive sample ought to be learned cooperatively rather than adversarially with the encoder network. This is based on the observation that the representation of a positive sample that cooperatively minimizes the contrastive loss with the encoder will be pulled towards the given anchor, instead of being pushed away if it were adversarially learned by maximizing the loss. The cooperative positive sample will also be chosen from the memory bank shared with the negative samples. This could mitigate false negative problem as memory bank samples in close proximity of a query anchor can be trained as positives, instead of being blindly treated as negatives. Both positive and negative samples will be end-to-end trained together with the encoder through such a Cooperative-adversarial Contrastive (CaCo) learning approach. Moreover, we propose to use the most probable positive samples that are stable to update for an end-to-end training of the CaCo model. Experiment results demonstrate the proposed method achieves 71.3% and 75.3% in top-1 accuracy over 200 epochs and 800 epochs of pre-training on Imagenet1K without multi-crop augmentations. It can achieve 75.7% in top-1 accuracy after multi-crop augmentations with 800 epochs pre-training.

The remainder of the paper is organized as follows. We will review the related works in Section II, and revisit the preliminary works on contrastive learning in Section III. The proposed CaCo method is presented in Section IV, followed by the experiments in Section V. We will conclude the paper in Section VI.

II. RELATED WORK

In this section, we will review the related works on contrastive learning from three perspectives - instance discrimination-based contrastive learning, hard negative samples, and trainable memory bank.

Instance Discrimination. Most of recent works on contrastive learning focus on instance discrimination-based methods [5]. Given a query anchor from a training minibatch, it explicitly constructs the positive and negative samples, and self-trains the representation network (i.e., an encoder) by distinguishing positives from negatives. It minimizes the InfoNCE loss (a form of the contrastive loss) that can be implemented as a cross entropy by treating each instance as a positive or a negative class. The positive sample of a query anchor is often given as the augmented view of the same anchor. Depending on different ways to form negatives, the contrastive learning has many variants. Among them are the SimCLR [6] where negatives come from the other samples of the same mini-batch, as well as the MoCo [7], [11] where exists a memory bank storing the representations over past mini-batches as negatives. The former usually relies on a large size of mini-batches to perform reliable training of the encoder, which is demanding for computing and memory. MoCo addresses this problem by reusing the obtained representation of past samples without having to re-compute them for each mini-batches. While some negative samples may belong to the same class as a query anchor in instance discrimination methods (i.e., false negatives), a nearest neighbor contrastive learning extends the SimCLR and the MoCo by instead assigning the negative most similar to the anchor as its positive [13], which somehow mitigates the false negative problem.

Hard Negative Samples. It has been observed that the harder negative samples that are difficult to distinguish from their positive counterparts play a more critical role in contrastive learning. They prevent the encoder from learning trivial features to shortcut the aforementioned instance discrimination task. There exist methods based on heuristically sampling the samples in proximity of positive anchors. For example, hard negative sampling methods [12], [14] apply importance sampling to find harder negatives with a concentration parameter. Recent works [15], [16] on adversarial pre-training also shed some light on learning representations resilient against adversarial attacks. However, they do not directly train negative samples end-to-end by treating memory bank as learnable network weights as in [8]. Instead, they apply the adversarial training to input images or augmented views that may adversely affect positive samples that ought to be constructed cooperatively as we find in this paper.

Trainable Memory Bank. The recent work on Adversarial Contrastive learning (AdCo) [8] made a significant progress to demonstrate that the hard negatives are directly learnable as a part of network weights that are adversaries to the encoder network in the other part. Instead of minimizing the InfoNCE loss as the encoder, it maximizes it that pushes the learned negatives towards their query anchors. It provides a principled way to finding more discriminative negatives for an effective training of the encoder. Along this line, an implicit feature modification (IFM) [17] is proposed to iteratively update the hard samples. The IFM also pushes the samples towards the representation of anchors like the AdCo, but its step size is set by a given level of budget. Instead, the iterative update direction of these samples in the AdCo is derived from the positive gradient of the InfoNCE loss as an adversary learner, and its step size is proportional to the probability of being a positive to a query anchor. In this way, the AdCo can produce more discriminative negatives by making it harder to distinguish them from the positive anchors.

In this paper, along the road outlined from instance discrimination and trainable memory bank, we will unify the joint training of both positive and negative samples given query anchors. Both positives and negatives will share a memory bank that is trained end-to-end. For this purpose, it needs to answer two critical questions in contrastive learning:

- Can we end-to-end train positive and negative samples jointly through a shared memory bank?
- If so, will adversarial criterion still work for training positives?

The answer to these questions leads to the proposed Cooperative-adversarial Contrastive (CaCo) learning presented in this paper.

III. PRELIMINARIES AND NOTATIONS

In this section, we revisit the contrastive learning and its adversarial training of memory bank samples. We will also define several notations that will be used later.
A. Contrastive Learning

Consider an unlabeled example \( x \) called a query anchor in the current training mini-batch and its representation \( z \) output from an encoder network \( z \triangleq \phi_{\theta}(x) \) parameterized by \( \theta \). To train the encoder \( \phi_{\theta} \), a positive sample \( x^+ \) of \( x \) is obtained by applying an augmentation (e.g., random crop, color jittering and Gaussian noises) to \( x \). The resultant \( x^+ \) provides an alternate view of the original anchor \( x \), along with its representation \( z^+ \). Usually, in a typical contrastive learning approach, the anchor \( x \) is also given by applying a random augmentation to the original image.

In instance discrimination-based contrastive learning such as MoCo [7] and AdCo [8], together with mini-batches over iterations, a memory bank \( M \triangleq \{b_j|j=1,\ldots,K\} \) of \( K \) representations are also provided as negatives, in contrast to the positive representation \( z^+ \) for an anchor. The memory bank is shared across all unlabeled anchors in a minibatch. Such a bank can be constructed by queueing the obtained representations from the past minibatches in a FIFO (First-In-First-Out) fashion such as in MoCo [7]. Alternatively, a more principled way is to directly learn the memory bank by maximizing the contrastive loss (i.e., InfoNCE loss), which has demonstrated more efficient training of both positives and negatives. Meanwhile, it could also mitigate the false negative problem in contrastive learning, since it is known that some negative samples in the learned memory bank would be positive to an anchor if their representations are sufficiently close. This inspires us to develop the following Cooperative-adversarial Contrastive (CaCo) learning method.

The previous results [8] show that the memory bank and the encoder can be updated alternately, resulting in more efficient representation learning since the negative samples can be updated more efficiently to provide critical information to specify the contrastive learning.

Although the AdCo [8] has made a significant progress to directly learn negative samples, the question still remains regarding if the positive sample \( x^+ \) and its representation \( z^+ \) for an unlabeled anchor can also be learned directly. This will give rise to a more elegant solution by unifying the end-to-end training of both positives and negatives. Meanwhile, it could also mitigate the false negative problem in contrastive learning, since it is known that some negative samples in the learned memory bank would be positive to an anchor if their representations are sufficiently close. This inspires us to develop the following Cooperative-adversarial Contrastive (CaCo) learning method.

B. Adversarial Contrastive Learning

Constructing the memory bank [5], [11], [18] plays a critical role in contrastive learning. The negative samples in it gives the important specification about how the underlying representation network \( \phi_{\theta} \) ought to be learned by discriminating between positives and negatives. Previous results show that hard negatives could provide more useful information, since hard negative samples can prevent the encoder from learning trivial representations focusing on low-level features such as local details rather than high-level semantic structures such as object shapes and categories.

Furthermore, it has been shown that hard negative samples naturally result from an adversarial contrastive learning (AdCo) approach [8], with a novel idea of directly learning the negatives to form a memory bank. This is contrary to the MoCo where the negative samples are collected by queueing the representations from past minibatches heuristically. Formally, the AdCo considers two adversary players – the memory bank \( M \) and the representation encoder \( \phi_{\theta} \). They are trained in a mutually adversarial manner by

\[
\ell(x) = -\log \frac{\exp(z \cdot z^+/\tau)}{\exp(z \cdot z^+/\tau) + \sum_{j=1}^{K} \exp(z \cdot b_j/\tau)}
\]

where \( \cdot \) denotes the inner product between two feature vectors and \( \tau \) is the temperature controlling the sharpness of the loss. SimCLR [11] also symmetrizes the loss by switching \( z \) and \( z^+ \) with the latter as a query anchor instead. Since the representations are normalized to a unit norm, the inner product gives the cosine similarity between samples.

\[
M^*, \theta^* = \arg \max_{b_j \in M} \min_{\theta} \sum_{x \in B} \ell(x)
\] (2)

The IV. CACO: COOPERATIVE-ADVERSARIAL CONTRASTIVE LEARNING

In this section, we will present the proposed Cooperative-adversarial Contrastive (CaCo) Learning. First we will explicitly integrate positive samples shared with negatives from the memory bank into the contrastive learning in Section IV-A. Then in Section IV-B we will show that positives and negatives ought to play cooperative and adversarial role in the network training, respectively, followed by the elaboration on cooperative-adversarial training over minibatches in Section IV-C.

A. Positives From Shared Memory Bank

Following the definition defined in the last section, consider the representation \( z \) of an unlabeled anchor \( x \). In the AdCo, we assume its negative samples all come from the trainable memory bank \( M \). Now let us make a bold step by assuming the corresponding positive representation also comes from \( M \). We denote the index of the positive sample in \( M \) by \( j^+ \in K \triangleq 1, \ldots, K \). Thus, for \( x \) with embedding \( z \), its positive representation is given by \( b_{j^+} \in M \). Accordingly, the contrastive InfoNCE loss becomes

\[
\ell(x) \triangleq -\log \frac{\exp(z \cdot b_{j^+}/\tau)}{\exp(z \cdot b_{j^+}/\tau) + \sum_{j \neq j^+} \exp(z \cdot b_j/\tau)} = -\log \frac{\exp(z \cdot b_{j^+}/\tau)}{\sum_{j=1}^{K} \exp(z \cdot b_j/\tau)}
\] (3)
which are for the positive and negative samples, respectively. Here, \( p(b_j | z) \) denotes the probability of a memory bank sample \( b_j \) being positive to the input representation \( z \),

\[
p(b_j | z) = \sum_{k=1}^{K} \exp(z \cdot b_{j+k} / \tau)
\]

Based on the idea behind the adversarial training, the negative samples can thus be updated by maximizing \( \ell(x) \), resulting in

\[
b_j \leftarrow b_j + \eta \left[ 1 - p(b_j | z) \right] z
\]

where \( \eta \) is the learning rate for the gradient update. This shows that each negative representation \( b_j \) will be updated towards the anchor representation \( z \), which will make it harder to distinguish the negative sample from the anchor \( z \). The previous results show that such harder negatives can provide more useful information to train the encoder. In this sense, the negative samples are adversarial players to the encoder. For the adversarial negative example training, AdCo [8] has carefully discussed it and utilized it for contrastive learning successfully.

Now the question is: should we also make the positive sample an adversary to the encoder? Let us take a look at the gradient of the positive representation \( b^+ \) in (4). If it were assumed to be an adversary, it would be updated by

\[
b_{j^+} \leftarrow b_{j^+} - \eta \left[ 1 - p(b_{j^+} | z) \right] z
\]

In this case, the positive \( b_{j^+} \) would be pushed away from the anchor, instead of being pushed towards it. This causes a dilemma – what we chose as a positive sample would eventually end up being pushed away from the anchor, making it more likely to be a negative rather than a positive sample. Obviously, this is not our intention to learn a positive sample.

This shows that the positive sample should be a cooperative player rather than an adversary to the encoder by minimizing the contrastive loss. This results in the following update to the positive sample

\[
b_{j^+} \leftarrow b_{j^+} + \frac{\eta}{\tau} \left[ 1 - p(b_{j^+} | z) \right] z
\]

which in turn pushes the positive representation towards the anchor, in line with our intuition to learn the positive sample close to the anchor.

### C. Cooperative-Adversarial Training

The above analysis yields the following Cooperative-Adversarial Contrastive (CaCo) objective

\[
\sum_{\mathbf{x} \in \mathcal{B}} \min_{b_{j^+}, \theta} \max_{b_j, j \neq j^+} - \log \frac{\exp(z \cdot b_{j^+} / \tau)}{\sum_{j=1}^{K} \exp(z \cdot b_j / \tau)}
\]

where both the positive sample \( b_{j^+} \) and the encoder with the parameters \( \theta \) minimize the loss, thus being cooperative players, while the negative samples \( b_j, j \neq j^+ \) are adversary instead to maximize it.

The above CaCo objective is defined over a mini-batch \( \mathcal{B} \). One can assign a distinct positive sample from the memory bank to each anchor \( z \) in \( \mathcal{B} \). In other words, the positive index \( j^+ \) is a function of \( z \) in (6). Thus, for a memory bank sample \( b_j \), we
use \( \mathcal{P}_j^+ (\mathcal{N}_j^-) \)'s, for which \( \mathbf{b}_j \) is a positive (negative resp.) sample. For each \( \mathbf{b}_j \), we have \( \mathcal{P}_j^+ \cap \mathcal{N}_j^- = \emptyset \) and \( \mathcal{P}_j^+ \cup \mathcal{N}_j^- = \mathcal{B} \). Then, according to (6), all memory bank samples can be updated by

\[
\mathbf{b}_j \leftarrow \mathbf{b}_j - \sum_{\mathbf{x} \in \mathcal{P}_j^+} \eta \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{b}_j} + \sum_{\mathbf{x} \in \mathcal{N}_j^-} \eta \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{b}_j} \tag{7}
\]

with a positive learning rate \( \eta \), where the minus “−” and the plus “+” in front of the last two terms reflect the cooperative and the adversarial updates to \( \mathbf{b}_j \).

**Effect of l2-Normalization on Memory Bank Updates.** It is worth noting that the gradient in (4) was derived by assuming both \( \mathbf{z} \) and all memory bank samples \( \mathbf{b}_j \)'s have already been normalized to a unit length. Even though they are normalized, the gradient update will result in new vectors not normalized to a unit length anymore. Thus, an l2-normalization is often required after the gradient update. Such a normalization can be built into (3) by replacing \( \mathbf{z} \) and \( \mathbf{b}_j \) with \( \mathbf{z}/\|\mathbf{z}\|_2 \) and \( \mathbf{b}_j/\|\mathbf{b}_j\|_2 \), respectively. This results in the following gradient

\[
\frac{\partial \ell(\mathbf{x})}{\partial \mathbf{b}_j} = -\frac{1}{\tau} \left[ 1 - p(\mathbf{b}_j | \mathbf{z}) \right] \frac{\|\mathbf{b}_j + \|\mathbf{z} - \mathbf{b}_j + \mathbf{b}_j^+ \|^2}{\|\mathbf{z}\|_2^2 \|\mathbf{b}_j + \mathbf{b}_j^+\|_2^2} \tag{7}
\]

This shows how to update the positive sample end-to-end by viewing its representation \( \mathbf{b}_j \)'s as free network weights. For simplicity, one can assume every time after a mini-batch update, \( \mathbf{b}_j \) and \( \mathbf{z} \) are normalized back to a unit length such that we always keep the current samples \( \|\mathbf{b}_j\|_2 = 1 \) and \( \|\mathbf{z}\|_2 = 1 \). In this case, the above gradient can be simplified to

\[
\frac{\partial \ell(\mathbf{x})}{\partial \mathbf{b}_j} = -\frac{1}{\tau} \left[ 1 - p(\mathbf{b}_j | \mathbf{z}) \right] (\mathbf{z} - \mathbf{b}_j + \mathbf{b}_j^+) = -\frac{1}{\tau} \left[ 1 - p(\mathbf{b}_j | \mathbf{z}) \right] (\mathbf{I} - \mathbf{b}_j + \mathbf{b}_j^+) \mathbf{z} \tag{8}
\]

Similarly, we can derive the gradient over negative samples by assuming all \( \mathbf{b}_j \)'s are normalized to a unit norm after each gradient update

\[
\frac{\partial \ell(\mathbf{x})}{\partial \mathbf{b}_j} = \frac{1}{\tau} p(\mathbf{b}_j | \mathbf{z})(\mathbf{I} - \mathbf{b}_j + \mathbf{b}_j^+) \mathbf{z}, j \neq j^+ \tag{9}
\]

From (8) and (9), we can see that both positive and negative samples will be updated towards the transformed direction \( \mathbf{T}_j \mathbf{z} \) with \( \mathbf{T}_j = \mathbf{I} - \mathbf{b}_j + \mathbf{b}_j^+ \). It is easy to verify that

- The gradient \( \mathbf{T}_j \mathbf{z} \) is orthogonal to \( \mathbf{b}_j \), i.e., \( \mathbf{b}_j^T \mathbf{T}_j \mathbf{z} = 0 \). It suggests that the gradient is tangent to the unit-hypersphere such that it tends to keep the unit norm of \( \mathbf{b}_j \) unchanged as expected.
- Along the gradient, the sample \( \mathbf{b}_j \) also tends to be pushed closer towards \( \mathbf{z} \) since their cosine similarity \( \mathbf{z}^T \mathbf{T}_j \mathbf{z} = 1 - (\mathbf{z} \cdot \mathbf{b}_j)^2 \geq 0 \) is nonnegative.

From this update, we observe that

- For cooperative positives, the more probable \( \mathbf{b}_j \) is positive to \( \mathbf{z} \) with a higher \( p(\mathbf{b}_j | \mathbf{z}) \), the less it will be pushed towards \( \mathbf{z} \). This is intuitive as a positive sample already closer to its anchor \( \mathbf{z} \) does not need to be updated too much;
- For adversarial negatives, we have an opposite observation – the more probable \( \mathbf{b}_j \) is positive to \( \mathbf{z} \), the more it should be pushed towards the anchor. In this way, it will become a harder negative to be distinguished from the positive counterparts, thereby providing more discriminative information to update the encoder.

**The Most Probable Positive.** The last question we need to answer is how we choose the positive sample given an anchor. Here, we propose to use the Most Probable Positive (MPP)
sample resulting in the minimum change that favors a stable update for the training purpose.

Specifically, once a positive sample \( b_{ij}^+ \) is chosen for an anchor \( z \), its representation will be updated along the (negative) gradient based on (8). Ideally, we expect that the updated sample will still be positive to the anchor. This ensures the positive assignments in \( P_j^+ \) (and thus \( N^+_j \) as well) to be as stable as possible so that they can be updated stably based on (10).

For this reason, we note that \( 1 - p(b_j^+ | z) \) is the step size (up to a constant coefficient \( \frac{1}{2} \)) to update the positive sample \( b_{ij}^+ \) in (8). To minimize the change to the positive sample, we choose the positive sample that minimizes this step size,

\[
j^+ = \arg \min_j 1 - p(b_j^+ | z) = \arg \max_j p(b_j^+ | z),
\]

which results in the Most Probable Positive (MPP) sample to the anchor \( z \). Intuitively, the chosen positive \( b_{ij}^+ \) will be the one that minimizes the error in assigning a false positive sample to \( z \) based on \( p(b_j^+ | z) \) in (5).

To address the most probable positive clearly, the diagram of the training process is illustrated in Fig. 2. For a fair comparison with the SOTA methods such as MoCo [7] and AdCo [8], the CaCo also has two streams composed of a momentum encoder and a query encoder, where the momentum encoder is used to identify the positives in the memory bank, and the contrastive loss will be propagated to query encoder and memory bank to train them. All positive and negative samples are treated as free parameters like the network weights, and they are directly learned through gradient descent/ascent by back-propagating the contrastive loss in the proposed cooperative-adversarial fashion, as shown in (10).

**Avoidance of Collapse.** To explain the avoidance of collapse, it is worth noting that both positive and negative samples share the memory bank across query anchors in a mini-batch, and they are dynamically updated over iterations in the proposed cooperative-adversarial training. In other words, a memory bank sample \( b \) can be positive for one query anchor, while being negative for another one, which keeps the memory bank from collapsing to a single trivial point. Memory bank samples will also be updated over iterations, and different positives will be assigned to the feature representation \( z \) of a sample, keeping the latter from collapsing to a prefixed point \( b^+ \).

V. Experiments

In this section, we evaluate the proposed CaCo and compare the results with the other unsupervised models including the state-of-the-art contrastive learning methods.

A. Implementation Details

For a fair comparison with the existing models [6], [7], we adopt the ResNet-50 as the backbone for unsupervised pre-training on ImageNet. The output feature map from the top ResNet-50 block is average-pooled and projects to a 256-D feature vector through three MLP layers with two 2048-D hidden layers and the ReLU [6]. The resultant vector is \( \ell_2 \) normalized to calculate the cosine similarity. We adopt the single-crop protocol used in [6], [7], [11] to augment data in single-crop experiments and we used multi-crop augmentations used in literature [23] to apply multi-crop in final experiments that can further boost performances.

We adopt up to 4,096 images in a mini-batch for pre-training across up to four 8-Nvidia Tesla V100 GPU servers for single-crop. For the multi-crop setting, based on the same number of four 8-Nvidia Tesla V100 GPU servers, we adopt a batch size of 2,048 images for pre-training. We will show that the proposed CaCo could also achieve competitive results with smaller batch sizes. By convention, the number of negative adversaries is set to 65,536 to make a fair comparison with the other methods [7], [8].

For the network pre-training, we adopt an initial learning rate of \( 0.03 \times \text{batch}_{ize}/256 \) and \( 0.3 \times \text{batch}_{ize}/256 \) in the SGD [30] and LARS optimizer [31] for updating the backbone network for smaller (< 1,024) and larger (\( \geq 1,024 \)) batch sizes, respectively, along with a weight decay of \( 10^{-4} \) and a momentum of 0.9. The memory bank is always directly optimized by SGD optimizer with learning rate 3.0 without weight decay and a momentum of 0.9. The cosine scheduler [32] is used to gradually decay the learning rates. We set the same temperature \( \tau \) to 0.08 when updating the memory bank and network backbone. After the backbone network is initialized, its output feature vectors over randomly drawn training images are used to initialize the memory bank. After that, the encoder and the shared memory bank with both positives and negatives are alternately updated.

B. Results on Imagenet1K

First, we perform a linear evaluation by fine-tuning a fully connected classifier for 100 epochs on top of the frozen 2048-D feature vector from the ResNet-50 backbone. Table I reports the results on Imagenet1K after 200 epochs of pre-training. With different batch sizes of 256, 1024 and 4,096 for network

| Method     | batch size | top-1 |
|------------|------------|-------|
| InstDisc [5]        | 256        | 58.5  |
| SimCLR [6]          | 256        | 61.9  |
| CPC v2 [9]          | 512        | 63.8  |
| PCL v2 [19]         | 256        | 67.6  |
| MoCo v2 [11]        | 256        | 67.5  |
| PIC [20]            | 512        | 67.6  |
| InfoMin Aug [21]    | 256        | 70.1  |
| SimSiam [22]        | 256        | 70.0  |
| SwAV [23]           | 4096       | 69.1  |
| NNCLR [13]          | 4096       | 70.7  |
| BYOL [24]           | 4096       | 70.6  |
| AdCo [8]            | 256        | 68.2  |
| CaCo                 | 256        | 70.9  |
| CaCo                 | 1024       | 71.3  |
| CaCo                 | 4096       | 72.0  |
pre-training, CaCo even outperforms the state-of-the-art SWAV, BYOL and NNCLR with a much larger batch size of 4096 when we used small batch-size. It is well known that the performances of these SOTA models improve with larger batch sizes [13], [23], [24]. However, larger batch sizes require multiple GPU servers with at least 32 Nvidia V100 GPU cards to accommodate 4096 images. This makes the network pre-training unaffordable and inconvenient to set up across multiple servers for most of research teams. The proposed CaCo not only outperforms SOTA methods with much smaller batch sizes, but also allows affordable network pre-training on a single GPU server. Our training time is on par with AdCo [8], MoCo [18] as shown in Table IV.

For single-crop settings, we also report the results with more than 200 pre-training epochs in Table II. Here we run the CaCo with a batch size of 1,024 and 4,096 images to compare with the other methods. Here 1,024 is the largest size that can fit the GPU memory of a single $8 \times V$100 server. It still outperforms the SOTA models with various batch sizes and epochs. This makes the CaCo affordable and competitive for network pre-training. It is worth noting that here we do not apply any tricks such as multi-crop augmentations [23], [33] and stronger augmentations [34] that have shown effective in improving the accuracy in unsupervised learning literature. This shows the CaCo is an elegant method easier to implement without relying on those tricks.

To compare with the methods using multi-crop augmentations [23], we have conducted the experiments following the same protocol used in SwAV [23]. The results are shown in Table III. We outperformed the other methods in this setting even with a smaller batch size of 2,048 based on the same number of four 8-Nvidia Tesla V100 GPU servers. We believe the performance 75.7% can be further boosted if we can increase the batch size to 4,096 as used in other methods. However, this requires doubling computing resources, which makes it unaffordable for many research groups.

### Table II

| Method   | epochs | batch size | top-1  |
|----------|--------|------------|--------|
| ScLa [25] | 400    | 256        | 61.5   |
| PiRL [26] | 800    | 1024       | 63.6   |
| CMC [10]  | 240    | 128        | 66.2   |
| SimCLR [6] | 800    | 4096       | 69.3   |
| PIC [20]  | 1600   | 512        | 70.8   |
| MoCo v2 [11] | 800    | 256        | 71.1   |
| InfoMin Aug [21] | 800    | 256        | 73.0   |
| SimSiam [22] | 800    | 256        | 71.3   |
| SwAV [23] | 800    | 4096       | 71.8   |
| NNCLR [13] | 800    | 1024       | 72.9   |
| BYOL [24] | 1000   | 4096       | 74.3   |
| AdCo [8]  | 800    | 256        | 72.8   |
| MoCo v3 [27] | 1000   | 4096       | 74.6   |
| NNCLR [13] | 800    | 4096       | 74.9   |
| CaCo     | 800    | 1024       | 74.1   |
| CaCo     | 800    | 4096       | 75.3   |

### Table III

| Method   | epochs | batch size | top-1  |
|----------|--------|------------|--------|
| SwAV [23] | 800    | 4096       | 75.3   |
| DINO [28] | 800    | 4096       | 75.3   |
| NNCLR [13] | 800    | 4096       | 75.6   |
| CaCo     | 800    | 2048       | 75.7   |

### Table IV

| Method   | Epoch | Time(s) | GPU   | GPU Time /epoch |
|----------|-------|---------|-------|-----------------|
| MoCo v2 [11] | 200 | 53.0    | 8xV100 | 2.12            |
| BYOL [24]    | 1000 | 8.0     | 512xTPU | 4.10            |
| SwAV* [23]   | 800  | 50      | 64xV100 | 4.06            |
| AdCo [8]     | 200  | 56.5    | 8xV100  | 2.26            |
| CaCo         | 200  | 56      | 8xV100  | 2.24            |

### Table V

| Method   | batch size | Top-1 |
|----------|------------|-------|
| MoCo-v1 [18] | 256     | 71.1  |
| MoCo-v2 [11] | 256     | 75.6  |
| SimCLR [6]  | 4096     | 74.3  |
| SimSiam [22] | 256     | 77.1  |
| InfoMin [21] | 256     | 81.9  |
| SwAV [23]   | 4096     | 82.1  |
| CaCo        | 4096     | 82.7  |

### C. Results on ImageNet100

Like in experiments on ImageNet1K dataset, we also pre-train over 200 epochs on ImageNet100 with the batch size of 4096. Then we perform a linear evaluation by fine-tuning a fully connected classifier for 100 epochs on top of the frozen 2048-D feature vector from the ResNet-50 backbone. CaCo can achieve 82.7% Top-1 accuracy. In contrast to the previous best model SwAV [23] using multi-crop augmentation, CaCo does not increase the computation time compared to the baseline MoCo. Meanwhile, since our memory bank includes both positives and negatives, the number of categories may influence our training process. This experiment shows that CaCo is generalizable under such a condition. In Fig. 4, we observe that CaCo can converge even faster in early stages on ImageNet100.

### D. Transfer Learning Results

We also perform transfer learning tasks on various datasets.\footnote{The link to Birdsnap dataset is broken or refers to a dataset with missing data, making it difficult to make a direct comparison.} The ResNet50 backbone was pretrained on Imagenet1K over...
800 epochs with a batch size of 4096. By following the linear evaluation protocol in [24], a fully connected layer of classifier is trained upon the frozen backbone.

Table VI reports the results under the linear evaluation compared with the other models. The results show the CaCo performs the best on 8 out of these 11 datasets among the compared self-supervised methods. This demonstrates the CaCo representation pre-trained on the ImageNet1K dataset can well generalize to downstream tasks on a variety of datasets.

For object detection, we evaluate the pre-trained network on VOC and COCO datasets used in the previous methods [6], [18]. On the VOC dataset [35], we train the detection head with VOC07+12 trainval dataset and test on VOC07 test dataset. On the COCO dataset [36], we fine-tune the network on the train2017 set with 118 k images and evaluate on the val2017. For the sake of a fair comparison, the object detection tasks are implemented by detectron2 [37] based on the pre-trained ResNet-50. As shown in Table VII, the performances on two datasets are much better than the supervised model trained on ImageNet. That suggests that CaCo has better generalization ability in downstream tasks. Meanwhile, compared to self-supervised methods, we also improve a lot with 83.5% AP50 on VOC dataset and 42.6% AP on COCO dataset, while previous best performance is 83.0% and 42.2%, respectively.

### E. Ablation Studies

In this section, we present an ablation study of CaCo. Here we focus on five most factors: the size of the memory bank (the sum of the positives and negatives), the contribution of positives and negatives to the performance, the impact of data augmentation, the importance of initialization of memory bank, the impact of pre-training dataset.

**Size of the Memory Banks.** We set up the experiments with batch-size 4096 with 200 epoch pre-training, which indicates the number of positive is always 4096. We only change the memory bank size across different experiments. As shown in Table VIII, we changed the memory size from 4 M to 12 M and observed the clear improvement of performance from 71.7% to 72.1%. That’s to say, CaCo can be benefited by the size increase of memory bank. This is reasonable since bigger memory bank size has two benefits: 1) more accurate positives by checking the most probable positive from a bigger memory bank; 2) the task become more challenge since encoder needs to select the correct positive out of a bigger pool including more negative pairs.

**Contribution of Positives and Negatives.** We conducted an ablation study to compare the contribution of positives and negatives. Here all experiments are based on a batch size of 1024 with 200 epoch pre-training. Here None setting is without memory bank, while both positive and negatives directly coming from the concurrent mini-batch. Here Positive setting is similar to that NNCLR [13], where we keep a queue to save the embeddings of previous batches and we select the nearest neighbor of query in memory bank as positive while negatives are from the concurrent mini-batch. Here Negative setting is similar to that AdCo [8], where the memory bank is adversarially trained and serves as the negatives, and the positives are from the current mini-batch. Here P+N setting is our CaCo setting, where both positives and negatives are from the memory bank and memory bank is collaboratively and adversarially trained by the contrastive loss in an end-to-end fashion. We keep all other settings to the same to have a fair comparison. The results are shown in Table IX. It’s clear that CaCo clearly improved compared to using positive or negative alone.

**Impact of Data Augmentation.** We compare MoCo-v2 and CaCo under different types of input data augmentations in Table X. As common practice in MoCo-v2 [11] and SimCLR [6], the most widely used data augmentation pipeline in contrastive learning is as follows: an image is first cropped from an input image and resized to 224 × 224 pixels. Then random color jittering, Gaussian Blur, grayscale conversion, horizontal flip, channel-wise color normalization are sequentially applied to generate the augmented images. For Single Crop (Without Color Jittering etc.), we removed the color jittering, gaussian blur and grayscale conversion as a simplified augmentation. Under
TABLE VIII
TOP-1 ACCURACY UNDER THE LINEAR EVALUATION ON IMAGENET WITH THE RESNET-50 BACKBONE. THE TABLE COMPARES THE METHODS OVER 200 EPOCHS OF PRETRAINING. # PARAMETERS: THE PARAMETER COUNTS OF NEGATIVE SAMPLES, AS TO BYOL AND SIMSIM, IT MEANS THE PARAMETER COUNTS OF THE PREDICTOR

| Method       | Symmetric Loss | Batch Size | #Negative Samples | #Params. | Top-1 (GPU · Time) /epoch |
|--------------|----------------|------------|-------------------|----------|--------------------------|
| SimCLR [6]   | ✓              | 8192       | -                 | -        | 67.0                     | 1.92 |
| MoCo v2 [11] | ✓              | 256        | 65536             | 8M       | 70.2                     | 3.34 |
| BYOL [24]    | ✓              | 4096       | -                 | -        | 70.6                     | 4.10 |
| SimSiam [22] | ✓              | 256        | 65536             | 8M       | 70.0                     | -     |
| AdCo [8]     | ✓              | 256        | 65536             | 2M       | 70.0                     | 3.50 |
| AdCo [8]     | ✓              | 256        | 16384             | 2M       | 70.0                     | 3.46 |
| AdCo [8]     | ✓              | 256        | 8192              | 1M       | 70.2                     | 3.45 |
| CaCo         | ✓              | 4096       | 37268             | 4M       | 71.7                     | 3.83 |
| CaCo         | ✓              | 4096       | 65536             | 8M       | 72.0                     | 3.84 |
| CaCo         | ✓              | 4096       | 98304             | 12M      | 72.1                     | 3.86 |

(a) AdCo/CaCo initialization

(b) Random Embedding Initialization

Fig. 3. Analysis of different initialization methods for CaCo. The top panel of each subfigure is the plot of Mean Maximum Positive Probability and NN accuracy (based on 20% of training set) over 200 epochs of pretraining under different settings. The second panel shows the percentage of category-space change in the memory bank over 200 epochs of pretraining. The third, forth and fifth panel show the histogram of category-space of the memory bank for epoch 1, epoch 10 and epoch 200, respectively. The first row shows the default AdCo initialization and the second row shows the random embedding initialization.

TABLE IX
TOP-1 ACCURACY UNDER THE LINEAR EVALUATION ON IMAGENET WITH THE RESNET-50 BACKBONE. HERE NONE SETTING DENOTES BOTH POSITIVE AND NEGATIVE ARE FROM CURRENT MINI-BATCH; POSITIVE SETTING DENOTES THE POSITIVE EMBEDDING IS FROM MEMORY BANK AND THE NEGATIVE EMBEDDING IS FROM THE CURRENT MINI-BATCH; NEGATIVE SETTING SUGGESTS THE POSITIVE EMBEDDING IS FROM CURRENT MINI-BATCH AND THE NEGATIVE EMBEDDING IS FROM THE MEMORY BANK; P+x IS OUR CACO SETTING, WHERE BOTH POSITIVE AND NEGATIVE ARE FROM THE MEMORY BANK. HERE THE #POSITIVE POOL AND #NEGATIVE POOL SUGGESTS THE TOTAL NUMBER OF EMBEDDINGS THAT WE SELECTED FROM TO SERVE AS POSITIVE AND NEGATIVES, RESPECTIVELY

| Setting     | Batch Size | #Positive Pool | #Negative Pool | Top-1  |
|-------------|------------|----------------|----------------|--------|
| None        | 1024       | 1024           | 1024           | 70.2   |
| Positive    | 1024       | 65536          | 1024           | 70.7   |
| Negative    | 1024       | 1024           | 65536          | 70.6   |
| Positive-Negative | 1024 | 65536          | 65536          | 71.3   |

TABLE X
TOP-1 ACCURACY UNDER THE LINEAR EVALUATION ON IMAGENET1K WITH THE PRETRAINED RESNET-50 BACKBONE. THE TABLE COMPARES DIFFERENT METHODS (MOCO-V2 VERSUS CA CO) USING DIFFERENT TYPES OF AUGMENTATIONS WHEN BEING PRE-TRAINED OVER 200 EPOCHS

| Augmentation          | MoCo-v2 [11] | CaCo   |
|-----------------------|--------------|--------|
| Single-Crop (Without Color Jittering etc.) | 54.8         | 58.2   |
| Single-Crop           | 67.5         | 72.0   |
| Multi-Crop            | 69.7         | 73.1   |

Representations to distinguish different instances. Also, from the results, we find that CaCo has higher top-1 accuracy of 58.2% than that of 54.8% for MoCo-v2. This suggests that CaCo can learn more effectively through the cooperative-adversarial learning of memory bank. For default augmentation pipeline, CaCo improved the performance from 67.5% to 72.0% by introducing a cooperative-adversarial memory bank. For multi-crop settings, we used the same multi-crop augmentations used in SwAV [23]. Here CaCo achieves 73.1% in top-1 accuracy with a large lead.
WANG et al.: CACO: BOTH POSITIVE AND NEGATIVE SAMPLES ARE DIRECTLY LEARNABLE VIA COOPERATIVE-ADVERSARIAL

Fig. 4. Analysis of CaCo performances on ImageNet-1K and ImageNet-100 datasets, both based on AdCo initialization. The first panel of each subfigure is the plot of Mean Maximum Positive Probability and NN accuracy (based on 20% of training set) over 200 epochs of pretraining under different settings. The second panel shows the percentage of category-space change in the memory bank over 200 epochs of pretraining. The third, forth and fifth panel show the histogram of classes in the memory bank for epoch 1, epoch 10 and epoch 200, respectively.

Fig. 5. The surrogate images of the Most Probable Positive (MPP) and random negatives from the learned memory bank for some query examples.

The embeddings of some images by a randomly initialized encoder (ResNet-50) as the memory bank’s initial embeddings; 3) AdCo initialization (default): we push the embeddings of a few images into the memory bank until it gets full with the encoder being continuously updated by minimizing the contrastive loss. The results are shown in Table XI. For the first setting, we find that it can not converge in a reasonable number of epochs. That may be because the most probable positive in memory bank is totally random at the beginning. For random embedding settings, the performance decreases compared to that of AdCo initialization.

To study the update and impact of memory bank during training process, we proposed several metrics to closely monitor it. 1) Mean Maximum Positive Probability: Given a query, the maximum probability corresponding to the most probable positive in (11) measures the likelihood of obtaining a true positive by the model. Therefore, the mean maximum positive probability reflects the overall confidence of the retrieved positives from the memory bank. 2) NN Accuracy: We also measure the nearest neighbor (NN) accuracy on validation dataset based on embedding similarity comparison with 20% training set on the dataset. All the embeddings are real-time calculation based on the pre-trained encoder of current epoch. 3) Category-Space of Memory Bank: We map the memory bank embeddings to the category-space to better track its update during training. Each

| Initialization          | Top-1 |
|-------------------------|-------|
| Random                  | Not converge |
| Random Embedding        | 70.3  |
| AdCo Init (default)     | 72.0  |

TABLE XI

TOP-1 ACCURACY UNDER THE LINEAR EVALUATION ON IMAGENET1K WITH THE RESNET-50 BACKBONE. THE TABLE COMPARES CACO UNDER DIFFERENT INITIALIZATION AND PRE-TRAINED WITH 200 EPOCHS
embedding in the memory back is assigned a class label based on its nearest neighbor in the validation set by measuring the similarity between embeddings. The category-space change during training process can help better directly reflect the update of memory bank. The percentage of category-space change reflects the percentage of how many memory bank samples change their class labels over consecutive epochs.

For random embedding initialization, as illustrated in Fig. 3, its converged performance in NN accuracy is as high as that of AdCo initialization, which reflects the continuous improvement in embedding representations. By analyzing the mean maximum probability and category-space change of embeddings in the memory bank, we find that the positive counterpart in memory bank is not as stable as in AdCo initialization at the beginning. After 30 epochs, it starts to stabilize and finally converge well. This is probably because the AdCo initialization constructs a memory bank more smoothly than other initialization since it keeps updating the underlying encoder while pushing initial embeddings into the memory bank. For category space distribution, we observe category-space distribution of memory bank come to more and more uniform through training. However, the pace of random embedding initialization is a little slower because the unstable positives in the initial stage of training. With training goes on, the percentage of category-space change consistently decreased along training. Second, the distribution of category-space turn into more uniform distribution through training. All observations verify the reliability and generalizability of CaCo on different pre-training datasets.

VI. CONCLUSION

This paper presents Cooperative-adversarial Contrastive (CaCo) learning for network pre-training. It directly learns both

**Fig. 6. t-SNE visualization of learned representations on CIFAR10, PETS and VOC2017 datasets, where different colors represent different classes on these datasets. The encoder is pre-trained by CaCo over 800 epochs on Imagenet1K.**
positives and negatives over a shared memory bank as cooperators and adversaries to an encoder by minimizing and maximizing the underlying contrastive loss, respectively. This results in an end-to-end training of positives and negatives iteratively updated towards query anchors along the directions tangent to the unit hypersphere. Jointly trained with the encoder, unsupervised features are learned to distinguish the learned adversarial negatives from the most probable cooperative positive given each query anchor. Experiments demonstrate that the CaCo outperforms the SOTA self-supervised methods on multiple downstream tasks without relying on multi-crop augmentations.

ACKNOWLEDGMENTS

Xiao Wang and Yuhang Huang implemented the idea and performed the experiments as remote research interns in the Seattle Research Center. Guo-Jun Qi conceived and formulated the idea as well as wrote the paper.

REFERENCES

[1] A. V. D. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.
[2] R. D. Hjelm et al., “Learning deep representations by mutual information estimation and maximization,” 2018, arXiv:1808.06670.
[3] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, “AET versus AED: Unsupervised representation learning by auto-encoding transformations rather than data,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2547–2555.
[4] G.-J. Qi and J. Luo, “Small data challenges in Big Data era: A survey of recent progress on unsupervised and semi-supervised methods,” 2019, arXiv:1903.11260.
[5] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3733–3742.
[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020, arXiv:2002.05709.
[7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” 2019, arXiv:1911.05722.
[8] Q. Hu, X. Wang, W. Hu, and G.-J. Qi, “AdCo: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1074–1083.
[9] O. J. Hénaff et al., “Data-efficient image recognition with contrastive predictive coding,” 2019, arXiv:1905.09272.
[10] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” 2019, arXiv:1906.05349.
[11] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” 2020, arXiv:2003.04297.
[12] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, “Hard negative mixing for contrastive learning,” 2020, arXiv:2010.01028.
[13] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, “With a little help from my friends: Nearest-neighbor contrastive learning of visual representations,” 2021, arXiv:2104.15458.
[14] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” 2020, arXiv:2010.04592.
[15] C.-H. Ho and N. Vasconcelos, “Contrastive learning with adversarial examples,” 2020, arXiv:2010.12050.
[16] M. Kim, J. Tack, and S. J. Hwang, “Adversarial self-supervised contrastive learning,” 2020, arXiv:2006.07589.
[17] J. Robinson, L. Sun, K. Yu, K. Batmanghelich, S. Jegelka, and S. Sra, “Can contrastive learning avoid shortcut solutions?,” 2021, arXiv:2106.11230.
[18] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1074–1083.
[19] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi, “Prototypical contrastive learning of unsupervised representations,” 2020, arXiv:2005.04966.
[20] Y. Cao, Z. Xie, B. Liu, Y. Lin, Z. Zhang, and H. Hu, “Parametric instance classification for unsupervised visual feature learning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 15614–15624.
[21] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, “What makes for good views for contrastive learning,” 2020, arXiv:2005.10243.
[22] X. Chen and K. He, “Exploring simple siamese representation learning,” 2020, arXiv:2011.10666.
[23] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” 2020, arXiv:2006.09882.
[24] J.-B. Grill et al., “Bootstrap your own latent: A new approach to self-supervised learning,” 2020, arXiv:2006.07733.
[25] Y. M. Asano, C. Rupprecht, and A. Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” 2019, arXiv:1911.05571.
[26] I. Misra and L. V. D. Maaten, “Self-supervised learning of pretext-invariant representations,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6707–6717.
[27] X. Chen, S. Xie, and K. He, “An empirical study of training self-supervised vision transformers,” 2021, arXiv:2104.02057.
[28] M. Caron et al., “Emerging properties in self-supervised vision transformers,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9630–9640.
[29] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, “Learning representations by predicting bags of visual words,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6928–6938.
[30] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in Proc. 19th Int. Conf. Comput. Statist., Springer, 2010, pp. 177–186.
[31] Y. You, I. Gitman, and B. Ginsburg, “Large batch training of convolutional networks,” 2017, arXiv:1708.03888.
[32] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” 2016, arXiv:1608.03983.
[33] H. Xu, H. Xiong, and G.-J. Qi, “K-Shot contrastive learning of visual features with multiple instance augmentations,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 11, pp. 8694–8700, Nov. 2022.
[34] X. Wang and G.-J. Qi, “Contrastive learning with stronger augmentations,” 2021, arXiv:2104.07713.
[35] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” Int. J. Comput. Vis., vol. 88, no. 2, pp. 303–338, 2010.
[36] T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. Eur. Conf. Comput. Vis., Springer, 2014, pp. 740–755.
[37] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2980–2988.

Xiao Wang (Graduate Student Member, IEEE) received the BS degree from the Department of Computer Science in Xi’an Jiaotong University, Xi’an, China in 2018. He is currently working toward the final year PhD degree with the Department of Computer Science at Purdue University, West Lafayette, IN, USA. He is the associate editor for IEEE Transactions on Intelligent Vehicles. He has published numerous 1st-author papers in a broad range of venues: Nature Methods, Nature Communications and IEEE Transactions on Pattern Analysis and Machine Intelligence. His research interests include deep learning, computer vision, bioinformatics, and intelligent systems.

Yuhang Huang received the BE degree in electronic information engineering from the University of Shanghai for Science and Technology, Shanghai, in 2019. He is currently working toward the ME degree with the Key Laboratory of Specialty Fiber Optics and Optical Access Networks and Shanghai Institute of Advanced Communication and Data Science, Shanghai University, Shanghai. His research interests include machine learning, computer vision, and pattern recognition.
Dan Zeng (Senior Member, IEEE) received the BS degree in electronic science and technology and the PhD degree in circuits and systems both from the University of Science and Technology of China, Hefei. She is a full professor and the dean of the Department of Communication Engineering with Shanghai University, directing the Computer Vision and Pattern Recognition Lab. Her main research interests include computer vision, multimedia analysis, and machine learning. She is serving as the associate editor for IEEE Transactions on Multimedia, the TC member of IEEE MSA and associate TC member of IEEE MMSP.

Guo-Jun Qi (Fellow, IEEE) is an adjunct professor with Westlake University, and the chief AI scientist leading and overseeing an international R&D team in the domain of Virtual Reality, Artificial Intelligence, as well as Digital Avatar Modeling and Animation with OPPO Research since 2021. He was the chief scientist for multiple intelligent cloud services, including smart cities, visual computing service, medical intelligent service, and connected vehicle service at Futurewei from 2018 - 2021. He was a faculty member with the Department of Computer Science and the director of MACHine Perception and LEarning (MAPLE) Lab with the University of Central Florida since August 2014. Prior to that, he was also a research staff member with IBM T.J. Watson Research Center, Yorktown Heights, NY. He has published more than 150 papers in a broad range of venues. Among them are the best student paper of ICDM 2014, “the best ICDE 2013 paper” by IEEE Transactions on Knowledge and Data Engineering, as well as the best paper (finalist) of ACM Multimedia 2007 (2015). He is a fellow IAPR and AAIA.