Knowledge-Based Systems 260 (2023) 110121

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A new contrastive learning framework for reducing the effect of hard negatives

Wentao Cui [a], Liang Bai [a,*], Xian Yang [b], Jiye Liang [a]

[a] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Institute of Intelligent Information Processing, Shanxi University, Taiyuan, Shanxi, China
[b] Alliance Manchester Business School, The University of Manchester, Manchester, UK

ARTICLE INFO

Article history:
Received 13 February 2022
Received in revised form 11 November 2022
Accepted 12 November 2022
Available online 17 November 2022

Keywords:
Self-supervised learning
Contrastive learning
Hard negatives
Student-t distribution
Neighbor consistency constraint

ABSTRACT

Contrastive learning as a self-supervised method has achieved great success. Although it is an instance-level discriminative method, the model can eventually learn latent semantic class information. Its core idea is pulling different views of the same instance closer but pushing out different instances. However, treating an instance as a class hinders the model from learning true latent semantic classes, which is caused by instances (called hard negatives) that are similar to the anchor but do not belong to the same semantic class. In this paper, we propose a new contrastive learning framework based on the Student-t distribution with a neighbor consistency constraint (TNCC) to reduce the effect of hard negatives. In this framework, we propose to use the loss based on the Student-t distribution as the instance-level discriminative loss to keep hard negatives far away. Furthermore, we add a new neighbor consistency constraint to maintain consistency within the semantic classes. Finally, we compare TNCC with recent state-of-the-art contrastive learning methods on five benchmark datasets to verify the effectiveness of the proposed framework.

1. Introduction

Since the training of deep neural networks relies on large-scale labeled datasets, many researchers have begun to conduct substantial research on unsupervised methods or self-supervised methods. Among them, the key to self-supervised learning (SSL) is how to set up a pretext task (e.g., context prediction [1], colorization [2], inpainting [3], rotation [4]) so that the model can obtain useful feature information from a large amount of unlabeled data. At present, self-supervised learning has achieved encouraging results in natural language processing [5,6] and computer vision in the field of image [7,8] and video [9].

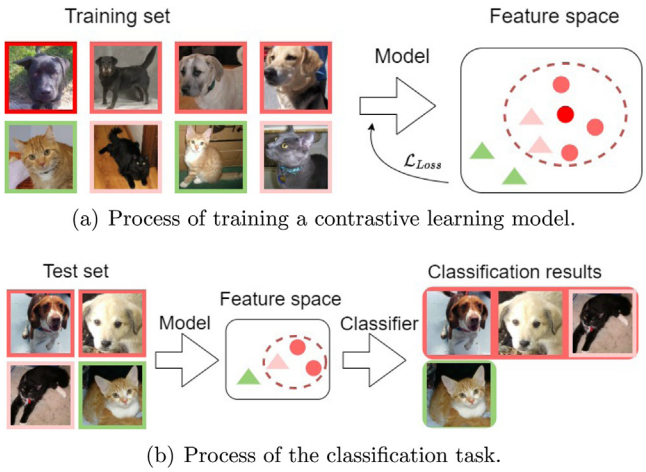
Recently, a particular kind of self-supervised learning method has become popular, known as instance-level discrimination, i.e., contrastive learning (CL). The core idea of CL methods is pulling feature embeddings of two transformed versions of the same instance (positives) close to each other but pushing embeddings of other instances (negatives) apart. For typical CL methods, negatives generated through data augmentations are critical. On the one hand, they can avoid collapsing solutions during the training process. On the other hand, they increase the complexity of the learning task so that the model can learn more essential features of the instance. In addition, most of the typical CL methods are based on the InfoNCE loss [10].

It is worth mentioning that although contrastive learning is an instance-level discriminative method, the model can eventually learn latent semantic class information of the data [11,12]. In other words, ideally, the final learned result should be that features of instances within the same semantic class are very close, but features of instances between different semantic classes should be far away. However, there are often some negatives with different semantic classes that are very similar to positives, which we call hard negatives. To understand this problem intuitively, we take the cat–dog dataset as an example, which is shown in Fig. 1. In this figure, the dog in the dark red border is the anchor and the others are negatives. There are some negatives with the same semantic class as the anchor (i.e., dogs with light red borders) and a few hard negatives (i.e., cats with pink borders). These hard negatives are close to the anchor in the feature space, which will mislead the model to learn the wrong latent semantic class information to a certain extent. When performing the downstream task, such as the classification task, the model's incorrect concept of semantic classes will lead to poor classification results.

We believe the reason is that the similarity measure in the InfoNCE loss is similar to a Gaussian kernel function. This function is positively correlated with the similarity of feature embeddings. Under this distribution, the cosine similarity between features is magnified in the same way, which makes the negatives with

* Corresponding author.
E-mail addresses: cuiwentao.sxu@qq.com (W. Cui), bailiang@sxu.edu.cn (L. Bai), xian.yang@manchester.ac.uk (X. Yang), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.knosys.2022.110121
0950-7051/



(a) Process of training a contrastive learning model.

(b) Process of the classification task.

Fig. 1. An example of hard negatives affecting model training and the downstream task, such as the classification task.

relatively high similarity to the anchor (i.e., some negatives of the same semantic class as the anchor and most hard negatives) have the same effect on the model. In this case, the model's concept of latent semantic classes is weakened. In this paper, we propose a new contrastive learning framework based on the Student-t distribution with a neighbor consistency constraint (TNCC) to address this problem. The Student-t distribution has heavier tails and can more easily produce values below the mean of similarity. Under this distribution, negatives with high similarity to the anchor will play a major role in training the model, while other negatives with relatively high similarity will be less important. In the later stage of training, the anchor will have high-similarity negatives with the same semantic class. This makes the model better learn the semantic classes under the Student-t distribution. On this basis, we further propose a neighbor consistency constraint to constrain the semantic class consistency between specific samples. Finally, our contributions to this paper can be summarized as follows:

- We replace the InfoNCE loss with the loss based on the Student-t distribution for contrastive learning. The long-tailed property of the Student-t distribution is taken advantage of to reduce the importance of the hard negatives.
- We add a new neighbor consistency constraint that uses particular simple samples and their nearest neighbors to maintain consistency within the semantic classes.
- We use some experiments to illustrate the effectiveness of the new contrastive learning framework compared with the state-of-the-art SSL methods.

The outline of the paper is as follows. In Section 2, we review some work related to this paper. In Section 3, we introduce the previous typical contrastive learning method and the related consistency constraint methods. Then, we present the loss, framework, and algorithm of TNCC in Section 4. The experimental results are presented in Section 5. The conclusions of this paper are provided in Section 6.

2. Related work

Data augmentation strategies and frameworks are essential for CL methods, so we review these two related contents. In addition, we have compiled relevant literature on Student-t distribution research.

Data augmentation strategies can be divided into strategies for positives and negatives. For positives, data augmentations are widely used [13–15], such as image cropping, rotation, color jittering, and Sobel filtering. Augmented instances can effectively improve the difference between positives and force the model to learn semantically invariant content between very different positives. It can greatly improve the feature extraction performance of the CL model. In addition, in the video field, Han et al. [16] proposed a sampling method of selecting matching samples from different views to form positives and proposed a new method of self-supervised collaborative training. Ding et al. [17] used contrastive learning to solve the person reidentification problem and used a similar threshold to dynamically select reliable similar images as positives.

For negatives, Kalantidis et al. [18] proposed two methods to generate new hard negatives at the feature embedding level. Chuang et al. [19] proposed a biased contrastive loss to correct the sampling bias, hoping to reduce the impact of false negatives as much as possible without labels. Given the characteristics of two disjoint datasets in the novel class discovery problem, Zhong et al. [20] proposed mixing specific labeled samples with unlabeled samples in the feature embedding space to generate new samples. Ho et al. [21] introduced the idea of adversarial examples into CL and used adversarial examples to generate harder pairs of positives and negatives. Robinson et al. [22] proposed a new sampling method to obtain harder negatives and avoid false negatives through the defined "hardness". Wang et al. [23] designed a negative generator trained against the encoder network in an adversarial manner to generate hard negative samples for contrastive learning in unpaired image-to-image translation.

Contrastive learning frameworks include the loss function and the specific Siamese network. In the CL method, which uses negatives, the framework usually uses the InfoNCE loss. Wu et al. [11] proposed using a memory bank to store instance-level class feature embedding [24] and used noise contrastive estimation (NCE) loss [25] to simplify the calculation process. Aiming at the problem of inconsistent feature embeddings in the memory bank, He et al. [26] proposed using queues to dynamically store and update feature embeddings and used a momentum update method to stabilize the Siamese network during the training process. Chen et al. [27] proposed a simple weight-sharing Siamese network framework, which utilized sufficient data augmentations, large batch sizes, and a new projection space to greatly improve the CL model's performance.

However, instance-level contrastive learning methods treat an instance as a class, which hinders the model from learning the true latent semantic class. To solve this problem, Cai et al. [28] assumed that feature embeddings obeyed a Gaussian distribution and calculated the upper bound of the InfoNCE loss with the mean and covariance of the samples, thereby introducing dependencies among different query-key pairs. Wei et al. [29] added a consistency constraint that used KL divergence to make each similarity distribution between the positive and negatives consistent. Fan et al. [30] proposed using dual thresholds (i.e., absolute similarity and relative similarity) to augment positives in the contrastive learning process.

Recently, there have been some new methods that only use positives. Caron et al. [31] introduced the idea of online clustering into the CL method (i.e., comparing the cluster assignment of multiple images). Grill et al. [32] proposed using an asymmetric Siamese network to predict the output of one view from another view, where one branch of the network is a momentum encoder, and its loss is the mean square error (MSE). Based on this work, Chen and He [33] removed the momentum encoder to further analyze the reason why the CL method using only positives is effective. However, regardless of whether negatives are used, the existing CL methods usually require a large batch size.

Student-t distributions have an important property: tail heaviness, which has been used in many aspects of deep learning.

Table 1
Definitions of main symbols.

Notation	Description
n	Number of instances
k	Range of simplest samples
m	Number of simplest samples
$T = \{t^a(\cdot)\}_{a=1}^{p}$	Family of augmentations
$f(\cdot)$	Base encoder network
$g_p(\cdot)$	Feature projection head
$g_n(\cdot)$	Class projection head
$X = \{x_i\}_{i=1}^{n}$	Dataset of n instances
X^a	The a-th view of X
$H = \{h_i\}_{i=1}^{n}$	Set of representations
H^a	The a-th view of H
$Z = \{z_i\}_{i=1}^{n}$	Set of feature embeddings
Z^a	The a-th view of Z
$Z^e = \{z_i^e\}_{i=1}^{m}$	Set of feature embeddings for simplest samples
$Z^{en} = \{z_i^{en}\}_{i=1}^{m}$	Set of feature embeddings of the nearest negative neighbor of z_i^e
$C = \{c_i\}_{i=1}^{n}$	Set of class embeddings
$C^e = \{c_i^e\}_{i=1}^{m}$	Set of class embeddings for simplest samples
$C^{en} = \{c_i^{en}\}_{i=1}^{m}$	Set of class embeddings of the nearest negative neighbor of c_i^e

Maaten and Hinton [34] presented a modification to stochastic neighbor embedding (SNE) [35] that used a Student-t distribution to show high-dimensional data in a low-dimensional latent space by assigning a place in a two-dimensional map to each data point. This modification is much faster to optimize because of the Student-t distribution instead of the Gaussian noise. To solve the instability of variational autoencoder (VAE) training caused by the zero-variance problem, Takahashi et al. [36] proposed using a Student-t distribution as the decoder distribution, which is robust to the error between the data point and its decoded mean. In addition, because labeled data in the medical field are scarce and expensive, Ahmad et al. [37] used the GAN with an auxiliary classifier, which samples the noise vector from a heavy-tailed Student-t distribution instead of a random noise Gaussian distribution to improve the diversity in the generated images.

3. Preliminary

In this section, we first give definitions of the main symbols used in this paper in Table 1. Then, we review previous methods related to our method in detail, specifically divided into the instance-level contrastive loss (i.e., InfoNCE) and the consistency constraint loss.

Typical contrastive learning with InfoNCE. Typical contrastive learning methods adopt InfoNCE or its variants for discriminating different instances, which encourages the positives to be pulled closer in the feature space yet the negatives to be pushed away. Specifically, given a training set $X = \{x_1, x_2, \ldots, x_n\}$ of n instances, we generate two correlated views (X^1, X^2), which can be written as $X^a = t^a(X)$, where t^a is randomly selected from the family of augmentations T. Then, we use a base encoder network $f(\cdot)$ to extract representations (H^1, H^2) and mapping (H^1, H^2) to feature embeddings (Z^1, Z^2) in the feature space through the feature projection head $g_p(\cdot)$, i.e., $Z^a = g_p(f(X^a))$. For convenience, we denote (z_i, z_j) as a positive pair (i.e., correlated feature embeddings). With similarity measured by the dot product, the InfoNCE loss is defined as:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{2n}\sum_{i=1}^{2n}\log\frac{\exp(z_i \cdot z_j/\tau)}{\sum_{i\neq k}\exp(z_i \cdot z_k/\tau)}, \tag{1}$$

where τ is a temperature hyperparameter and z is distributed on the hypersphere through ℓ_2 normalization.

Consistency constraint loss. In research of novel class discovery (NCD) [20,38], there is a consistency regularization term, which enforces the network to produce similar predictions for an instance x_i^1 and its correlated instance x_i^2. Specifically, in the NCD task, there are labeled dataset D^l and unlabeled dataset D^u, containing C^l and C^u classes, respectively. Given instances X, we can obtain correlated views (X^1, X^2), where $X^a = t^a(X)$. Then, we obtain their feature embeddings (Z^1, Z^2) through a base encoder network $f(\cdot)$ and a feature projection head $g_p(\cdot)$, the same as above. Finally, the mean squared error (MSE) is used as the consistency constraint loss:

$$\mathcal{L}_{MSE} = \frac{1}{C^l}\sum_{i=1}^{C^l}(g_i^l(z^{l1}) - g_i^l(z^{l2}))^2 + \frac{1}{C^u}\sum_{j=1}^{C^u}(g_j^u(z^{u1}) - g_j^u(z^{u2}))^2, \tag{2}$$

where $g(\cdot)$ is a linear classifier. It should be noted that the consistency constraint is for related views such as the positive pair in contrastive learning.

4. Our method

In this section, we introduce our TNCC method. More specifically, Section 4.1 introduces the loss of TNCC, and Section 4.2 describes the framework and the specific algorithm of TNCC.

4.1. TNCC loss

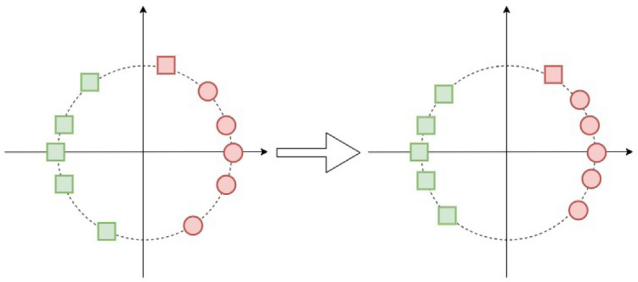
The loss of TNCC is divided into two parts. The first constraint is an instance-level contrastive learning loss based on the Student-t distribution (CLT), and the other constraint is a neighbor consistency constraint (NCC) that is used to evaluate the difference in semantic class information between instances. Note that the neighbor information is unreliable in the early training. Thus, we set a coefficient w that gradually increases with the training epoch to obtain better performance as in [38]. The overall objective function for TNCC can be expressed as:

$$\mathcal{L}_{\text{TNCC}} = \mathcal{L}_{\text{CLT}} + w\mathcal{L}_{\text{NCC}}. \tag{3}$$

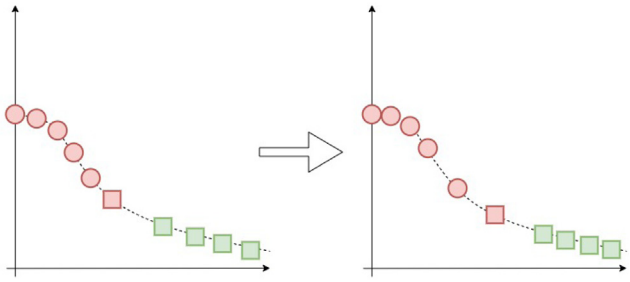
The contrastive learning loss based on the Student-t distribution. In typical contrastive learning, the learned latent features of instances are mapped onto a hypersphere over which the probability distribution is assumed to be uniform. The similarity measure (i.e., $e^{(\frac{z_i \cdot z_j}{\tau})}$) in $\mathcal{L}_{\text{InfoNCE}}$ is similar to a Gaussian kernel function. The cosine similarity between features is magnified by $\frac{1}{\tau}$ times in the same way. This makes anchors' relatively high similarity negatives, including some negatives of the same semantic class as the anchor and most hard negatives, have almost the same effect on the model. During typical contrastive learning, hard negatives will get closer to the anchor, as shown in Fig. 2(a). When there are a large number of hard negatives, the model will be affected in learning true latent semantic information. Therefore, we use the Student-t distribution with one degree of freedom to define a new instance-level contrastive loss (CLT), which is described as follows:

$$\mathcal{L}_{\text{CLT}} = -\frac{1}{2n}\sum_{i=1}^{2n}\log\frac{(1+\|z_i - z_j\|^2)^{-1}}{\sum_{i\neq k}(1+\|z_i - z_k\|^2)^{-1}}, \tag{4}$$

where (z_i, z_j) is a positive pair. Since the Student-t distribution with one degree of freedom has heavier tails, it can more easily produce values below the mean of similarity. Under the Student-t distribution, negatives with high similarity to the anchor will play a major role in training the model, while other negatives



(a) Process of typical contrastive learning methods.



(b) Process of CLT.

Fig. 2. Comparison of typical contrastive learning methods with CLT. Circles and squares represent instances of the two semantic classes. The red square is the hard negative. (a) Typical contrastive learning methods gradually bring the red square closer to the side of the circle. In contrast, (b) CLT exploits the long tail to gradually move the hard negatives away.

with relatively high similarity will be less important. This is equivalent to having a hidden threshold to ensure that fewer reliable negatives are more effective. Therefore, during the CLT optimization process, the anchor will have high-similarity negatives with the same semantic class as in Fig. 2(b). This enables the model to effectively learn the concepts of semantic classes under the Student-t distribution. Another advantage of $\mathcal{L}_{\text{CLT}}$ is that it requires no hyperparameters.

The neighbor consistency constraint for the simplest samples. During the training process, we observed a phenomenon: in each batch, there are always some samples whose feature embeddings are far from most other feature embeddings. We call these particular samples "the simplest samples", which are denoted as Z^e. To be more intuitive, we count the farthest negatives of each sample (the anchor) in the first batch within different training epochs of the CLT model on the CIFAR-10 dataset. Detailed experimental settings are provided in Section 5.3. As shown in Fig. 3, the Y-axis represents the frequency of this instance as the farthest negative. Augmented instances represent the instances in the batch that are used to train the model. For the convenience of observation, we add their corresponding original instances in the next line. We can see that the farthest negatives are concentrated in a few instances. Interestingly, the semantic classes of most of the farthest negatives are the same, such as "frog", "automobile", and "horse". We consider these samples to be well identifiable by the model during instance-level contrastive learning.

Therefore, we add the neighbor consistency constraint that uses these discriminative simplest samples to impose consistency constraints on the nearest neighbor, further enhancing the consistency of semantic classes. Considering the credibility and proportion of the simplest samples, we set up the limitation of distance and quantity when selecting these samples. Specifically, we first add the top-k furthest negatives of each sample to the simplest samples Z^e. Then, we count the number of occurrences of each simplest sample and keep only the top-m most frequent simplest samples. Next, we add the nearest negative neighbor of each simplest sample to Z^{en}. Finally, we project Z^e and Z^{en} into the class space through the class projection $g_n(\cdot)$ for the consistency constraint:

$$\mathcal{L}_{\text{NCC}} = \frac{1}{m}\sum_{i=1}^{m}(c_i^e - c_i^{en})^2, \tag{5}$$

where $c_i^e = softmax(g_n(z_i^e))$ and c_i^{en} are the same. As shown in Fig. 4, circles and triangles represent instances of two semantic classes, and blue circles and triangles represent the simplest samples. The instances in the dashed box are the simplest sample and its nearest negative neighbor. The green box represents that the simplest sample has the same class as its nearest neighbor, and the red box represents that the simplest sample has a different class from its neighbor. After the process of NCC, neighbors of the same class as the simplest sample hardly move, while the other neighbors whose class is different from the simplest sample are pushed away. In addition, since the simplest samples contribute little to the instance-level contrastive loss, NCC also improves the utilization of these samples.

4.2. TNCC framework and algorithm

As shown in Fig. 5, the TNCC framework can be divided into two parts, corresponding to two losses. The former part is consistent with the typical contrastive learning framework, except that we use the loss $\mathcal{L}_{\text{CLT}}$. The latter part is the NCC module, where we add the semantic class consistency constraint. In more detail, we first augment instances X to obtain relevant views (X^1, X^2), and then we obtain feature embeddings (Z^1, Z^2) through the base encoder network $f(\cdot)$ and the feature projection head $g_p(\cdot)$. We calculate the loss $\mathcal{L}_{\text{CLT}}$ with Eq. (4). Next, we filter out qualified simplest samples Z^e in the current batch and find their nearest negative neighbors Z^{en}. Furthermore, we obtain class embeddings by projecting Z^e and Z^{en} into the class space through a class projection head $g_n(\cdot)$. Finally, we calculate the loss $\mathcal{L}_{\text{NCC}}$ with Eq. (5) and the overall loss $\mathcal{L}_{\text{TNCC}}$ with Eq. (3). The overall algorithm flow is shown in Algorithm 1.

Algorithm 1 The TNCC algorithm.

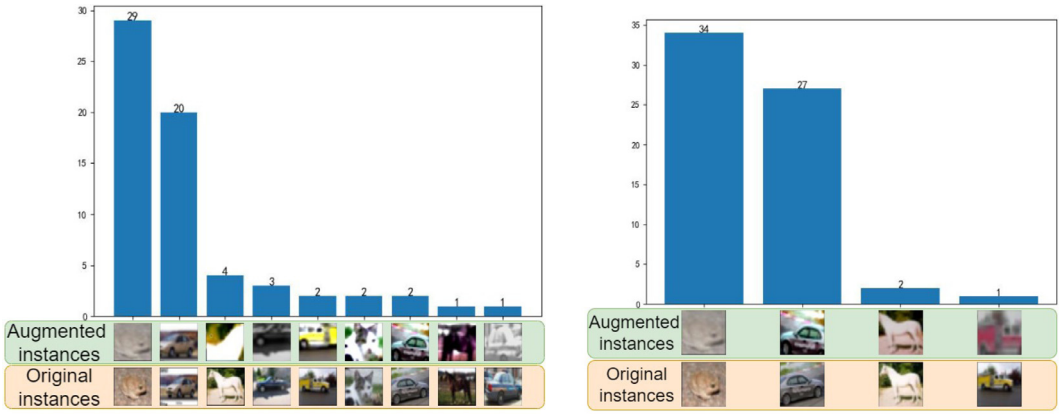
Input: range of simplest samples k; number of simplest samples m; batch of different augmented samples $\{x_i^1\}_{i=1}^n$, $\{x_i^2\}_{i=1}^n$; base encoder network $f(\cdot)$; feature projection head $g_p(\cdot)$; class projection head $g_n(\cdot)$

Output: the well trained encoder network $f(\cdot)$

for $i = 1$ to n **do**
  $h_i^1 = f(x_i^1), h_i^2 = f(x_i^2)$
  $z_i^1 = g_p(h_i^1), z_i^2 = g_p(h_i^2)$
calculate the loss $\mathcal{L}_{\text{CLT}}$ by Eq. (4)
for $i = 1$ to $2n$ **do**
  add the top-k furthest negatives of z_i to simplest samples Z^e
keep the top-m most frequent simplest samples in Z^e and select their nearest negative neighbors Z^{en}
$C^e = g_n(Z^e), C^{en} = g_n(Z^{en})$
calculate the loss $\mathcal{L}_{\text{NCC}}$ by Eq. (5)
optimize the network with the loss $\mathcal{L}_{\text{TNCC}}$ by Eq. (3)
return the encoder network $f(\cdot)$

5. Experiments

In this section, we compare our method with some advanced CL methods under datasets (CIFAR-10, CIFAR-100, STL-10, ImageNet-100, Voc2007) and verify the effectiveness of each module of the framework through ablation experiments.



(a) The CLT model was trained for 200 epochs. (b) The CLT model was trained for 400 epochs.

Fig. 3. The farthest negative of each anchor in the first batch within different training epochs of the CLT model on the CIFAR-10 dataset.

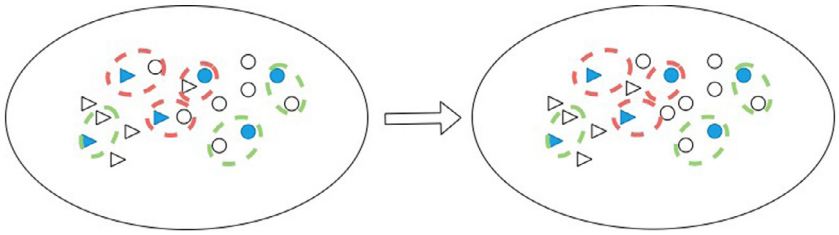


Fig. 4. A schematic representation of the NCC based optimization process.

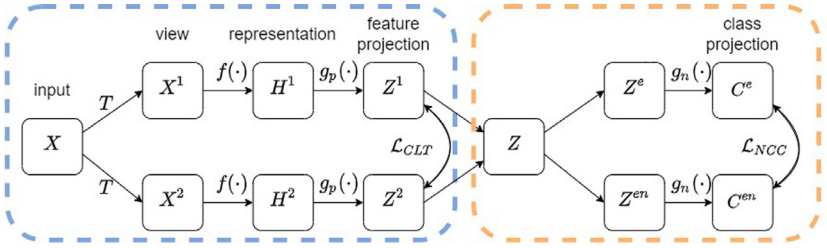


Fig. 5. The TNCC framework.

5.1. Experimental setup

In our experiments, we use the following **datasets**:

- **CIFAR-10 and CIFAR-100.** The CIFAR-10 and CIFAR-100 [39] datasets consist of 60,000 32 × 32 color images, specifically, 50,000 training images and 10,000 test images. There are 10 classes, with 6,000 images per class in CIFAR-10. And CIFAR-100 has 100 classes containing 600 images each.
- **STL-10.** The STL-10 [40] dataset is derived from the ImageNet-1k dataset, with 96 × 96 resolution images in 10 classes. It contains 100,000 unlabeled images for unsupervised learning. In addition, there are 1,300 labeled images for each class, including 500 training images (10 pre-defined folds), and 800 test images per class.
- **ImageNet-100.** ImageNet-100, i.e. IN-100, is a subset of the ImageNet-1k dataset [41] from ImageNet Large Scale Visual Recognition Challenge 2012. It contains random 100 classes. Each class contains 1,300 training images and 50 test images.
- **Voc2007.** The Voc2007 [42] dataset contains 9,963 images, specifically, 5,011 training images, and 4,952 test images. Voc2007 contains a total of 20 classes, and the number of samples in each class is inconsistent. The sizes of each image are inconsistent, roughly 500 × 375 (horizontal image) or 375 × 500 (vertical image).

Setting. To facilitate comparison with other CL methods, we use the same experimental settings where possible in our experiments, including the backbone network and the batch size. Specifically, we uniformly use ResNet-50 as the backbone, except that the first convolutional layer of DCL [19] and HCL [22] has been slightly changed, according to the network settings of the original paper. For the batch size, we set it to 64 on the IN-100 dataset and 32 on other datasets. We use the Adam optimizer [43], and the learning rate is set according to the paper or the best value, as shown in Table 2. Then, we reproduced some recent CL methods based on the codes and parameter setting methods provided in the paper. For the Voc2007 dataset, we train models for 500 epochs, and for other datasets, we train them for 400 epochs. In the NCC, the range of simplest samples k and the number of simplest samples m are set as shown in Table 3. The hidden layer dimension of the feature projection head $g_p(\cdot)$ is 2048, and the feature embedding size is 64, 128, or 256. The class embedding size is set as the number of classes in the dataset. In other methods, latent-space features are ℓ_2

Table 2
Learning rate schedule.

Method	MoCo	MoCo v2	SimCLR	BYOL	DCL	HCL	TNCC(ours)
LR	0.03	0.03	0.0003	0.0002	0.001	0.001	0.0003

Table 3
Hyperparameter settings of NCC on each dataset, the range of simplest samples k, and the number of simplest samples m.

Method	CIFAR-10	CIFAR-100	STL-10	IN-100	Voc2007
k	10	5	8	10	2
m	8	10	14	10	5

normalized, unless otherwise specified. In MoCo [26] and MoCo v2 [44], we set the temperature $\tau = 0.07$, the memory bank size $k = 65536$, and the momentum $m = 0.999$. In SimCLR [27], we set the temperature $\tau = 0.5$. In DCL [19], we set the temperature $\tau = 0.5$ and the positive class prior $\tau^+ = 0.1$. In HCL [22], we set the temperature $\tau = 0.5$. The positive class prior τ^+ and the concentration parameter β are set following [22] on different datasets. In BYOL [32], we set the exponential moving average parameter $\tau = 0.99$.

Image transformation details. For SimCLR [27], BYOL [32], and our TNCC, we first extract crops with a random size from 0.2 to 1.0 of the original area and then scale these crops to 224×224. Next, we apply horizontal flipping with probability 0.5, color jittering with configuration (0.8, 0.8, 0.8, 0.2) with probability 0.8, and grayscaling with probability 0.2. Last but not least, when testing, we only resize the image to 224×224.

For other methods, considering the differences in the network, i.e., DCL [19] and HCL [22], or differences in augmentations between MoCo [26] and MoCo v2 [44], we follow the augmentation settings in the original paper, both during training and testing.

Evaluation protocol. Following the widely adopted linear evaluation protocol, we only use the well-trained frozen backbone network (i.e., ResNet50) with fixed parameters to extract feature embeddings. Then, we use these sample features of the training set to train a supervised linear classifier for 500 epochs. Finally, we test the classification accuracy on the testing set. For the optimizer used in the training of the classifier, most methods use the Adam optimizer and set the learning rate according to the paper. However, MoCo and MoCo v2 use the SGD optimizer following the paper setting.

5.2. Experimental results

We test our model performance in terms of its classification ability and transfer ability. The specific experimental results are as follows:

Comparison with the state of the art. Through experiments, we found that TNCC works well when the feature embedding size is small. For ease of comparison, we test the classification accuracy of all methods under different feature embedding size settings. Table 4 shows our classification results. It can be seen that each method has a different preference for feature embedding size on different datasets, while TNCC fluctuates relatively more under different feature dimensions, i.e., on the CIFAR-100 and Voc2007 datasets. The CIFAR-100 dataset contains 100 classes, and the Voc2007 dataset contains 20 classes. We think the reason is that the Euclidean distance is used in the CLT module to measure the similarity between examples without ℓ_2 normalization, as in $\mathcal{L}_{InfoNCE}$. The Euclidean distance between samples' feature embeddings may be quite different in each dimension, which makes the CLT module sensitive to the feature dimension. However, in general, our method is better than previous work, except for the result on CIFAR-100, which is close to BYOL.

Table 4
Classification accuracy (Acc) under linear evaluation on the CIFAR-10, CIFAR-100, and STL-10 datasets. Mean average precision (mAP) on the Voc2007 dataset. Dim means feature dimensions.

Method	Dim	CIFAR-10	CIFAR-100	STL-10	Voc2007
		Acc	Acc	Acc	mAP
MoCo [26]	128	77.02	52.01	80.97	–
MoCo v2 [44]		84.39	60.90	85.63	–
SimCLR [27]		89.16	62.65	87.40	61.13
DCL [19]		87.03	57.27	82.98	53.67
HCL [22]		87.51	58.80	83.82	55.45
TNCC(ous)		89.97	62.53	89.62	60.02
BYOL [32]	256	88.70	64.23	87.36	56.89
TNCC(ous) [22]		89.79	62.42	89.85	61.33
MoCo [26]	64	77.37	50.66	81.51	–
MoCo v2 [44]		84.81	60.70	86.43	–
SimCLR [27]		88.53	62.80	88.16	60.71
DCL [19]		86.26	57.93	82.92	53.96
HCL [22]		87.80	59.55	84.11	54.20
BYOL [32]		88.13	64.21	87.34	57.68
TNCC (ours)		**90.67**	**64.37**	**90.11**	**61.75**

Table 5
Results of transfer learning across the CIFAR-10, CIFAR-100, and STL-10 datasets with ResNet50. The source dataset is used to train the model, and the target dataset is used to test the classification accuracy (Acc).

Source	Target	Method	Acc
CIFAR-10	CIFAR-100	SimCLR [27]	59.81
		TNCC (ours)	**61.64**
	STL-10	SimCLR [27]	72.59
		TNCC (ours)	**76.53**
CIFAR-100	CIFAR-10	SimCLR [27]	83.16
		TNCC (ours)	**83.86**
	STL-10	SimCLR [27]	70.21
		TNCC (ours)	**70.92**
STL-10	CIFAR-10	SimCLR [27]	84.15
		TNCC (ours)	**84.78**
	CIFAR-100	SimCLR [27]	57.24
		TNCC (ours)	**58.23**

In addition, we found that the performance of the linear classifier of some models decreased during the long training process. For example, the accuracy of HCL is 60.98 when training a linear classifier on the CIFAR-100 dataset for 100 epochs, and the accuracy decays to 58.80 when it is trained to 500 epochs. Conversely, the linear classifier of SimCLR is stable over long training epochs. We think it is fair since the parameters of all pretrained models are fixed during the whole testing process, and all the linear classifiers are trained with the same epochs.

Transfer learning. To verify whether TNCC can learn transferable features, we evaluate transfer learning performance with the CIFAR-10, CIFAR-100, and STL-10 datasets in linear evaluation settings. Specifically, we use the model trained on one dataset to test the classification accuracy of the other two datasets. Table 5 shows the results of our comparison with SimCLR. It can be seen that the transferable features learned by our method are better than SimCLR throughout the transfer learning test. In particular, our model trained on the CIFAR-10 dataset has the greatest improvement in transfer ability. We believe that a large amount of noise-free training data is crucial for the transfer ability of contrastive learning. This also means that our method has room for improvement when there is little training data for each class or when the training data contain irrelevant data.

Table 6
Ablation comparison results among SimCLR and TNCC.

Method	CIFAR-10	CIFAR-100	STL-10	Voc2007
	Acc	Acc	Acc	mAP
SimCLR [27]	88.53	62.80	88.16	60.71
CLT (ours)	89.61	63.81	89.83	61.40
NCC(ours)	40.57	14.83	29.44	–
TNCC (ours)	90.67	64.37	90.11	61.75

Table 7
Experimental results with different batch sizes on CIFAR-10.

Method	Batch size			
	16	32	64	128
SimCLR [27]	87.16	88.53	89.81	90.16
CLT (ours)	88.69	89.61	–	–
TNCC (ours)	88.75	90.67	90.40	–

Table 8
Classification accuracy on the ImageNet-100 dataset. Top-1 and top-5 correspond to the accuracy of a linear classifier.

Method	Top 1	Top 5
MoCo [26]	55.02	80.96
SimCLR [27]	69.03	90.21
CLT (ours)	68.17	89.82
TNCC (ours)	68.66	89.77

5.3. Ablation studies

Since TNCC consists of two parts, CLT and NCC, we conduct more detailed ablation experiments to verify the effectiveness of each module of the framework. Table 6 shows the ablation comparison results among the TNCC and SimCLR modules. Tables 7 and 8 show the effect of different batch sizes, i.e., different negative sample sizes.

Effect of the contrastive learning loss based on the Student-t distribution. Since our loss function is based on the Student-t distribution, it avoids the temperature hyperparameter τ in $\mathcal{L}_{InfoNCE}$ by taking advantage of the long tail of the Student-t distribution. That is, there are no hyperparameters in CLT. In addition, CLT does not perform ℓ_2 normalization on the feature embedding, so we test the effect of using different BN layers in the projection head $g_p(\cdot)$ on CIFAR-10, CIFAR-100, and STL-10. The results are shown in Fig. 6. From the figure, it can be observed that no BN layers give the worst effect, and adding one BN layer only after the first linear layer makes the model steadily improve. In the case that the BN layer is added after both linear layers, it only has a large improvement on the STL-10 dataset. Considering that STL-10 and IN-100 are the subsets of IN-1K, we set $g_p(\cdot)$ to the structure of the linear layer, BN layer, ReLU, linear layer, and BN layer on STL-10 and IN-100. For other datasets, we add only one BN layer after the first linear layer.

Effect of the neighbor consistency constraint for simplest samples. First, we intuitively show the farthest negative from the anchor in one batch on the CIFAR-10 dataset, as shown in Fig. 3. In the experiment, we set the manual seed to 42 so that the samples in the first batch of each epoch are the same. Then, we use the model trained with $\mathcal{L}_{\text{CLT}}$ for different epochs to extract features of samples in the first batch. The batch size is set to 32, so there are 64 anchors in one batch, and each anchor has 62 negatives because the anchor itself and its augmentation form a positive pair. Interestingly, the model seems to have a strong identification for some kinds of instances (such as frogs, automobiles, and horses). Based on the above phenomenon, we designed the NCC module.

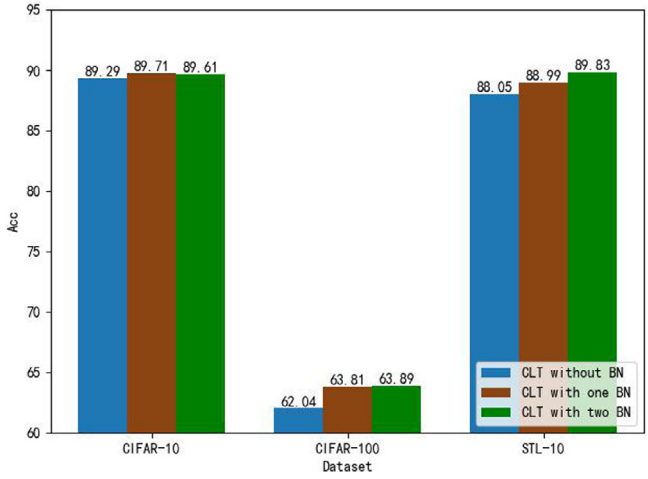


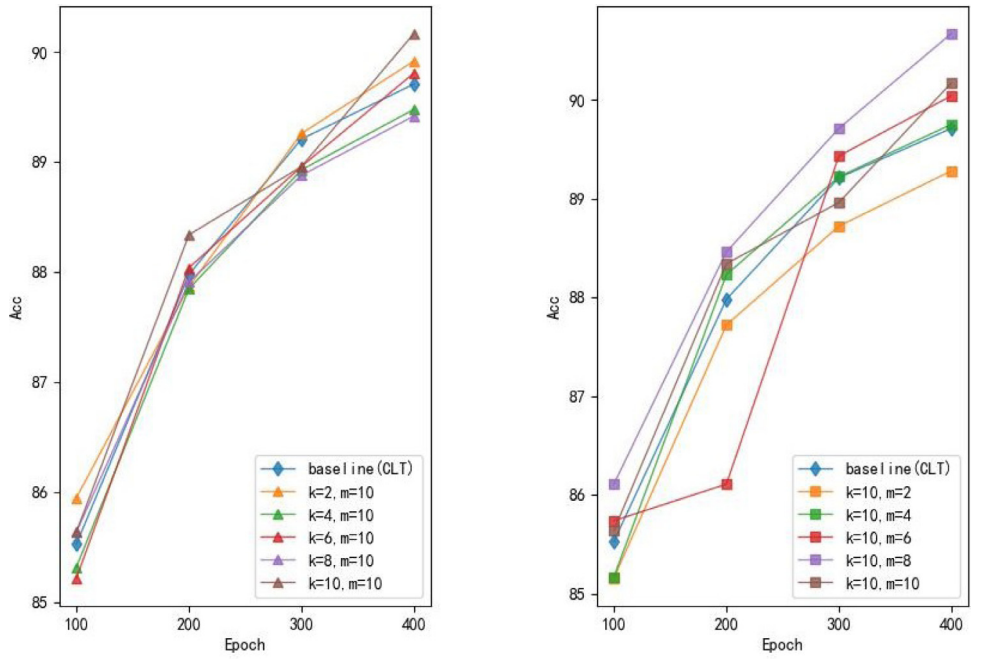
Fig. 6. The effect of the BN layer in the CLT module.

However, $\mathcal{L}_{\text{NCC}}$ used alone to train the model will cause collapsed solutions (e.g., outputting the same feature embedding for all images [32]) due to the Siamese network and the MSE loss. During training, it can be observed that the training loss rapidly approaches 0, but the model does not learn any useful representation. As shown in Table 6, the performance of the model trained with $\mathcal{L}_{\text{NCC}}$ is poor, and the classification accuracy depends entirely on the linear classifier. The backbone network is no different from a randomly initialized model. Therefore, it is important to ensure that the model does not collapse.

Based on the CLT module, the NCC module can steadily participate in the training of the model. The improved performance of NCC is the most obvious in the CIFAR-10 dataset, as shown in Table 6, which is consistent with our intuition. Since NCC implicitly adds the semantic class constraint to specific samples, it can enhance the consistency of semantic classes between samples. In CIFAR-10, there are only ten semantic classes and a large number of indistinguishable samples of cats and dogs, cars and trucks. The more hard negatives there are, the better the improved performance of NCC.

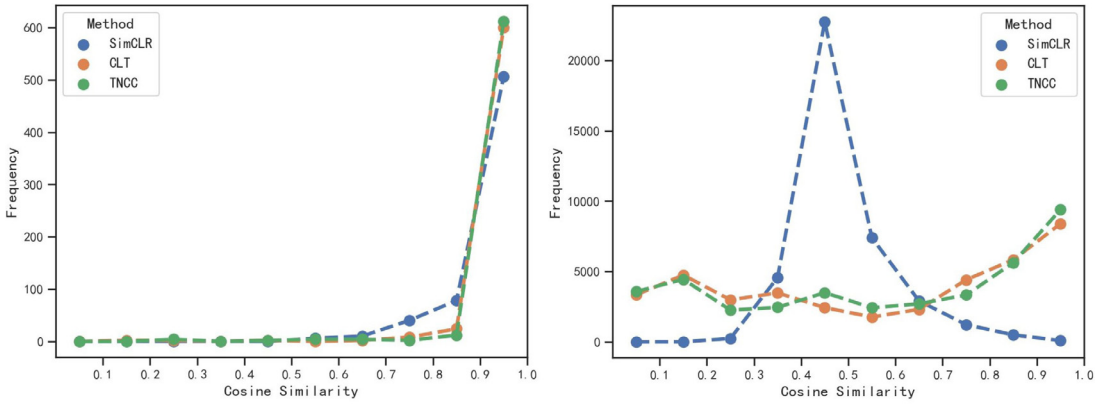
In addition to the above analysis, we test the effect of k and m in the NCC module on CIFAR-10. Specifically, we first fix $m = 10$ and increase k from 2 to 10 to study the effect of the range of simplest samples on the modular performance. Then, we fix $k = 10$ and increase m from 2 to 10 to study the effect of the number of simplest samples. The results are shown in Fig. 7. The number of simplest samples is critical for NCC, and too much or too little will lead to incorrect guidance for the model. However, overall, it is robust, as we can see that the performance of the framework after adding NCC is only worse than that of CLT when $m = 2$. The range of the simplest samples does not appear to be very regular. We believe this is due to the data itself and experimental settings, such as the number of semantic classes and the batch size.

Effect of different batch sizes. We test the classification accuracy of the SimCLR model and our CLT module and TNCC model at batch sizes of 16, 32, 64, and 128 on the CIFAR-10 dataset. The results are shown in Table 7. It can be seen that the model trained with $\mathcal{L}_{\text{CLT}}$ has excellent classification performance with few negatives. However, when the batch size is set to 64 or larger, the process of training the model using $\mathcal{L}_{\text{CLT}}$ becomes unstable. Specifically, there are cases where the loss suddenly becomes ***Nan***. We tried to reduce the learning rate of the Adam optimizer, normalize the feature embeddings, and add a very small number, such as $1e-12$, to the calculation of the Student-t distribution. When the batch size is 64, reducing the learning rate works well



(a) Effect of k in NCC. (b) Effect of m in NCC.

Fig. 7. The effect of the NCC module.



(a) Cosine similarity of positive pairs. (b) Cosine similarity of negative pairs.

Fig. 8. Cosine similarity of positive pairs and negative pairs for embeddings trained on CIFAR-10 with SimCLR, CLT, and TNCC.

on IN-100, as shown in Table 8, but not on CIFAR-10. Moreover, TNCC can be trained stably, although there is a certain drop in performance. We believe that the nearest neighbor consistency constraint in the NCC module plays a crucial role. However, the TNCC model will also become unstable if the batch size is set to 128. The NCC module can alleviate the instability of the CLT module under large batch sizes, but it cannot fundamentally solve it. Studying why CLT becomes unstable and how to solve this problem is one of our future works.

5.4. Verification experiments

To verify whether our model has learned the concept of semantic classes, we conduct an experiment on the CIFAR-10 dataset to plot the histograms of cosine similarities of positive pairs and negative pairs for the learned representations. We set the batch size to 32 and the manual seed to 42. There are 64 positive pairs and 3968 negative pairs (i.e., 64×62) in each batch. To avoid randomness, we count the first ten batches. We load the SimCLR, CLT, and TNCC models trained for 400 epochs for similarity statistics. Then, we convert the value of cosine similarity to $[0, 1]$ by $(cos + 1)/2$. We divide $[0, 1]$ equally into ten intervals and count the number of similarity values in each interval. The result is shown in Fig. 8.

It can be intuitively seen that the cosine similarity of positive pairs in CLT and TNCC is almost all in $[0.9, 1]$, while in SimCLR, it is distributed in $[0.6, 1]$. The cosine similarity of negative pairs in SimCLR is centered at $[0.3, 0.7]$, which means that the vast majority of negatives are pushed away equally by the anchor, even if they have the same class label as the anchor. In contrast, the cosine similarity of negative pairs in CLT and TNCC is distributed in various intervals, indicating that the anchor has different penalties for different negatives. This is exactly the result we want.

Based on the TNCC model, we looked at the negatives in the first batch where the similarity to the anchor was in $[0.9, 1]$. The first four anchors and their high-similarity negatives are

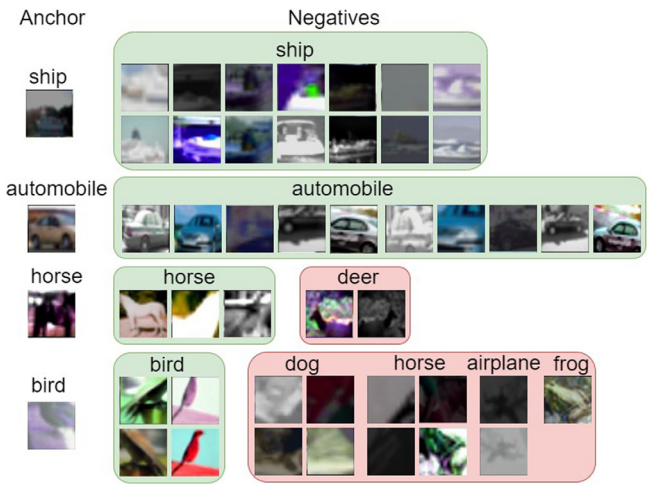


Fig. 9. The first four anchors and their high-similarity negatives in the first batch are based on TNCC. Images in the figure are augmented images in the batch. For the convenience of observation, we add the corresponding label above the images.

visualized in Fig. 9. Since the augmented images in batches may be illegible, we add corresponding labels to the images. Negatives with the same label as the anchor are boxed in green; conversely, those in red are negatives with different labels. We found that the model has a strong ability to recognize instances of mechanical classes, such as "ship" and "automobile", as all negatives with the same label as the anchor have high similarity. The model's recognition of animal classes is relatively weak. For example, in "bird", there are instances of "dog", "horse", "airplane", and "frog". However, there are also some negatives whose labels are the same as those of the anchor. This proves that our method enables the model to effectively learn the concepts of semantic classes.

6. Conclusions

In this paper, we propose a new contrastive learning framework (TNCC) to reduce the effect of hard negatives and enable the model to learn the concept of semantic classes. Specifically, we first use a loss based on the Student-t distribution (CLT) as an instance-level contrastive loss, exploiting its long-tailed property to keep hard negatives away. Then, we add a neighbor consistency constraint (NCC) to constrain the semantic class consistency between the simplest samples and their nearest neighbors. The results of linear evaluation and transfer learning are comparable to or superior to the state-of-the-art methods. Furthermore, we conduct ablation experiments and verification experiments to verify the effectiveness of each module and the overall framework.

CRediT authorship contribution statement

Wentao Cui: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization. **Liang Bai:** Conceptualization, Resources, Project administration, Funding acquisition. **Xian Yang:** Writing – review & editing. **Jiye Liang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors are very grateful to the editors and reviewers for their valuable comments and suggestions. This work is supported by National Key Research and Development Program of China (No. 2021ZD0113303), the National Natural Science Foundation of China (Nos. 62022052, 62276159).

References

[1] Carl Doersch, Abhinav Gupta, Alexei A. Efros, Unsupervised visual representation learning by context prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1422–1430.
[2] Richard Zhang, Phillip Isola, Alexei A. Efros, Colorful image colorization, in: European Conference on Computer Vision, Springer, 2016, pp. 649–666.
[3] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, Alexei A Efros, Context encoders: Feature learning by inpainting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2536–2544.
[4] Nikos Komodakis, Spyros Gidaris, Unsupervised representation learning by predicting image rotations, in: International Conference on Learning Representations, 2018.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, in: International Conference on Learning Representations, 2019.
[7] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, Xi Peng, Contrastive clustering, in: 2021 AAAI Conference on Artificial Intelligence, 2021.
[8] Junnan Li, Pan Zhou, Caiming Xiong, Steven Hoi, Prototypical contrastive learning of unsupervised representations, in: International Conference on Learning Representations, 2020.
[9] Xiaolong Wang, Allan Jabri, Alexei A. Efros, Learning correspondence from the cycle-consistency of time, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2566–2576.
[10] Aaron van den Oord, Yazhe Li, Oriol Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.
[11] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, Dahua Lin, Unsupervised feature learning via non-parametric instance discrimination, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3733–3742.
[12] Tongzhou Wang, Phillip Isola, Understanding contrastive representation learning through alignment and uniformity on the hypersphere, in: International Conference on Machine Learning, PMLR, 2020, pp. 9929–9939.
[13] Michael Laskin, Aravind Srinivas, Pieter Abbeel, Curl: Contrastive unsupervised representations for reinforcement learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 5639–5650.
[14] Yonglong Tian, Dilip Krishnan, Phillip Isola, Contrastive multiview coding, in: European Conference on Computer Vision, Springer, 2020, pp. 776–794.
[15] Ben Poole, Chen Sun, Cordelia Schmid, Dilip Krishnan, Phillip Isola, Yonglong Tian, What makes for good views for contrastive representation learning? in: Neural Information Processing Systems, 2020.
[16] Tengda Han, Weidi Xie, Andrew Zisserman, Self-supervised co-training for video representation learning, Adv. Neural Inf. Process. Syst. 33 (2020) 5679–5690.
[17] Yuhang Ding, Hehe Fan, Mingliang Xu, Yi Yang, Adaptive exploration for unsupervised person re-identification, ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) 16 (1) (2020) 1–19.
[18] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, Diane Larlus, Hard negative mixing for contrastive learning, in: Neural Information Processing Systems, 2020.
[19] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, Stefanie Jegelka, Debiased contrastive learning, in: Neural Information Processing Systems, 2020.

[20] Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, Nicu Sebe, Neighborhood contrastive learning for novel class discovery, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10867–10875.
[21] Chih-Hui Ho, Nuno Vasconcelos, Contrastive learning with adversarial examples, in: Neural Information Processing Systems, 2020.
[22] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, Stefanie Jegelka, Contrastive learning with hard negative samples, in: International Conference on Learning Representations, 2021.
[23] Weilun Wang, Wengang Zhou, Jianmin Bao, Dong Chen, Houqiang Li, Instance-wise hard negative example generation for contrastive learning in unpaired image-to-image translation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 14020–14029.
[24] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, Thomas Brox, Discriminative unsupervised feature learning with convolutional neural networks, Adv. Neural Inf. Process. Syst. 27 (2014) 766–774.
[25] Michael Gutmann, Aapo Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.
[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, Ross Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[27] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.
[28] Qi Cai, Yu Wang, Yingwei Pan, Ting Yao, Tao Mei, Joint contrastive learning with infinite possibilities, Adv. Neural Inf. Process. Syst. 33 (2020) 12638–12648.
[29] Chen Wei, Huiyu Wang, Wei Shen, Alan Yuille, CO2: Consistent contrast for unsupervised visual representation learning, in: International Conference on Learning Representations, 2021, URL https://openreview.net/forum?id=U4XLJhqwNF1.
[30] Hehe Fan, Ping Liu, Mingliang Xu, Yi Yang, Unsupervised visual representation learning via dual-level progressive similar instance selection, IEEE Trans. Cybern. (2021).
[31] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, Armand Joulin, Unsupervised learning of visual features by contrasting cluster assignments, in: Neural Information Processing Systems, 2020.
[32] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Pires, Zhaohan Guo, Mohammad Azar, et al., Bootstrap your own latent: A new approach to self-supervised learning, in: Neural Information Processing Systems, 2020.
[33] Xinlei Chen, Kaiming He, Exploring simple siamese representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.
[34] Laurens Van der Maaten, Geoffrey Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).
[35] Geoffrey Hinton, Sam T. Roweis, Stochastic neighbor embedding, in: Neural Information Processing Systems, Vol. 15, Citeseer, 2002, pp. 833–840.
[36] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, Satoshi Yagi, Student-t variational autoencoder for robust density estimation, in: International Joint Conference on Artificial Intelligence, 2018, pp. 2696–2702.
[37] Bilal Ahmad, Sun Jun, Vasile Palade, Qi You, Li Mao, Mao Zhongjie, Improving skin cancer classification using heavy-tailed student T-distribution in generative adversarial networks (TED-GAN), Diagnostics 11 (11) (2021) 2147.
[38] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, Andrew Zisserman, Automatically discovering and learning new visual categories with ranking statistics, in: International Conference on Learning Representations, 2019.
[39] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, in: Handbook of Systemic Autoimmune Diseases, Vol. 1, (4) 2009.
[40] Adam Coates, Andrew Ng, Honglak Lee, An analysis of single-layer networks in unsupervised feature learning, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.
[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009 pp. 248–255.
[42] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, Andrew Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[43] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[44] Xinlei Chen, Haoqi Fan, Ross Girshick, Kaiming He, Improved baselines with momentum contrastive learning, 2020, arXiv preprint arXiv:2003.04297.