TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 09/23 pp2092–2103
DOI: 10.26599/TST.2024.9010228
Volume 31, Number 4, August 2026

Research Article

Group Collaborative Unsupervised Deep Metric Learning for Feature Embedding

Zeqian Chen, Shichao Kan, and Min Li*

Abstract: Learning a compact feature embedding is crucial for effective image representation. Current feature embedding methods, including both supervised and unsupervised approaches, rely on deep metric learning techniques that aim to pull positive samples of the same class closer and push negative samples from different classes farther apart. However, supervised metric learning methods may exhibit bias towards the ground truth labels, leading to overfitting on the training set. On the other hand, unsupervised metric learning methods could suffer from degraded performance due to the long-tailed distribution of the clusters. To address these challenges, we propose a group collaborative unsupervised deep metric learning method for feature embedding. Specifically, we train the deep feature embedding model based on the teacher-student framework. The student network produces the final compact embedding, while the teacher network generates pseudo-labels for group collaborative learning and knowledge distillation. Both networks share a similar network structure, and the parameters of the teacher network are updated using the momentum-based moving average of the parameters of the student network. Experimental results on benchmark image retrieval datasets demonstrate the effectiveness and efficiency of the proposed method, achieving an improvement in Recall@1 of up to 1.8%.

Key words: feature embedding; deep metric learning; group collaborative learning

1 Introduction

Feature embedding aims to map high-dimensional data into a compact embedding space through metric learning methods. The fundamental concept involves optimizing the model by pulling positive samples of the same class closer together and pushing negative samples from different classes farther apart. It has been widely applied to face recognition[1, 2], image retrieval[3–6], person re-identification[7], and vehicle re-identification[8].

Current state-of-the-art feature embedding methods rely on a substantial amount of labeled data for effective learning. However, manually labeling a large quantity of samples is time-consuming and laborious. Furthermore, while supervised learning methods could enhance performance, the limited number of classes will impact the model's generalization capability. Existing unsupervised metric learning approaches often resort to assigning pseudo-labels to individual training instances through methods such as clustering. These conventional training strategies frequently fall short in capturing nuanced intra-class variations.

The current state-of-the-art unsupervised deep metric learning method, referred to as Self-Taught Metric Learning (STML)[9], excels in predicting semantic similarity with higher quality through the teacher-student framework. In this framework, the semantic similarity of data within the embedding space of the teacher network is leveraged for the approximate estimation of its class-equivalence relations. The

• Zeqian Chen, Shichao Kan, and Min Li are with School of Computer Science and Engineering, Central South University, Changsha 410000, China. E-mail: czq_1997@csu.edu.cn; kanshichao@csu.edu.cn; limin@mail.csu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2024-04-01; revised: 2024-08-04; accepted: 2024-11-13

predicted similarity by the teacher network serves as pseudo-labels for student network learning, and the teacher network is subsequently updated through a momentum-based moving average of the student.

During the STML training, nearest-neighbor sampling was employed to gather a batch of images. While this sampling method is straightforward and efficient, ensuring that each training batch includes diverse and relevant instances, the labels of the collected data often exhibit significant distribution differences in the early stages of training due to the long-tailed distribution clusters obtained using the clustering algorithm, as shown in Fig. 1. This, in turn, may impact the subsequent training results.

To address the distribution difference during batch sampling, we introduce a new method called Group Collaborative Unsupervised Deep Metric Learning (GC-UDML). As shown in Fig. 2, the GC-UDML network It includes two backbone networks for generating embeddings, one for supervision and one for learning, and a Group Collaborative Learning (GCL) decoder[10]. The teacher network generates supervision for the student network by expressing the contextual semantic similarity between embedding vectors. At the same time, the GCL decoder will also generate pseudo-label supervision for the student network based on the results of pre-clustering. Our backbone network for feature extraction uses a deep Convolutional Neural Network (CNN) (e.g., GoogleNet or Inception) that has been fully pre-trained on ImageNet. Other CNNs can also be used for configuring GC-UDML model.

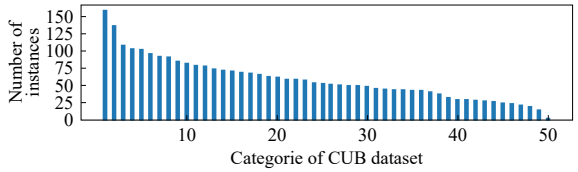

Fig. 1 Number of instances per class using the k-means clustering algorithm on the CUB-200-2011 (CUB) dataset.

Our GC-UDML model is evaluated on the standard benchmarks for deep metric learning and deep feature embedding. Experimental results demonstrate that the GC-UDML method converges faster and achieves an improvement in Recall@1 of up to 1.8%. The main contributions of our work are as follows:

- We introduce a new method called GC-UDML for category-level image retrieval. At the same time, the use of knowledge distillation loss between groups further optimizes the model, enabling it to adapt to the issue of the long-tail distribution of pseudo-labels in batch data.
- We exam our model on three uniformly distributed category-level image retrieval benchmark datasets and

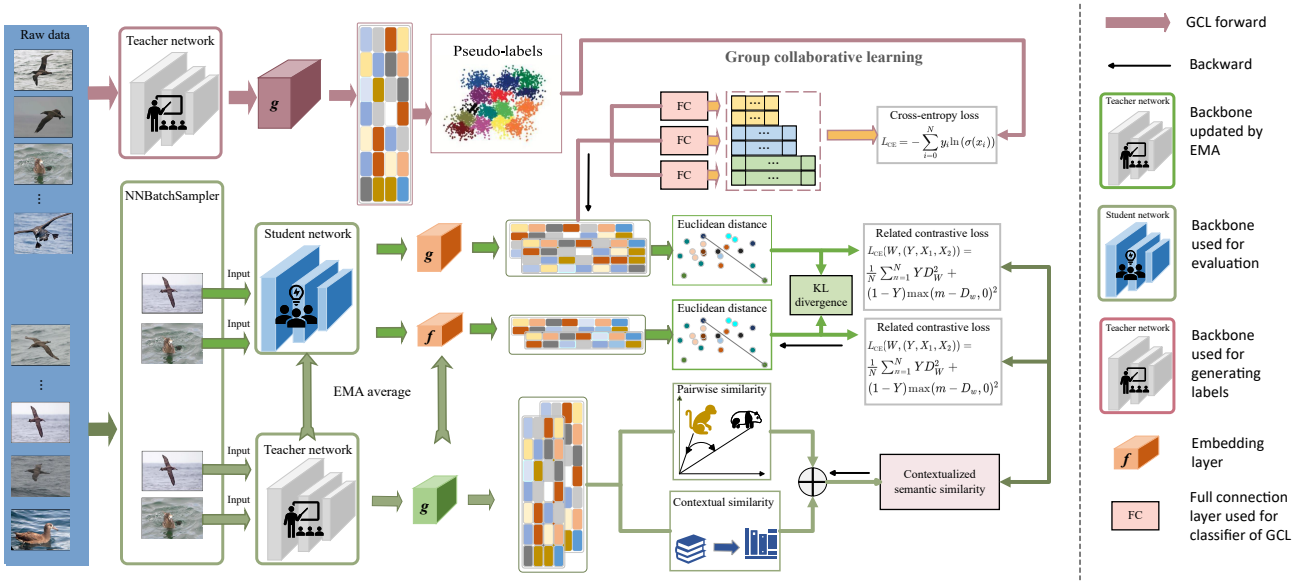

Fig. 2 Structure of the GC-UDML method based on the teacher-student framework. The parameters of the student network undergo updates through the contrastive loss function, the contextualized semantic distillation loss function, and the group collaborative learning loss function. Meanwhile, the parameters of the teacher network are updated through the Exponential Moving Average (EMA) of the student network. *g* and *f* are embedding layers of different dimensions. NNBatchSampler: Nearest Neighbor BatchSamoler; FC: Full Connection layer; KL: Kullback Leibler.

one instance dataset with a real-label long-tail distribution. We also conduct an in-depth discussion on the impact of various model parameters.

- We also delve into whether our strategy for addressing the long-tail distribution could be applied to other unsupervised metric learning methods. Taking Unsupervised Deep Metric Learning with Self-Supervision (UDML-SS) model as an example, the model's ability to handle the long-tail distribution significantly improved after incorporating the group learning strategy.

2 Related Work

Metric learning aims to learn an effective metric function that makes the Euclidean distance between similar images closer and the distance between dissimilar images farther, accurately measuring the similarity between images. Deep metric learning has a wide range of research directions and methods in this field, including the following: (1) Designing ranking-based loss functions: such as triplet loss[11], margin loss[12], contrastive loss[13], and structural matching loss[14]. These loss functions constrain the distances between similar samples to be closer and the distances between dissimilar samples to be farther, enabling the learning of effective representations. (2) Using proxies in tuple generation: methods such as VPL[15], ProxyNCA++[16], and Probabilistic DML[17] use proxies as surrogates in tuple generation. These methods address the complexity issue of tuple generation by training the model using proxy-based tuples. (3) Generating pseudo-labels using heuristic algorithms: methods like LogRatio Loss[18], ROUL[19], and SAN[20] generate pseudo-labels using heuristic algorithms. These methods consider the class equivalence relation between training data by leveraging pseudo-labels and existing supervised metric learning losses. (4) Orthogonal extensions: improving the performance of deep metric learning through orthogonal extensions, such as auxiliary feature learning, mutual learning, generation of artificial data, knowledge transfer, and more. These research directions contribute significantly to the field of deep metric learning, advancing research and applications in this area.

Unsupervised metric learning includes pseudo-labeling generation[19–21]. and instance discrimination[22, 23]. Instance discrimination adopts a contrastive learning strategy[18, 24, 25], treating each training sample as a separate class. The model is trained to make the feature embeddings of different samples in the embedding space as far apart as possible. Therefore, individual discrimination models can capture the unique details and features of each instance, enhancing their ability to generalize to similar data. However, this approach faces certain difficulties in subsequent potential class variation.

Self-supervised representation learning aims to acquire high-quality intermediate features that can be generalized to other downstream tasks. Various annotation-free tasks have been proposed such as rotation prediction[26], jigsaw puzzle solving[27], image colorization[28], and image inpainting[29]. Pseudo-labeling has also been extensively used as it enables the learning of models using conventional loss functions. Existing approaches in this direction discover pseudo classes through clustering or consider each training instance as a surrogate class. DeepCluster[30] proposed a scalable clustering approach for unsupervised visual feature representation learning. This method involves iteratively applying the k-means clustering to the features generated by deep neural networks and updating the parameters using a discriminative loss. In the UDML-SS[31] framework, along with the discriminative loss, rotation prediction is also incorporated into the model.

3 Proposed Method

3.1 Method overview

The GC-UDML framework is illustrated in Fig. 2. Our main goal is to train a universal image embedding network capable of representing any image. This network enables the extraction of embeddings for both query images and gallery images. Subsequently, we calculate the Euclidean distances between the query embedding and the gallery embeddings. These distances are used for ranking, which leads to the retrieval of the top-K images. To train the embedding network. the GC-UDML is built with a teacher network t and a student network s, both sharing the same backbone. The difference between t and s lies in the fact that the teacher network incorporates a high-dimensional embedding layer g^t with the encoder ϕ^t, while the student network comprises two parallel embedding layers, g^s and f^s, both sharing the same encoder ϕ^s. The layer g^s, serving as the model's final output layer, typically has a lower dimension (e.g., 256 or 512), while the f^s layer is used to update the teacher embedding layer g^s through an exponential moving

average, and it typically has a higher dimension (e.g., 512 or 1024). During training, the teacher network is utilized to estimate the semantic similarity between data, providing an approximation of their class equivalence relationship. This approximation serves as comprehensive supervision for training the student network.

To more effectively train the embedding layer, we have incorporated pseudo-labels into the training process and employed a GCL classifier to attempt classification of the embeddings generated by the student embedding layer g^s. The pseudo-labels used by GCL classifier is derived from the teacher network g^t in the preceding epoch. It features a GCL decoder that includes a series of classifiers, denoted as $\{l_k\}_{k=1}^N$, which collectively create an ever-growing classification space. All classifiers, except for the initial classifier l_1, must be capable of distinguishing predicate classes from both previous and current groups. In other words, the classification space in l_k should encompass the classification spaces of l_1, l_2, …, l_{k-1}, and l_k itself. The ultimate classification l_k is selected to obtain the final predictions during the evaluation stage.

3.2 Class grouping for the training of GCL decoder

In general, the distribution of class examples through clustering is not uniform, potentially impacting metric learning performance. To tackle this challenge, we introduce a method called GCL that discerns distinct clusters within multiple groups. Concretely, pseudo-labels obtained from clustering are assigned to their respective samples, followed by the grouping of diverse classes into separate groups. Ultimately, samples from different groups are classified using distinct classifiers.

Suppose the teacher network and student network are denoted as t and s, respectively, initialized with the same parameters. To curate the training data, we first obtain the embeddings of samples from the entire dataset using the teacher network t, given by the equation $z_k^t = (g^t \circ \phi^t)(x_i)$, where $\circ$ denotes function composition. These embeddings are then utilized to generate pseudo-labels through the k-means clustering. Subsequently, the classes of these clustered vectors are sorted based on the number of samples in each group in descending order, creating a set $P_{all} = \{p_i\}_{i=1}^N$, which is used to generate class groups $\{P_k\}_{k=1}^K$. The entire dataset is divided into K subsets $\{X_i\}_{i=1}^K$ with classification space P_k, employing the median resampling strategy[10]. This strategy ensures the balance of all instances in group P_k during training, as described below.

For each classification space P_k, it calculates the median amount $\mathrm{Med}(P_k)$ across all classes within P_k. For each class p_i in P_k, the sampling rate ϕ_i^k is computed by

$$\phi_i^k = \begin{cases} \dfrac{\mathrm{Med}(P_k)}{\mathrm{Count}(P_i)}, & \text{if } \mathrm{Med}(P_k) < \mathrm{Count}(p_i); \\ 1.0, & \text{if } \mathrm{Med}(P_k) \geqslant \mathrm{Count}(p_i) \end{cases} \tag{1}$$

Based on this, we can obtained subsets $\{X_i\}_{i=1}^K$, which have length of

$$\mathrm{Len}(X_i) = \sum_{i=1}^{N} \phi_i^k \times \mathrm{Count}(p_i) \tag{2}$$

It is noteworthy that the parameters of the GCL classifier are initially set to random values. To prevent any adverse impact on the backbone of both the teacher and student networks, we conduct pre-training for the GCL classifier.

3.3 Training procedure of GC-UDML

To train the GC-UDML model, we first consider using the contextual semantic similarities provided by the teacher network as synthetic supervision for the student network, and employing the relaxed contrastive loss to measure the contextual semantic similarities. Following STML[9], relative distance between embedding vector f_i^s and f_j^s is defined as $d_{ij}^{f_s} := \left\| z_i^{f^s} - z_j^{f_s} \right\|_2 \Big/ \left(\frac{1}{n} \sum_{k=1}^n \left\| z_i^{f_s} - z_k^{f_s} \right\|_2 \right)$, where $\|\cdot\|_2$ denotes L2 norm, then the relaxed contrastive loss $\mathcal{L}_{RC}$ can be expressed as

$$\mathcal{L}_{\mathrm{RC}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \neq i}^{n} w_{ij} \left(d_{ij}^{f_s} \right)^2 + \frac{1}{n} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \left(1 - w_{ij} \right) \left[\delta - d_{ij}^{f_s} \right]_+^2 \tag{3}$$

where $[\cdot]_+$ denotes the ReLU function, and w_{ij} denotes the contextual semantic similarities, which is divided into two parts: pairwise similarity w_{ij}^P and contextual similarity w_{ij}^C. The pairwise similarity w_{ij}^P is defined as

$$w_{ij}^P = \exp\left(- \frac{\left\| z_i^t - z_j^t \right\|_2^2}{\sigma} \right) \tag{4}$$

where σ is the Gaussian kernel bandwidth. Next, context is considered as k reciprocal nearest neighbors.

Defining the k-nearest neighbors $N_k(i)$ of x_i and the set of k reciprocal nearest neighbors $R_k(i) = j|(j \in N_k(i)) \wedge (i \in N_k(j))$ of a data point x_i, the contextual similarity w_{ij}^C is defined as

$$w_{ij}^C = \frac{1}{|N_{k/2}(i)|} \sum_{h \in N_{\frac{k}{2}}(i)} \hat{w}_{hj}^C \tag{5}$$

where $\hat{w}_{ij}^C$ is an asymmetric Jaccard similarity, which is expressed as

$$\hat{w}_{ij}^C = \begin{cases} \dfrac{|R_k(i) \cap R_k(j)|}{|R_k(i)|}, & \text{if } j \in R_k(i); \\ 0, & \text{else} \end{cases} \tag{6}$$

We must sample a mini-batch before calculating $\mathcal{L}_{\mathrm{RC}}$, the nearest neighbor batch sampling strategy used by STML[9] randomly selects k samples and their n nearest neighbors as a mini-batch. It is simple, efficient, and ensures a certain degree of correlation between the sampled instances. However, during the initial training phase, we observed that these nearest neighbor samples are not always from the same class; instead, they are randomly distributed across different classes, albeit in a small proportion. This can lead to the degradation of metric learning performance. Specifically, for each object x_i chosen by the median resampling strategy, after obtaining the embedding vector $z_k^{g_s}$ from the student backbone ϕ^s and embedding layer g^s, the class probability prediction $\tilde{y}_i^k$ generated by classifier l_k is

$$\tilde{y}_i^k = \mathrm{softmax}\left(\mathrm{FC}\left(z_i^{g_s}\right)\right) \tag{7}$$

where $\mathrm{FC}(\cdot)$ denotes the FC layer. All classifiers in GCL will be optimized by two loss funcntions. One is the Group Cross-Entropy (GCE) loss and the other is the knowledge distillation loss. The GCE loss is defined as follows:

$$\mathcal{L}_{\mathrm{GCE}} = \sum_{k=1}^{K} \frac{1}{X_k} \sum_{x_i \in X_k} \mathcal{L}_{\mathrm{CE}}(\bar{y}_i^k, \tilde{y}_i^k) \tag{8}$$

where $\bar{y}_i^k$ denotes the pseudo-label generated from the teacher network g^t, and $\tilde{y}_i^k$ denotes the predicted label of mini-batch data $\{x_i\}$ through the student network g_s. $\mathcal{L}_{\mathrm{CE}}$ is the Cross-Entropy (CE) lost function. The knowledge distillation loss is defined as follows:

$$\mathcal{L}_{\mathrm{GKD}} = \frac{2}{k(k-1)} \sum_{n=1}^{k} \sum_{m=1}^{n-1} \frac{1}{|X_m|} \sum_{x_i \in X_k} \mathcal{L}_{\mathrm{KL}}(\tilde{y}_i^m, \tilde{y}_i^n) \tag{9}$$

where $\mathcal{L}_{\mathrm{KL}}$ is the Kullback Leibler (KL) divergence loss.

The loss function used to train the GCL network is defined as follows:

$$\mathcal{L}_{\mathrm{GCL}} = \mathcal{L}_{\mathrm{GCE}} + \alpha \mathcal{L}_{\mathrm{GKD}} \tag{10}$$

where α controls the weighting of the GKD loss term. The overall loss function used to train the GC-UDML model is defined as follows:

$$\mathcal{L}_{\text{GC-UDML}} = \mathcal{L}_{\mathrm{RC}} + \beta \mathcal{L}_{\mathrm{GCL}} \tag{11}$$

By incorporating the GCL loss function weighted by β, the model can converge faster and obtain better feature embedding performance. Algorithm 1 describes the training procedure of GC-UDML in detail.

Algorithm 1 GC-UDML training strategy

Input: teacher network t, student network s, momentum m, classifier group for GCL $\{l_i(\cdot)\}_{i=1}^K$

Set t's parameter $\theta_t = \theta_{\phi^t} \cup \theta_{g^t}$ and s's parameter $\theta_s = \theta_{\phi^s} \cup \theta_{g^s}$;
Identically initialize the backbone of t and s;
for all samples in data $\{x_i\}_{i=1}^N$ **do**
    $z_i^t \leftarrow (g^t \circ \phi^t)(x_i)$;
end for
Cluster embedding vector z_i^t for pseudo-label $\bar{y}_i$;
Generate label set $\{p_t\}_{t=1}^T$ from pseudo-label $\bar{y}_i$ and divide into groups $\{P_k\}_{k=1}^K$;
for number of epochs **do**
    Construct mini-batches;
    for number of iterations **do**
        for all samples in a mini-batch $\{x_k\}_{k=1}^n$ **do**
            $z_k^{g_t} \leftarrow (g^t \circ \phi^t)(x_k)$;
            $z_k^{f_s} \leftarrow (f^s \circ \phi^s)(x_k)$, $z_k^{g_s} \leftarrow (g^s \circ \phi^s)(x_k)$;
        end for
        Compute $\mathcal{L}_{\mathrm{RC}}$ and let $\{z^{g^s}\} = \{z_k^{g^s}\}_{k=1}^n$;
        Divide $\{z^{g^s}\}$ into $\mathcal{Z}^{g_s} = \{z^{g^s}\}_1 \cup \{z^{g^s}\}_2 \cdots \cup \{z^{g^s}\}_K$ according to groups $\{P_k\}_{k=1}^K$ by labels;
        for $\{z^{g_s}\}_i \in \mathcal{Z}^{g_s}$ and $z_\sigma \in \{z^{g_s}\}_i$ **do**
            $\tilde{y}_\sigma^i \leftarrow l_i(z_\sigma^{g_s})$;
        end for
        Compute $\mathcal{L}_{\mathrm{GCL}}$ using Eqs. (8) and (9) and optimize t;
        $\theta_t \leftarrow m\theta_t + (1-m)\theta_s$;
    end for
end for

4 Experiment

4.1 Datasets and evaluation

Our model is evaluated and compared with state-of-

the-art models on three benchmark datasets: CUB-200-2011 (CUB)[32], Cars-196 (Cars)[33], and Stanford Online Product (SOP). Additionally, we also evaluate the performance of STML and GC-UDML on the Microsoft Common Objects in Context (COCO) dataset[34]. We follow the same train/test data split[9] as follows:

- CUB-200-2011: The CUB dataset has 11 788 images and 200 classes in total. We use the first 100 categories (5864 images) for training and the remaining 100 categories for testing.
- Cars-196: The Cars dataset has 16 185 images of 196 classes of cars. The first 98 categories are used for training, and the rest for testing.
- SOP: The SOP dataset consists of 120 053 images of 22 634 online products from eBay.com. We use the first 11 318 products (59 551 images) for training and the remaining 11 316 products (60 502 images) for testing
- Microsoft COCO: COCO is a semantic segmentation dataset that consists of 80 categories and over 200 000 labeled instances. We sample 100 000 instances from the 2017 train dataset by filtering out excessively large or small images. Then, we randomly undersample 70 000 data samples from the filtered dataset to maintain the original data distribution for training. For testing, we use 20 000 instances of the 2017 test dataset. We chose not to retain the background and only retained instances on images.

We conduct our experiments based on the GoogleNet[35] and Inception-BN[36] following the same backbone[9]. The performance on these datasets is assessed using Recall@k, which measures the proportion of queries that have at least one relevant sample among their k nearest neighbors in a learned embedding space.

4.2 Experimental settings

We utilize the Inception network pre-trained on ImageNet as the backbone for GC-UDML. To mitigate the adverse effects of randomly initialized parameters in the GCL classifier on the backbone networks after deploying, we conduct pre-training of the GCL classifier using pseudo-labels to achieve preliminary convergence. Throughout this process, we freeze the network parameters of the teacher and student networks, and we employ sequential sampling to input all the data into the GCL classifier. Simultaneously, we calculate the CE loss for each fully connected layer of GCL, as well as the knowledge distillation loss between these layers. The learning rate is set to 1×10^{-2}, a slightly higher rate than the end-to-end training. Besides all experiments, we crop the images to a size of 227 × 227 and applied cropping, flipping, and rotation as data augmentation techniques. We utilize the AdamP[37] optimizer GC-UDML model.

In the formal training phase after pre-training, for each epoch, we perform median resampling on the dataset to obtain a subset of samples as the training data for that epoch. We continue to use the NNBatchSampler[9] method to obtain mini-batches. Unlike STML, in the formal training phase after pre-training, we do not randomly sample p initial instances from the entire dataset. Instead, we randomly select a total of p samples from the classification spaces of each classifier in the GCL model. From these p initial samples, we then choose q nearest neighbors to form the mini-batch. This approach ensures that each classifier in the GCL model is allocated at least one data sample, promoting balanced training across all classifiers.

In the parallel classifier setup of the GCL decoder, we categorize the classes of pseudo-labels into two to four groups, with varying numbers of classes in each group, ranging from few to many. Through experimentation, optimal performance is observed when these classes are divided into two groups at a ratio of 1:4. For instance, in a dataset with 100 classes, we assign the top 20 classes with the highest number of samples to the classification space of the first classifier, and the remaining 80 classes to the classification space of the second classifier. For CUB and Cars datasets, we use 100 clusters while we show evaluations of SOP with 100, 1000, and 10 000 clusters, as it contains a much larger number of samples. The rates of $\mathcal{L}_{\mathrm{GKD}}$ and $\mathcal{L}_{\mathrm{GCL}}$ are set to 6×10^{-4} and 1×10^{-4}, respectively.

4.3 Comparison with the state-of-the-art methods

We compare our method with state-of-the-art unsupervised deep metric learning methods. In accordance with the STML approach, we perform experiments with GoogLeNet, exploring embedding dimensions of 64, 128, and 512. Furthermore, we extend our investigation to BN-Inception, employing an embedding dimension of 512. Results are shown in Table 1. From the experimental results, we observe that our method achieved the best performance in most situations. Especially, Recall@1 improves by 0.4%,

Table 1 Comparition of the performance of GC-UDML and other unsupervised deep metric learning methods under the GoogleNet (G) and BN-Inception (BN) architectures, with the superscript number indicating the embedding layer's dimensionality. A dash (-) denotes unreported result. Bold values highlight the best results among the compared methods.

(%)

Method	Architecture	Performance								
		CUB			Cars			SOP		
		Recall@1	Recall@2	Recall@4	Recall@1	Recall@2	Recall@4	Recall@1	Recall@10	Recall@100
MOM[38]	G^{64}	45.3	57.8	68.6	-	-	-	-	-	-
ROUL[19]	G^{64}	52.0	64.0	74.5	-	-	-	-	-	-
STML[9]	G^{64}	55.0	67.5	78.6	41.7	53.2	**65.6**	63.5	78.9	88.7
GC-UDML	G^{64}	**55.8**	**67.6**	**79.1**	**41.8**	**53.9**	65.5	**64.9**	**79.8**	**89.8**
PSLR[23]	G^{128}	48.1	60.1	71.8	43.7	54.8	66.1	51.1	66.5	79.8
ROUL[19]	G^{128}	56.7	68.4	78.3	45.0	56.9	68.4	53.4	68.8	81.7
SAN[20]	G^{128}	55.9	68.0	78.6	44.2	55.5	66.8	58.7	73.1	84.6
STML[9]	G^{128}	58.1	70.1	**80.5**	47.7	**59.3**	**70.2**	64.6	79.1	89.3
GC-UDML	G^{128}	**58.9**	**70.4**	**80.5**	**47.8**	58.9	69.8	**65.3**	**79.7**	**89.8**
UDML-SS[31]	G^{512}	54.7	66.9	77.4	45.1	56.1	66.5	63.5	78.0	88.6
TAC-CCL[21]	G^{512}	57.5	68.8	78.8	46.1	56.9	67.5	63.9	77.6	87.8
UHML[18]	G^{512}	58.9	70.6	80.4	47.7	58.9	70.3	65.1	78.2	88.3
STML[9]	G^{512}	**59.9**	**71.5**	**80.9**	49.1	60.9	71.1	65.0	79.3	**89.7**
GC-UDML	G^{512}	**59.9**	71.4	80.8	**51.3**	**63.0**	**73.9**	**65.3**	**79.7**	**89.7**
UDML-SS[31]	BN^{512}	63.7	75.0	83.8	-	-	-	-	-	-
STML[9]	BN^{512}	67.0	77.5	85.5	64.8	73.8	81.6	68.8	82.2	91.1
GC-UDML	BN^{512}	**67.4**	**77.7**	**85.7**	**66.6**	**75.5**	**82.9**	**70.0**	**82.8**	**91.3**

1.8%, and 1.2% on the CUB, Cars, and SOP datasets, respectively, based on the BN-Inception network.

In the experiments conducted on the COCO dataset, we compare only the STML and GC-UDML methods, and Table 2 presents the corresponding results. Specifically, when using the GoogleNet model with an embedding dimension of 512, GC-UDML achieves a 3.5% improvement in Recall@1 metric compared to STML. The t-SNE visualization of the trained CUB dataset is depicted in Fig. 3, and some example queries are shown in Fig. 4.

4.4 Ablation study

In our ablation study, we perform experiments on the CUB and Cars datasets, and the results are presented in Figs. 5 and 6. Figure 5 depicts the impact of hyperparameters α (the rate of GKD loss) and β (the rate of GCL loss) on Recall@1 in the CUB and Cars datasets. Optimal values for α and β are observed to be 6×10^{-1} and 1×10^{-4}, respectively. Figure 6 and Table 3 explore the effects of the number of clusters determined by k-means (on the left of Fig. 6) and the partitioning method (on the right of Fig. 6) on Recall@1 for the Cars dataset and the SOP dataset. The optimal number of clusters is found to be 100 for the Cars dataset. Regarding the partition ratio, the best value is 1:4. These values are set as default in our experiments.

Table 2 Performance on the COCO data between GC-UDML and STML. Bold values highlight the best results among the compared methods. G denotes GoogleNet, with the superscript number indicating the embedding layer's dimensionality.

(%)

Method	Architecture	Recall@1	Recall@2	Recall@4
STML[9]	G^{64}	70.8	77.8	83.6
GC-UDML	G^{64}	**71.7**	**79.8**	**85.0**
STML[9]	G^{128}	70.6	77.5	82.7
GC-UDML	G^{128}	**71.7**	**79.8**	**85.0**
STML[9]	G^{512}	68.2	76.5	82.9
GC-UDML	G^{512}	**70.6**	**77.7**	**83.2**

4.5 Expansion of the GCL model

In this section, we investigate the performance of the GCL classifier strategy in comparison to other methods that utilize k-means clustering to generate pseudo-labels such as DeepCluster[30] and UDML-SS[24].

UDML-SS is one of the representative methods for unsupervised metric learning based on k-means clustering. The main idea behind UDML-SS is to generate pseudo-labels by k-means from feature extracted by backbone network, in Fig. 7. These pseudo-labels are then used to compute the multi-similarity loss with the output of the metric learning

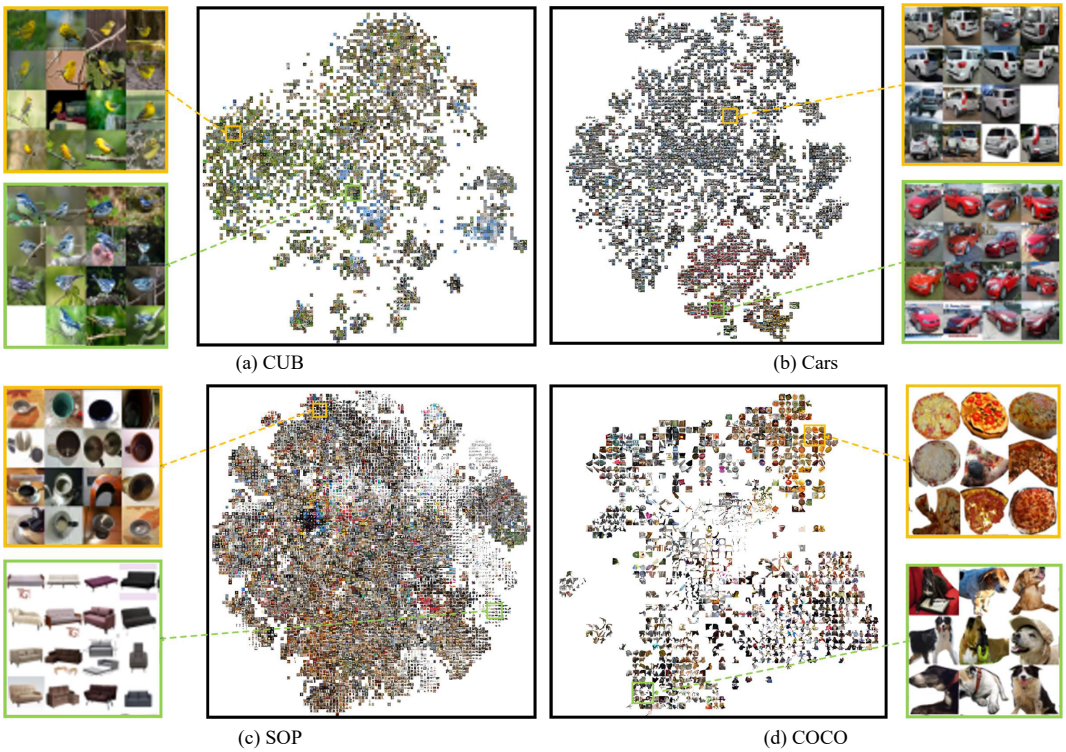

Fig. 3 t-SNE visualization of GC-UDML on (a) CUB, (b) Cars, (c) SOP, and (d) COCO datasets.

Fig. 4 Retrieval results of some example queries on CUB, Cars, SOP, and COCO datasets. The query images and the negative retrieved images are highlighted with yellow and red.

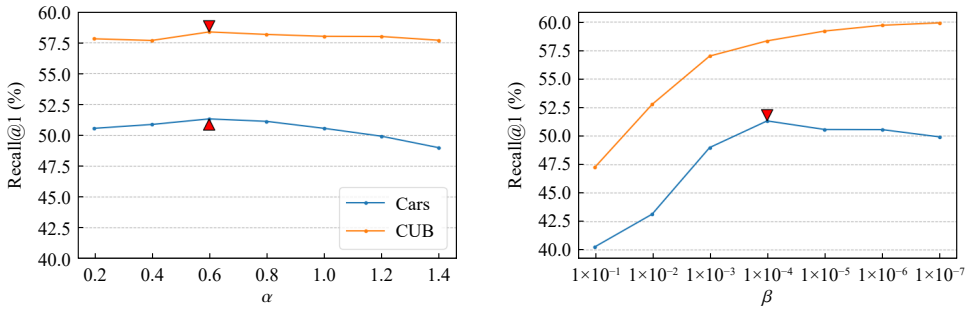

Fig. 5 Variation in Recall@1 concerning the hyper-parameters α (the rate of GKD Loss) and β (the rate of GCL Loss) on the CUB and Cars datasets. The default experimental setting is denoted by triangle markers.

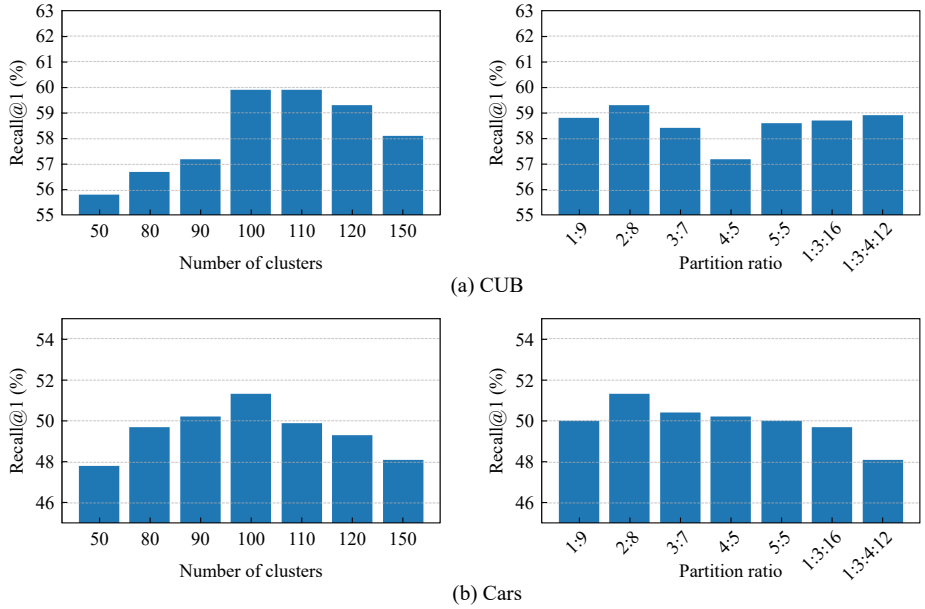

Fig. 6 Influence of the number of clusters determined by k-means (on the left) and the partitioning method (on the right) on Recall@1 on the (a) CUB and (b) Cars dataset.

Table 3 Experiment on number of clusters impact on our model when training SOP dataset. Bold values highlight the best results among the compared methods. G denotes GoogleNet, with the superscript number indicating the embedding layer's dimensionality.

(%)

Number of clusters	Architecture	Recall@1	Recall@2	Recall@4
100	G^{512}	52.4	68.8	79.8
500	G^{512}	60.1	74.0	83.9
1000	G^{512}	62.8	77.3	84.3
5000	G^{512}	64.9	78.1	86.4
10 000	G^{512}	**65.3**	**79.7**	**89.7**

network. Additionally, the rotation labels (0°, 90°, 180°, 270°) of the images are used to calculate the rotation prediction loss with the output of the rotation prediction network. This unsupervised learning approach is simple and efficient. However, the imbalanced distribution of pseudo-labels generated by k-means clustering is also evident.

We mention that during each epoch of the training process, UDML-SS samples p samples from each of the n pseudo-classes, resulting in a total of np samples as the training dataset for that particular epoch. This sampling method, referred to as the SS sampler, ensures that the network is trained on a small balanced subset of the dataset for each epoch. The second row in Table 4 proves the following point: when we remove this sampling mechanism and feed the entire dataset to the network, the performance of UDML-SS significantly declines.

We want to investigate if the GCL strategy can perform better on other methods that also use k-means clustering. Therefore, we attempt to incorporate the

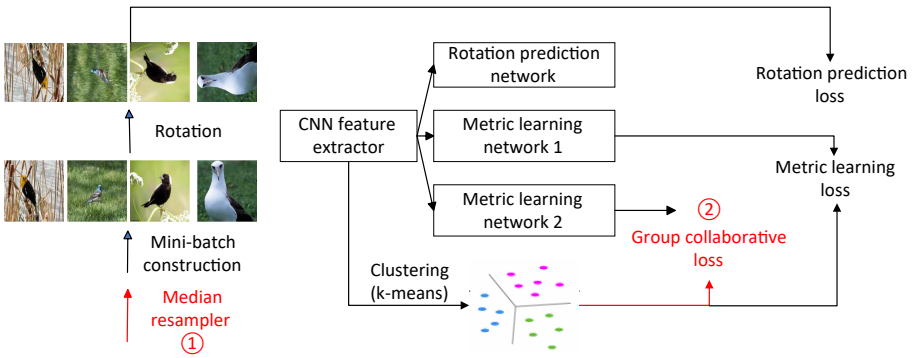

Fig. 7 Illustration of how the GCL strategy is applied to the UDML-SS method. The red arrows in the figure represent two components of the GCL strategy: ① median resampling method; ② group collaborative loss calculation. We separately incorporate these two components in our experiments to test the robustness of the GCL strategy on other methods.

Table 4 Performance of the GCL strategy on other k-means-based methods on CUB dataset. Specifically, the first and the second rows show the results without any components of the GCL strategy on UDML-SS. The third row displays the results with the median resampling strategy. The fourth row presents the results with the complete GCL strategy. Bold values highlight the best results among the compared methods.

(%)

Method	Architecture	Recall@1	Recall@2	Recall@4
UDML-SS[31]	G^{512}	54.7	66.9	77.4
W/o SS sampler	G^{512}	45.1	58.2	70.3
W/ ReSampler	G^{512}	53.2	65.8	77.1
W/ GCL strategy	G^{512}	**55.5**	**67.9**	**78.7**

GCL strategy into the UDML-SS method. As shown in Fig. 7, this step mainly consists of two parts: (1) before generating mini-batches in UDML-SS, we perform median resampling on the dataset; (2) we add an additional metric learning network (composed of two fully connected layers with an embedding size of 1024) and deploy the GCL classifier after this network to calculate the additional GCL Loss.

The experimental results are shown in Table 4. The second row represents the results when mini-batch generation in UDML-SS is not performed. The third row represents the results when mini-batch generation in UDML-SS is not performed, but the median resampling strategy is applied. The fourth row represents the results when the complete GCL strategy, including both Parts (1) and (2) mentioned above, is incorporated. GoogleNet with 512 dimension is used as backbone for both method in this experiment and the CUB dataset is used for testing. According to the second line, we find that in the UDML-SS method, if the sampling part is removed, Recall@1 of the results will decrease dramatically. However, when we use the median resampling strategy to replace the SS sampler, the results are much better compared to not using any sampler. Recall@1 improved by 1.5% when applying the complete GCL strategy.

5 Conclusion

In this paper, we introduce the group collaborative unsupervised deep metric learning method for feature embedding. Our experiments are conducted on image retrieval benchmarks, and the results demonstrate that the proposed method outperforms state-of-the-art methods, achieving an improvement in Recall@1 of up to 1.8%. Furthermore, we also verify that the GCL strategy works on UDML-SS method, achieving an improvement in Recall@1 of 1.5%.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 62202499) and the High Performance Computing Center of Central South University.

References

[1] Q. Zhou, Z. Zhou, Z. Bao, W. Niu and Y. Liu, IIN-FFD: Intra-inter network for face forgery detection. *Tsinghua Science and Technology*, vol. 29, no. 6, pp. 1839–1850, 2024.

[2] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, Sphereface: Deep hypersphere embedding for face recognition, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 6738–6746.

[3] S. Kim, M. Seo, I. Laptev, M. Cho, and S. Kwak, Deep metric learning beyond binary supervision, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp.

2283–2292.

[4] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, No fuss distance metric learning using proxies, in *Proc. 2017 IEEE Int. Conf. Computer Vision*, Venice, Italy, 2017, pp. 360–368.

[5] D. V. Devalraju and C. Chandra Sekhar, Uncertainty-guided metric learning without labels, in *Proc. 2025 IEEE/CVF Conf. Applications of Computer Vision*, Tucson, AZ, USA, pp. 7029–7038.

[6] S. Kan, Y. Deng, Y. Liang, L. Cen, Z. Qu, Y. Cen and Z. He, Unsupervised collaborative metric learning with mixed-scale groups for general object retrieval, arXiv preprint arXiv: 2403.10798, 2024.

[7] Y. Zhang and Y. Qin, Lightweight person re-identification network based on random color dropout and self-attention, *Tsinghua Science and Technology*, vol. 31, no. 2, pp. 976–992, 2026.

[8] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, Deep relative distance learning: Tell the difference between similar vehicles, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2167–2175.

[9] S. Kim, D. Kim, M. Cho, and S. Kwak, Self-taught metric learning without labels, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 7421–7431.

[10] X. Dong, T. Gan, X. Song, J. Wu, Y. Cheng, and L. Nie, Stacked hybrid-attention and group collaborative learning for unbiased scene graph generation, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 19405–19414.

[11] S. Ding, L. Lin, G. Wang, and H. Chao, Deep feature learning with relative distance comparison for person re-identification, *Pattern Recognit*, vol. 48, no. 10, pp. 2993–3003, 2015.

[12] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, ArcFace: Additive angular margin loss for deep face recognition, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 4685–4694.

[13] F. Wang and H. Liu, Understanding the behaviour of contrastive loss, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 2495–2504.

[14] W. Zhao, Y. Rao, Z. Wang, J. Lu, and J. Zhou, Towards interpretable deep metric learning with structural matching, in *Proc. 2021 IEEE/CVF Int. Conf. Computer Vision*, Montreal, Canada, 2021, pp. 9867–9876.

[15] J. Deng, J. Guo, J. Yang, A. Lattas, and S. Zafeiriou, Variational prototype learning for deep face recognition, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 11901–11910.

[16] E. W. Teh, T. DeVries, and G. W. Taylor, ProxyNCA++: Revisiting and revitalizing proxy neighborhood component analysis, in *Proc. 16th European Conf. Computer Vision*, Glasgow, UK, 2020, pp. 448–464.

[17] M. Kirchhof, K. Roth, Z. Akata, and E. Kasneci, A non-isotropic probabilistic take on proxy-based deep metric learning, in *Proc. 17th European Conf. Computer Vision*, Tel Aviv, Israel, 2022, pp. 435–454.

[18] J. Yan, L. Luo, C. Deng, and H. Huang, Unsupervised hyperbolic metric learning, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 12460–12469.

[19] S. Kan, Y. Cen, Y. Li, V. Mladenovic, and Z. He, Relative order analysis and optimization for unsupervised deep metric learning, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 13994–14003.

[20] Y. Li, S. Kan, J. Yuan, W. Cao, and Z. He, Spatial assembly networks for image representation learning, in *Proc. 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Nashville, TN, USA, 2021, pp. 13871–13880.

[21] Y. Li, S. Kan, and Z. He, Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss, in *Proc. 16th European Conf. Computer Vision*, Glasgow, UK, 2020, pp. 141–157.

[22] M. Ye, X. Zhang, P. C. Yuen, and S. F. Chang, Unsupervised embedding learning via invariant and spreading instance feature, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 6203–6212.

[23] M. Ye and J. Shen, Probabilistic structural latent representation for unsupervised embedding, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2020, pp. 5456–5465.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, A simple framework for contrastive learning of visual representations, in *Proc. 37th Int. Conf. Machine Learning*, Virtual Event, 2020, p. 149.

[25] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, Momentum contrast for unsupervised visual representation learning, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2020, pp. 9726–9735.

[26] S. Gidaris, P. Singh, and N. Komodakis, Unsupervised representation learning by predicting image rotations, in *Proc. 6th Int. Conf. Learning Representations*, Vancouver, Canada, 2018. https://dblp.org/db/conf/iclr/iclr2018.html#GidarisSK18.

[27] M. Noroozi and P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in *Proc. 14th European Conf. Computer Vision*, Amsterdam, The Netherlands, 2016, pp. 69–84.

[28] R. Zhang, P. Isola, and A. A. Efros, Colorful image colorization, in *Proc. 14th European Conf. Computer Vision*, Amsterdam, The Netherlands, 2016, pp. 649–666.

[29] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, Context encoders: Feature learning by inpainting, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2536–2544.

[30] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, Deep

clustering for unsupervised learning of visual features, in *Proc. 15th European Conf. Computer Vision*, Munich, Germany, 2018, pp. 139–156.

[31] X. Cao, B. C. Chen, and S. N. Lim, Unsupervised deep metric learning via auxiliary rotation loss, arXiv preprint arXiv: 1911.07072, 2019.

[32] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, *Caltech-UCSD Birds 200*. CNS-TR-2010-001, Pasadena, CA, USA: California Institute of Technology, 2010.

[33] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, 3D object representations for fine-grained categorization, in *Proc. 2013 IEEE Int. Conf. Computer Vision Workshops*, Sydney, Australia, pp. 554–561, 2013.

[34] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, Microsoft COCO: Common objects in context, in *Proc. 13th European Conf. Computer Vision*, Zurich, Switzerland, 2014, pp. 740–755.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, in *Proc. 2015 IEEE Conf. Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 1–9.

[36] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proc. 32nd Int. Conf. Machine Learning*, Lille, France, 2015, pp. 448–456.

[37] B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J. W. Ha, AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights, in *Proc. 9th Int. Conf. Learning Representations*, Virtual Event, 2021. https://dblp.org/db/conf/iclr/iclr2021.html#HeoCOHYKUH21.

[38] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, Mining on manifolds: Metric learning without labels, in *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 7642–7651.

Zeqian Chen received the BEng degree in automation from Central South University, Changsha, China, in 2019. He is currently pursuing the MEng degree in computer science and engineering at School of Computer Science and Engineering, Central South University, Changsha, China. His research interests include metric learning and image processing.

Shichao Kan received the PhD degree in information science from the Institute of Information Science, Beijing Jiaotong University, Beijing, China, in 2021. From 2019 to 2020, he was a visiting scholar at University of Missouri–Columbia, Columbia, MO, USA. He is currently a professor at School of Computer Science and Engineering, Central South University, Changsha, China. His research interests include multimodal large models, computer vision, machine learning, deep learning, and artificial intelligence.

Min Li received the MS and PhD degrees from Central South University, Changsha, China, in 2004 and 2008, respectively. Since 2008, she has been working at Central South University, Changsha, China. Currently, she is the dean of School of Computer Science and Engineering, Central South University, Changsha, China. She has published more than 100 academic papers in international journals and conferences. Her main research direction is bioinformatics and data mining.