Deep Metric Learning with Self-Supervised Ranking

Zheren Fu[1*], **Yan Li**[2*], **Zhendong Mao**[1†], **Quan Wang**[3], **Yongdong Zhang**[1]
[1]University of Science and Technology of China, Hefei, China
[2]Kuaishou Technology, Beijing, China
[3]Beijing Research Institute, University of Science and Technology of China, Beijing, China
fzr@mail.ustc.edu.cn, liyan@kuaishou.com, {zdmao, zhyd73}@ustc.edu.cn, quanwang1012@gmail.com

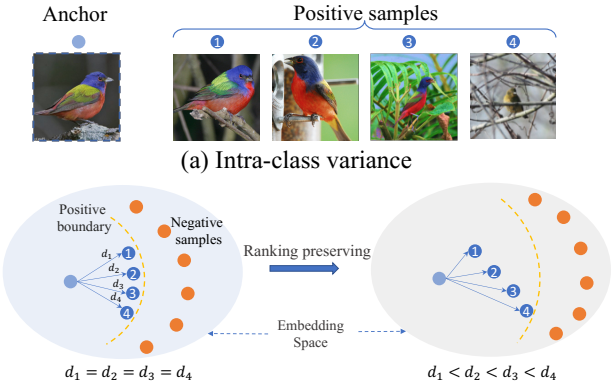
Abstract

Deep metric learning aims to learn a deep embedding space, where similar objects are pushed towards together and different objects are repelled against. Existing approaches typically use inter-class characteristics, *e.g.*, class-level information or instance-level similarity, to obtain semantic relevance of data points and get a large margin between different classes in the embedding space. However, the intra-class characteristics, *e.g.*, local manifold structure or relative relationship within the same class, are usually overlooked in the learning process. Hence the data structure cannot be fully exploited and the output embeddings have limitation in retrieval. More importantly, retrieval results lack in a good ranking. This paper presents a novel self-supervised ranking auxiliary framework, which captures intra-class characteristics as well as inter-class characteristics for better metric learning. Our method defines specific transform functions to simulates the local structure change of intra-class in the initial image domain, and formulates a self-supervised learning procedure to fully exploit this property and preserve it in the embedding space. Extensive experiments on three standard benchmarks show that our method significantly improves and outperforms the state-of-the-art methods on the performances of both retrieval and ranking by 2%-4%.

Introduction

Deep Metric learning aims to learn effective distance or similarity measures between arbitrary objects with the success of deep learning. It's a crucial topic in computer vision and has been applied to a variety of tasks, including face verification (Liu et al. 2017), person re-identification (Xiao et al. 2017), and fine-grained image retrieval (Qian et al. 2019). The paradigm of deep metric learning aims to project data onto an embedding space, where the embeddings of visual-semantically similar samples (*e.g.*, images of the same class) are close together, while dissimilar ones (*e.g.*, images from different classes) are far apart from each other. Most of the deep metric learning approaches focus on designing objective functions which can be defined in terms of pairwise similarities between an anchor (regarded as a query in retrieval) and corresponding positive/negative samples (Sohn 2016; Ge 2018). Their performances heavily rely on sampling strategies (Wu et al. 2017; Suh et al. 2019), pairs weighting (Wang et al. 2019a,b), and examples generation (Zhao et al. 2018; Zheng et al. 2019) to mine informative samples. Recent works improve the model through using boost-like policies, including attentions (Kim et al. 2018), features separation (Sanakoyeu et al. 2019), and self-supervised strategies (Roth et al. 2019; Wang et al. 2020). These ensemble approaches are built on standard objective functions to distinguish negative samples. Generally, existing methods concentrate on learning class-discriminative embeddings by maximizing inter-class variance.

However, these deep metric learning approaches completely dismiss intra-class variance in the embedding space, *i.e.*, relative distance between positive samples and the anchor or local manifold structure. They regard all positive samples equally since the lack of annotations and try their best to discriminate positive and negative samples, while



(a) Intra-class variance

(b) Previous method (c) Our method

Figure 1: Illustration of the limitation of previous works. Best retrieval result ranking should be $1, 2, 3, 4$, but previous method may return a random order like $2, 4, 1, 3$ since they overlook the relationship of different positive samples. (a) Inherent intra-class variances from the same category. (b) Previous methods only maximize the inter-class variance but cannot preserve the intra-class properties. (c) Our method captures intra-class variance by keeping their ranking information to achieve better retrieval result.

*Equal contribution, this work was done during Zheren Fu's internship at Kuaishou. †Corresponding author.

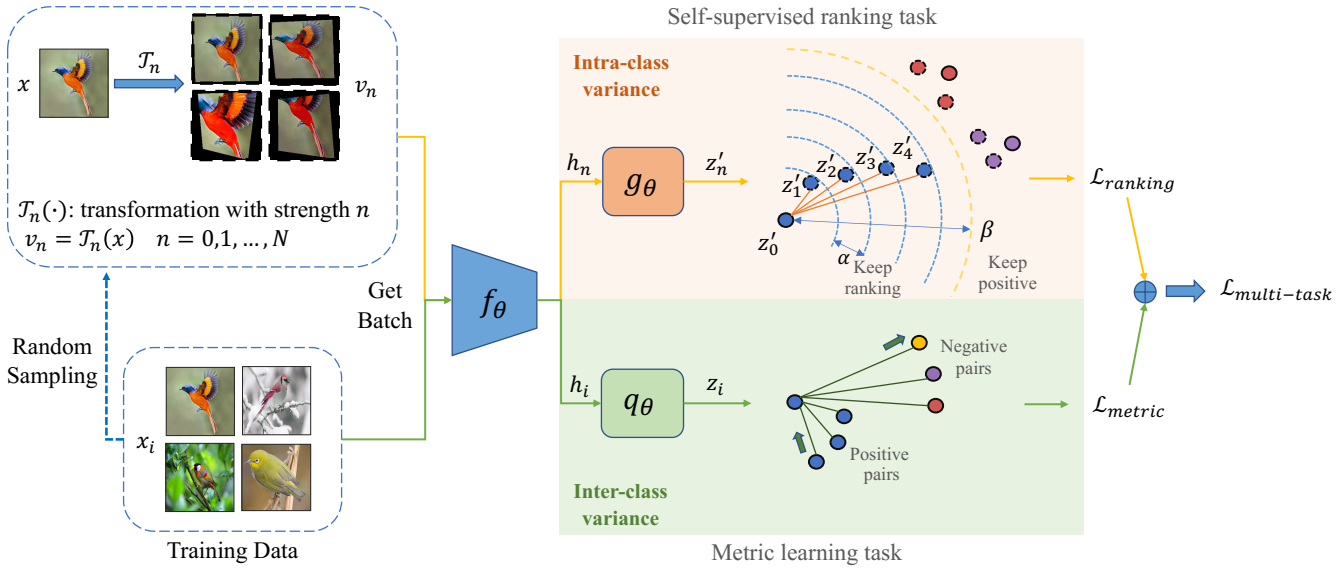


Figure 2: Overview of our approach. We use one backbone network f_θ and two head networks g_θ, q_θ to optimize two loss functions jointly. Primary metric learning loss $\mathcal{L}_{metric}$ is built in embeddings space z_i. Self-supervised surrogate objective function $\mathcal{L}_{ranking}$ is solved in embedding space z'_n to support metric learning by preserving intra-class variance and ranking information, which are generated by transform functions $\mathcal{T}_n$ of different strengths n with no extra manual annotations.

the ranking of different positive samples is discarded totally. In a word, previous methods mainly concentrate on how to maximize the inter-class variance and increase the margin of different class-neighbors in the embedding space, while the intra-class variance is minimized and local structure is destroyed unconsciously. Fig. 1(a) shows the latent intra-class variance within a certain class, without considering this property, previous methods would learn a less efficient embedding space, shown in Fig. 1(b), as compared with the optimal result in Fig. 1(c). In summary, previous methods are not able to fully exploit the intra-class variance, which is also very important to learn better embeddings.

In this work, we propose a novel self-supervised ranking learning auxiliary framework, which can be easily integrated with existing metric learning techniques. The overall framework of our method is illustrated in Fig. 2. In inter-class characteristics mining phase, any existing metric learning algorithms can be applied. Meanwhile, we define a standard criterion to generate and measure intra-class variance, and propose a self-supervised learning procedure to learn to preserve the corresponding ranking information in the embedding space. In this way, the learned embeddings not only maintain inter-class separability but also discriminate subtle intra-class variance, leading to a better global and local embedding structure for retrieval and ranking. Our contributions are summarized as follows: (1) We design a typical paradigm to preserve local structure by measuring intra-class variance for deep metric learning. (2) We propose a general self-supervised auxiliary framework with specific transform functions and ranking preserving strategy. It can not only capture intra-class properties, but also learn discriminative semantic embeddings. (3) Evaluation results on three benchmark datasets demonstrate that our method can improve and outperform the performances of state-of-the-art approaches on both retrieval and ranking by 2%-4%.

To our best knowledge, our method is the first self-supervised framework which capture intra-class variance for deep metric learning.

Related Work

Deep Metric Learning

Deep metric learning is a fundamental algorithm to learn similarity measure between objects with the advent of deep neural networks. A large variety of loss functions have been proposed these years and can be categorized into two classes, pair-based and proxy-based. Pair-based methods (Wu et al. 2017; Wang et al. 2019b) are built on a group of pairwise distances between the anchor and positive or negative samples. These losses can mine rich structural relationships among data by examining multiple sample pairs. But their performances are heavily dependent on hard sampling strategies (Suh et al. 2019) or negative examples generation (Zheng et al. 2019) to select more valuable samples in a mini-batch. The proxy-based losses (Qian et al. 2019; Deng et al. 2019) utilize proxy embeddings as the class-related representation and a part of network parameters. They encourage each image as the anchor point to be close to the proxies of the same class and far away from these of different classes, instead of other image points. They reduce the computational complexity and obtain faster convergence when the number of classes is small.

Based on the above loss functions, ensemble methods (Opitz et al. 2018; Kim et al. 2018) are proposed to boost the performance using advanced techniques. For example, Divide (Sanakoyeu et al. 2019) uses the divide-and-conquer strategy to learn on partitional embedding spaces. MIC

(Roth et al. 2019) combines inter-class discriminative features with characteristics shared across classes. HORDE (Jacob et al. 2019) includes pooling feature representations as well as the higher-order moments. XBM (Wang et al. 2020) uses the memory mechanism to expand large batch sizes.

Learning-to-rank strategy is widely used in metric learning. They learn the ranking objective functions by minimizing ranking quality measures, such as Recall, R-precision, and MAP (Järvelin et al. 2002), which are used to evaluate the performance of ranking results. The loss functions can mainly be divided into three categories with different motivations and formulations (Chen et al. 2009). The pointwise loss considers the absolute correlation of a single returned result under a given query. The pairwise loss considers the relative correlation between two results under a certain query. The listwise loss considers the list of output results, and directly optimizes the whole list to make it as close to the best ranked list as possible.

Self-supervised Learning

Self-supervised learning (SSL) has made great development in the past years. It aims to learn transferable embeddings without relying on manual annotations and can be used for various downstream tasks as pre-training. The supervisory signals come from diverse well-designed pretext tasks, including colorization (Zhang et al. 2017), rotation prediction (Gidaris et al. 2018), and clustering (Caron et al. 2018). Lately, contrastive based self-supervised methods (He et al. 2020) achieve strong performances close to the supervised baseline. They train models by reducing the distance between representations of augmented views of the same image (as positive pairs) and increasing the distance between representations of different augmented views from different images (as negative pairs). They focus on the semantic relationship of pairwise images and have common ground with metric learning methods. Besides, it's has been proved that SSL is effective for some specific problems such as semi-supervised learning (Zhai et al. 2019) and crowd counting (Liu et al. 2019). Inspired by this, researchers start to make use of SSL for metric learning (Roth et al. 2019; Wang et al. 2020), which further enhance the discrimination of inter-class variance. In contrast, our framework uses the self-supervised method to measure and learn intra-class variance.

Proposed Method

First we introduce the preliminaries of deep metric learning. Let $\mathcal{X} = \{x_1, ..., x_K\}$ denotes a set of training images in the original RGB space, and y_i is the corresponding label of x_i, $y_i \in [1, 2, \ldots, C]$. The deep metric learning models are comprised of two networks: a representation encoder f_θ and a head encoder q_θ with θ the parameters. f_θ is a Convolutional Neural Network(CNN) with global pooling and q_θ is a fully-connected layer. Through these networks, we can get two-stage embeddings, $h_i = f_\theta(x_i) \in \mathbb{R}^F$ and $z_i = q_\theta(h_i) \in \mathbb{R}^D$. The goal of deep metric learning is to learn f_θ and q_θ jointly such that embeddings of similar images, z_i, are close to each other while those of dissimilar ones are far apart. Formally, the distance between two images x_i, x_j in the embedding space is defined as $d_E(,)$:

$$d_E(x_i, x_j) = d(z_i, z_j) = \|z_i - z_j\|_2 . \tag{1}$$

where $d(,)$ is the Euclidean distance and various kinds of loss functions can be used to learn the embeddings.

Measure of Intra-class Variance

Definition of intra-class variance Intra-class variance is the diverse visual representations of the semantic similar object such as scale, color, viewpoint, and so on. It is fine-grained detail changes under the certain class contrast to inter-class variance. Given an image x, its positive sample $x_p \in \mathcal{X}_p$ and negative sample $x_n \in \mathcal{X}_n$, the intra-class and inter-class variances are defined as $d_E(x, x_p)$ and $d_E(x, x_n)$ respectively. Most existing methods focus on increasing the margin between $d_E(x, x_p)$ and $d_E(x, x_n)$, so as to ensure the following constraint hold:

$$\max_{x_p \in \mathcal{X}_p} d_E(x, x_p) < \min_{x_n \in \mathcal{X}_n} d_E(x, x_n). \tag{2}$$

Here we take Triplet loss (Schrof et al.2015) as an example, since it is a fundamental part of many loss functions (Sohn 2016; Wu et al. 2017; Wang et al. 2019a):

$$\mathcal{L}_{metric}(\mathcal{L}_{triplet}) = [d_E(x, x_p) - d_E(x, x_n) + \alpha]_+, \tag{3}$$

where $[x]_+$ means $max(0, x)$. Triplet loss makes the anchor point x closer to positive points x_p than other negative points x_n by a margin α ($y_a = y_p \neq y_n$), thus only inter-class variance (margin between difference classes) is optimized, while the intrinsic intra-class variance is ignored. For a image x with its positives x_{p1}, x_{p2}, to learn better representations, z, z_{p1}, z_{p2}, in the embedding space, it is desirable that the following constraint also hold (Zhang et al. 2014):

$$\text{if } d_M(x, x_{p1}) < d_M(x, x_{p2}), \text{ then } d(z, z_{p1}) < d(z, z_{p2}). \tag{4}$$

where d_M is the measure of intra-class variance in the original image domain. Eq (4) means the relationships between intra-class variance, ***e.g.***, relative rankings, in the embedding space are consistent with those in the original image domain. Current human-labeled signal, ***e.g.***, class label or pairwise label, treats images from the same category equally, ***i.e.***, two images are similar, it not able to further distinguish between similar images. To find a proper metric d_M to quantify the intra-class variance, we start with the help of self-supervised learning strategy.

Simulative transform functions Specifically, we define a general transform function T_n, which simulates the various changes of intra-class properties, n is a positive integer and represents transformation intensity, where a larger n indicates a larger quantitative variance:

$$T_n = \{T_I \mid T_{scale}, T_{color}, T_{viewpoint}, ...\}, \tag{5}$$

T_n is an expandable set for any proper transforms T_I and the controllable T_n can quantify intra-class variance and measure them approximately. For image x and two positive integer i, j, we hold:

$$\text{if } i < j, \text{ then } d_M(x, T_i(x)) < d_M(x, T_j(x)). \tag{6}$$

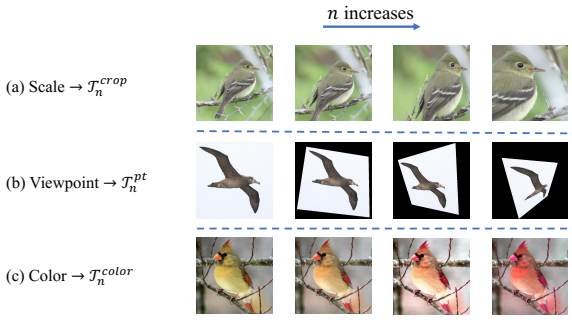


Figure 3: Examples of intra-class variance and relevant transform functions. Including various changes of (a) scale, (b) viewpoint, and (c) color, corresponding to Random Crop, Perspective Transform, and Color Jitter respectively.

We choose three representative intra-class variances and relevant transforms to compose final transform $\mathcal{T}_n \subseteq T_n$. First we choose the spatial transform random crop (with resizing) to reflect diverse scales, as shown in Fig. 3(a). I_{ij} means the RGB value of image I in position (i, j), R_n is the cropped region whose size is inversely proportional to n.

$$\mathcal{T}_n^{crop}(I) = I_{ij} \cdot \mathbb{I}_{(i,j)\in R_n}, \tag{7}$$

Then we use the perspective transform to imitate the variety of viewpoint, as shown in Fig. 3(b). n reflects the distortion scale which can be determined by the relative distance between start-points and end-points. $I = (I_x, I_y, I_z)$ means the original coordinate, and A_n is coefficients matrix:

$$\mathcal{T}_n^{pt}(I) = A_n \cdot I, \tag{8}$$

In addition, we apply the appearance transformation, color jitter, which randomly and sequentially changes the brightness, contrast, and saturation of an image to express the various color distributions, as shown in Fig. 3(c). n reflects adjusting factor to control the jitter range:

$$\mathcal{T}_n^{color}(I) = CJ_n(I; bri, con, sat), \tag{9}$$

Finally, we cascade the above transforms into a complete transform function $\mathcal{T}_n$ to generate intra-class variance.

$$\mathcal{T}_n = \mathcal{T}_n^{crop} * \mathcal{T}_n^{pt} * \mathcal{T}_n^{color}. \tag{10}$$

Self-supervised Ranking Framework

Now we introduce our self-supervised ranking auxiliary framework. First, we uniformly sample an image x from $\mathcal{X}$. With the transform function $\mathcal{T}_n$, we get a number of augmented views from x by applying the transform with different strength: $\{v_n = \mathcal{T}_n(x)\}_{n=0}^{N}$, where $\mathcal{T}_0(x) = x$ and larger n indicates larger transform strength. Then we use the same backbone network f_θ to encode images and a multi-layer perception (MLP) g_θ to establish a new embedding space. So we get corresponding embeddings $h_n = f_\theta(v_n) \in \mathbb{R}^F$ and $z'_n = g_\theta(h_n) \in \mathbb{R}^D$, as shown in Fig 2.

For embeddings z'_n from different augmented views v_n, it is required that the embedding of the original image x, z'_0, should be closer to the embeddings of augmented views with lower strength than those with higher strength, which is:

$$\text{if } i < j, \text{ then } d(z'_0, z'_i) < d(z'_0, z'_j). \tag{11}$$

This ranking preserving objective is formulated based on the pairwise ranking loss (Schrof et al.2015). Without loss of generality, we use similarity $S(,)$ in our following introduction. The similarity of a image pair with weaker augmentations should be larger than that with stronger augmentations in the embedding space by a fixed margin α.

$$\mathcal{L}_{base} = \left[S(z'_0, z'_j) - S(z'_0, z'_i) + \alpha\right]_+, \quad \text{when } i < j. \tag{12}$$

Then we use the listwise ranking loss to integrate all augmented views:

$$\mathcal{L}_{list} = \frac{1}{N-1}\sum_{n=1}^{N-1}[S(z'_0, z'_{n+1}) - S(z'_0, z'_n) + \alpha]_+, \tag{13}$$

In a mini-batch, we denote the m-th image as x_m ($m = 1, 2, ..., M$) and the related augmented views as $v_{m,n}$. Besides, we use LogSumExp and SoftPlus functions to smooth Eq. (13). After summing over all x_m, $\mathcal{L}_{list}$ becomes:

$$\mathcal{L}_{sort} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{s}log[1 + \sum_{n=1}^{N-1} e^{s(-\mathcal{S}_{m,n}+\mathcal{S}_{m,n+1}+\alpha)}]. \tag{14}$$

where $\mathcal{S}_{m,n}$ is $S(z'_{m,0}, z'_{m,n})$ and s is the scale factor.

Eq. (13) has a limitation that gradients are fixed, the value is ± 1 when a training pair violates the constraint and 0 otherwise. The loss cannot mine any informative sample pairs (Wang et al. 2019a) and leads to the trivial samples (Wang et al. 2019b) during training. By contrast, the gradient of Eq. (14) is weighted according to the relative hardness, which is the degrees of strength that a pair violates the constraint. As shown in Eq. (15), a harder pair can get larger gradient magnitude, where $h_{m,n} = e^{s(-\mathcal{S}_{m,n}+\mathcal{S}_{m,n+1}+\alpha)}$.

$$\frac{\partial \mathcal{L}_{sort}}{\partial \mathcal{S}_{m,n}} = \frac{h_{m,n-1} - h_{m,n}}{1 + \sum_{j=1}^{N-1} h_{m,j}}. \tag{15}$$

Meanwhile, since $v_{m,n}$ is generated by certain transformations, it is still a positive sample of image x_m. Therefore, we should add the constraint for positive pairs to ensure $\mathcal{S}_{m,n}$ is larger than a boundary β. We also use the smooth version of the pointwise loss (Yi et al. 2014) to control the relative hardness:

$$\mathcal{L}_{pos} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{s}log[1 + \sum_{n=1}^{N} e^{-s(\mathcal{S}_{m,n}-\beta)}]. \tag{16}$$

With the weighted sum of Eq (14) and Eq (16), where λ controls the balance, we reach the self-supervised list-wise ranking loss:

$$\mathcal{L}_{ranking} = \mathcal{L}_{sort} + \lambda\mathcal{L}_{pos}, \tag{17}$$

Our auxiliary framework is independent of the choice of metric learning losses, which is explained in Section 3.1. We just incorporate Eq. (17) into Eq. (3) and train the entire network (f_θ, g_θ, q_θ). The overall objective $\mathcal{L}$ makes up of a general metric learning loss and our proposed self-supervised ranking learning loss, where γ weights the importance. The whole training procedure is outlined in Alg. 1.

$$\mathcal{L}_{multi-task} = \mathcal{L}_{metric} + \gamma\mathcal{L}_{ranking}. \tag{18}$$

Algorithm 1 Model training process with our method

Input:
images X, class labels Y,
neural networks $f_\theta, g_\theta, q_\theta$,
hyper-parameters $\alpha, \beta, s, \lambda, \gamma, p_{task}$
Output:
network parameters θ $(\theta_f, \theta_g, \theta_q)$

$epoch \leftarrow 0$
while Not Converged **do**
  repeat
    $x, y \leftarrow MiniBatch(X, Y)$
    $z \leftarrow Embedding(x; f_\theta, q_\theta)$
    $Compute\ \mathcal{L}_{metric}(z, y)$
    $\theta_f, \theta_q \leftarrow Backward(\mathcal{L}_{metric})$
    if $p < p_{task}, p \sim U(0, 1)$ **then**
      $v_n \leftarrow \mathcal{T}_n(x)$
      $z'_n \leftarrow Embedding(v_n; f_\theta, g_\theta)$
      $Compute\ \mathcal{L}_{ranking}(z'_n)$
      $\theta_f, \theta_g \leftarrow Backward(\mathcal{L}_{ranking})$
    end if
  until Epoch End
  $epoch \leftarrow epoch + 1$
end while

Discussion

Since we hold on an auxiliary self-supervised task for training, the extra runtime and computation cost need to be considered objectively. Our proposed ranking loss, $\mathcal{L}_{ranking}$, belongs to general pair-based loss, which usually has high computational complexity, *e.g.*, $O(M^2)$ for Contrastive loss (Chopra et al. 2005) and Triplet loss (Schrof et al. 2015) with sampling strategies, $O(M^3)$ for Lifted Structure (Oh Song et al. 2016) and N-pair (Sohn 2016), with mini-batch size M. By contrast, the complexity of $\mathcal{L}_{ranking}$ is only $O(M)$, as the number of transformations is limited (4 in our experiments). Low time consumption and fast convergence speed are advantageous in the training stage.

Meanwhile, our training stage alternates between the metric learning task and the self-supervised task. The latter requires extra forward/backward processes and parameters for the model. In fact, we execute the auxiliary optimizing process by proper probability p_{task} every iteration thus the additional runtime is acceptable. The parameters of the MLP g_θ are negligible, as compared with the backbone network. It is also worth noting that our method does not increase parameters and runtime in the inference stage.

Last but not least, as compared with other multi-task auxiliary method MIC (Roth et al. 2019), we don't use extra operation modules, like gradient reversal layer, which brings performance instability and parameter complexity.

Implementation Details

We use PyTorch (Paszke et al. 2019) to implement our method on a single GTX 1080Ti GPU with 11GB memory. ResNet50 (He et al. 2016) with Global Max Pooling pretrained on ImageNet (Russakovsky et al. 2015) is used as backbone network f_θ. We replace last layer with a randomly initialized fully-connected layer q_θ for metric learning. Besides, an MLP with a 512-dim hidden layer g_θ is added to solve the self-supervised task. The output embeddings are L_2 normalized when computing similarity and the dimension (D) is 128 or 512. The input images are first resized to 256 $\times$ 256, then cropped to 224 $\times$ 224. For training, we use random crop and random horizontal flips for data augmentation. For testing, we only use the single-center crop. We use AdamW (Loshchilov et al. 2017) optimizer with $4e^{-4}$ weight decay and 120 batch size. The initial learning rate is 10^{-4} and scaled up 10 times on output layers for faster convergence. Mini-batches are constructed with the balanced sampler. The hyper-parameters setting is: $\alpha = 0.05, \beta = 0.5, s = 12, \lambda = 1.0, \gamma = 0.8, p_{task} = 0.8, M = 20, N = 4$. We evaluate our framework on three metric learning losses, Triplet loss (Schrof et al. 2015), Margin loss (Wu et al. 2017), MS loss (Wang et al. 2019a), whose parameters are the default.

Experiments

Given a query, retrieval can be separated into two stages, first distinguishes positive neighbors, then ranks them according to the degree of similarities. We evaluate the performances of our method in terms of retrieving and ranking.

Datasets

We evaluate our proposed method on three widely-used datasets following the standard protocol (Oh Song et al. 2016). **(1) CUB-200-2011 (CUB)** (Wah et al. 2011) contains 11,788 images of 200 species of birds. We use 5,864 images of its first 100 classes for training and 5,924 images of the remaining classes for testing. **(2) Cars-196 (Cars)** (Krause et al. 2013) contains 16,185 images of 196 car models. We use 8,054 images of its first 98 classes for training and 8,131 images of the other classes for testing. **(3) Stanford Online Products (SOP)** (Oh Song et al. 2016) contains 120,053 online product images of 22,634 categories sold on eBay.com. We use 59,551 images of 11,318 classes for training and 60,502 images of the rest for testing.

Comparison on Retrieving Results

First, we evaluate the retrieval performance of our method in terms of Recall@K and NMI (Schütze et al. 2008). Our framework can help models discriminate inter-class variance because local structures of the embedding space are fully exploited and the process is stimulative to learn class-related boundaries. Tab. 1 show that our method brings considerable improvement. Triplet (Schrof et al. 2015) and Margin loss (Wu et al. 2017) get surprising promotions with our framework, 3%-6% R@1 gains on all datasets. MS loss (Wang et al. 2019a) also obtains a 4% gain on Cars. In a word, our model is a universal auxiliary framework for deep metric learning regardless of images category and loss functions.

We also compare our approach with the state-of-the-art deep metric learning methods. We list the performances with corresponding configurations since the backbone and embedding dimension (generally, larger is better) can affect

Method	Dim	CUB-200-2011					Cars-196					Stanford Online Products				
		R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	R@1000	NMI
Triplet (Schrof et al. 2015)	64	42.6	55.0	66.4	77.2	55.4	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	-	89.5
Triplet (ReImp)	128	59.6	71.2	80.9	88.3	65.5	69.7	79.5	86.5	91.5	61.7	73.5	86.8	94.4	98.1	89.2
Ours (Triplet)	128	**62.4**	**73.1**	**82.8**	**89.4**	**67.6**	**73.3**	**82.0**	**88.2**	**92.5**	**64.2**	**75.8**	**88.6**	**95.2**	**98.3**	**89.7**
Margin (Wu et al. 2017)	128	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	**70.3**	72.7	86.2	93.8	98.0	**90.7**
Margin (ReImp)	128	63.4	75.0	84.2	91.0	68.8	79.5	87.5	92.4	95.5	68.0	74.6	87.6	94.6	98.2	89.3
Ours (Margin)	128	**66.5**	**76.8**	**85.5**	**91.0**	**69.7**	**84.5**	**90.2**	**93.7**	**96.1**	70.1	**77.9**	**89.5**	**95.4**	**98.4**	90.1
MS (Wang et al. 2019a)	64	57.4	69.8	80.0	87.8	-	77.3	85.3	90.5	94.2	-	74.1	87.8	94.7	98.2	-
MS (ReImp)	128	63.3	74.8	83.7	90.3	67.8	81.6	89.0	93.6	96.4	69.8	77.2	89.4	95.5	**98.5**	89.9
Ours (MS)	128	**65.6**	**76.5**	**85.2**	**90.8**	**69.4**	**85.5**	**91.4**	**94.7**	**97.0**	**70.5**	**78.1**	**89.7**	**95.6**	**98.5**	**90.2**

Table 1: Retrieval accuracy on three standard datasets. $ReImp$ indicates our re-implementation with official codes and settings. We use ResNet50 (He et al. 2016) as backbone and 128 as embedding dimension.

Method	Setting	CUB-200-2011					Cars-196					Stanford Online Products				
		R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	R@1000	NMI
HDC (Oh Song et al. 2017)	G^{384}	53.6	65.7	77.0	85.6	-	73.7	83.2	89.5	93.8	-	69.5	84.4	92.8	97.7	-
A-BIER (Opitz et al. 2018)	G^{512}	57.5	68.7	78.3	86.2	-	82.0	89.0	93.2	96.1	-	74.2	86.9	94.0	97.8	-
ABE (Kim et al. 2018)	G^{512}	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	98.2	-
XBM (Wang et al. 2020)	G^{512}	61.9	72.9	81.2	88.6	-	80.3	87.1	91.9	95.1	-	77.4	89.6	95.4	98.4	-
HTL (Ge 2018)	BN^{512}	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	98.4	-
RLL-H (Wang et al. 2019b)	BN^{512}	57.4	69.7	79.2	86.9	63.6	74.0	83.6	90.1	94.1	65.4	76.1	89.1	95.4	-	89.7
SoftTriple (Qian et al. 2019)	BN^{512}	65.4	76.4	84.5	90.4	69.3	84.5	90.7	94.5	96.9	70.1	78.3	90.3	95.9	-	**92.0**
Circle (Sun et al. 2020)	BN^{512}	66.7	77.4	86.2	91.2	-	83.4	89.8	94.1	96.5	-	78.3	90.5	96.1	98.6	-
MIC (Roth et al. 2019)	R^{128}	66.1	76.8	85.6	-	69.7	82.6	89.1	93.2	-	68.4	77.2	89.4	95.6	-	90.0
Divide (Sanakoyeu et al. 2019)	R^{128}	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	98.1	90.2
PADS (Roth et al. 2020)	R^{128}	67.3	78.0	85.9	-	69.9	83.5	89.7	93.8	-	68.8	76.5	89.0	95.4	-	89.9
RaMBO (Rolínek et al. 2020)	R^{512}	63.5	74.8	84.1	90.4	-	-	-	-	-	-	77.8	90.1	95.9	**98.7**	-
Ours (Margin)	R^{128}	66.5	76.8	85.5	91.0	69.7	84.5	90.2	93.7	96.1	70.1	77.9	89.5	95.4	98.4	90.1
Ours† (Margin)	R^{512}	**68.2**	**78.1**	**86.5**	**91.6**	**70.3**	**87.7**	**92.5**	**95.4**	**97.3**	**72.1**	**78.6**	**90.6**	**96.2**	**98.7**	90.5

Table 2: Comparison with the state-of-the-art deep metric learning methods. Backbone networks of the models are denoted by abbreviations: G–GoogleNet (Szegedy et al. 2015), BN–Inception (Ioffe et al. 2015), R–ResNet50 (He et al. 2016). Superscripts in the networks denote embedding sizes. † indicates models using larger embedding size (512).

Method	CUB		Cars		SOP	
	MAP	RP	MAP	RP	MAP	RP
Contrastive (Chopra et al. 2005)	32.9	34.5	32.5	34.0	42.6	41.7
N-pair (Sohn 2016)	31.2	32.6	30.3	31.9	38.8	38.2
ProxyNCA (Attias et al. 2017)	33.0	34.6	31.2	33.1	43.0	42.2
Triplet-semi (Schrof et al. 2015)	31.4	33.1	29.7	31.5	41.2	40.5
Margin (Wu et al. 2017)	32.9	34.6	33.4	34.8	44.6	43.8
MS (Wang et al. 2019a)	32.7	34.4	34.1	35.6	45.2	44.3
Ours (Margin)	**33.6**	**35.1**	36.6	37.5	45.4	44.5
Ours (MS)	32.9	34.6	**37.0**	**38.0**	**45.7**	**44.9**

Table 3: Ranking accuracy on three datasets. All of the methods use the setting R^{128} for a fair comparison.

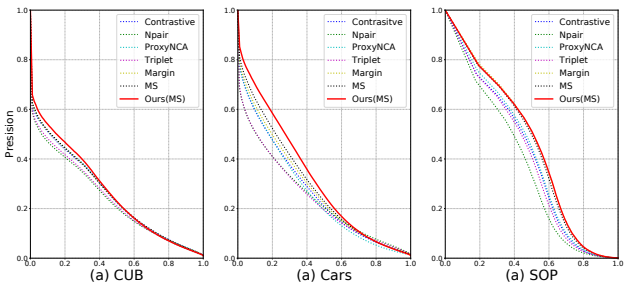


Figure 4: Precision-Recall curves for retrieval. All of the methods use the setting R^{128} for a fair comparison.

performances greatly. Tab. 2 demonstrates that our method outperforms state-of-the-art methods on all datasets. For example, it surpasses current loss functions, *e.g.*, SoftTriple (Qian et al. 2019), RLL-H (Wang et al. 2019b) and Circle (Sun et al. 2020) by 2% - 4% in Recall@1.

When compared with other boost-like methods with the same backbone (R^{128}) and baseline loss (Margin loss), our approach still gets better promotion and generalization. For example, we achieve a higher performance than Divide (Sanakoyeu et al. 2019) by 65.9% $\rightarrow$ 66.5% on CUB, and outperform MIC (Roth et al. 2019) by 82.6% $\rightarrow$ 84.5% on Cars and PADS (Roth et al. 2020) by 76.5% $\rightarrow$ 77.9% on SOP. It is worth noting that, despite our method uses 128-d embeddings, it still gets better results than some state-of-the-art ensemble methods with 512-d embeddings, such as A-BIER (Opitz et al. 2018), ABE (Kim et al. 2018) and RaMBO (Rolínek et al. 2020).

Comparison on Ranking Results

Now we evaluate the performances with Precision-Recall curve, Mean Average Precision (MAP) and R-Precision (RP) as metrics, which are more informative and rational to measure ranking accuracy. These protocols have been mentioned in recent work (Musgrave et al. 2020, Fehervari et al. 2019). Tab. 3 and Fig. 4 show our model improves the ranking performances when compared to other methods, which confirms the effectiveness in learning intra-class variance. There are lots of samples in every class on CUB and Cars datasets, hence their intra-class variances are abundant and the ranking improvements are impressive. However, SOP has a few examples under each class, the capture of intra-class features is not sensitive for retrieval, which leads to little gain with our method.

Qualitative results are illustrated in Fig. 5. Our method

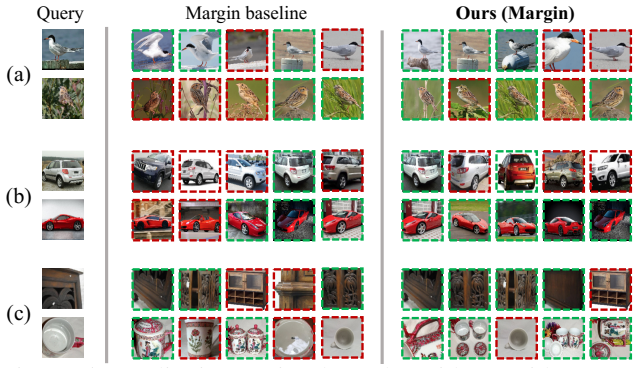


Figure 5: Qualitative retrieval results with or without our method on Margin loss (Wu et al. 2017). (a), (b), (c) is for CUB, Cars, SOP datasets respectively. For each query image (leftmost), top-5 retrieval results are presented with left-to-right ranking according to relative distances. Correct results are highlighted with green, while incorrect red.

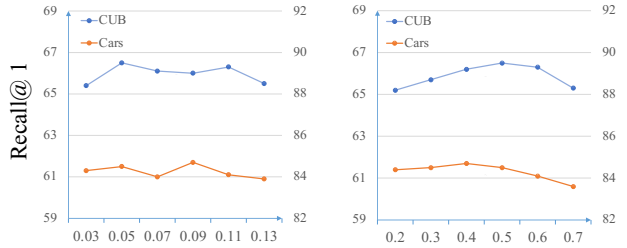


Figure 6: Impact of hyper-parameters. We evaluate two crucial hyper-parameters by R@1 results, ranking margin α (in $\mathcal{L}_{sort}$) and positive boundary β (in $\mathcal{L}_{pos}$).

promotes models to learn more robust embeddings by capturing intra-class variance. Now the new embeddings can help to retrieve images correctly and also keep their relative rankings while the original baseline fails, especially when there are misleading poses (Fig. 5a) or colors (Fig. 5b).

Ablation Study

We provide ablation experiments to verify the effectiveness of our method and evaluate the contribution of different modules. We choose Margin loss (Wu et al. 2017) and train models on CUB and Cars datasets.

Hyper-parameters In Fig. 6, we show the impact of two important hyper-parameters, α, β. When one is variable, the other is fixed as the default setting for controlled experiments. As the boundary of positive pairs, β can't be too large or small otherwise the performances drop heavily. And the performances are stable when α is changed in a proper range. Note that the setting in Section is not best since we did not tune them elaborately according to the test set.

Transform functions Self-supervised learning methods are sensitive to the choice of image augmentations (He et al. 2020; Chen et al. 2020), so we assess the impact of transform functions for our model. As shown in Fig. 7, the performance of our method drops when we remove the specific transform functions or their combinations. Especially the random crop is the most important transform function for the best result, as a common data augment in training networks. Then the color jitter and perspective transform take the second and third place respectively.

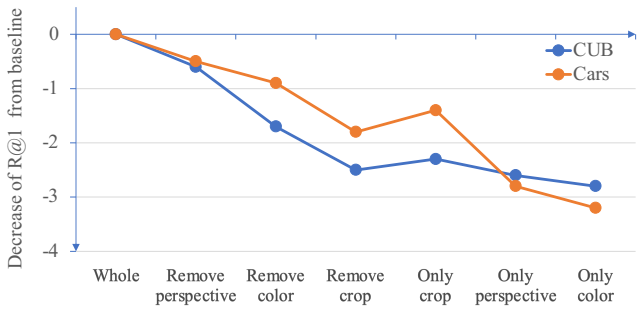


Figure 7: Impact of removing transform. Decrease of R@1 when we remove three transform functions gradually, (random) crop, color (jitter), perspective (transform).

L_{sort}	L_{pos}	g_θ	CUB	Cars
			63.6	79.6
✓			61.9	78.3
	✓		60.7	76.5
	✓	✓	62.5	79.4
✓		✓	64.3	82.3
✓	✓		65.8	84.0
✓	✓	✓	**66.5**	**84.5**

Table 4: Contributions of different modules. The R@1 results of multiple combinations. '✓' means retaining the corresponding modules on our framework otherwise removing.

Framework modules In order to analyze the effectiveness of different modules, *i.e.*, $\mathcal{L}_{sort}$, $\mathcal{L}_{pos}$ and g_θ, we evaluate our framework with different compositions. Tab. 4 shows that these modules are complementary. When only $\mathcal{L}_{sort}$ or $\mathcal{L}_{pos}$ is incorporated into the self-supervised learning procedure, the performances even get worse. By contrast, the combination of the two can help to learn better embeddings, and the best result comes with the incorporation of all modules. We also find that $\mathcal{L}_{sort}$ is most important and L_{pos}, g_θ can further enhance its effect.

Conclusion

This work presents a novel self-supervised ranking auxiliary framework on deep metric learning. We define the standard form of intra-class variance and present relative transform function to measure them. A specific network and objective are developed to preserve the corresponding ranking information in the embedding space, which helps to learn more discriminative embeddings. Experimental results show that our approach significantly improves the baselines on retrieval and ranking tasks, and outperforms state-of-the-art methods on all benchmarks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (grant No.U19A2057, No.61525206, No.61876223), the National Key Research and Development Program (grant No. 2020YFB1406603), the Fundamental Research Funds for the Central Universities (grant No.WK3480000008).

References

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 132–149.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709* .

Chen, W.; Liu, T.-Y.; Lan, Y.; Ma, Z.-M.; and Li, H. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*, 315–323.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, 539–546. IEEE.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4690–4699.

Fehervari, I.; Ravichandran, A.; and Appalaraju, S. 2019. Unbiased evaluation of deep metric learning algorithms. *arXiv preprint arXiv:1911.12528* .

Ge, W. 2018. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 269–285.

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Jacob, P.; Picard, D.; Histace, A.; and Klein, E. 2019. Metric learning with horde: High-order regularizer for deep embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, 6539–6548.

Järvelin, K.; and Kekäläinen, J. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20(4): 422–446.

Kim, W.; Goyal, B.; Chawla, K.; Lee, J.; and Kwon, K. 2018. Attention-based ensemble for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 736–751.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, 554–561.

Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; and Song, L. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 212–220.

Liu, X.; Van De Weijer, J.; and Bagdanov, A. D. 2019. Exploiting unlabeled data in cnns by self-supervised learning to rank. *IEEE transactions on pattern analysis and machine intelligence* 41(8): 1862–1878.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* .

Movshovitz-Attias, Y.; Toshev, A.; Leung, T. K.; Ioffe, S.; and Singh, S. 2017. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, 360–368.

Musgrave, K.; Belongie, S.; and Lim, S.-N. 2020. A metric learning reality check. *arXiv preprint arXiv:2003.08505* .

Oh Song, H.; Jegelka, S.; Rathod, V.; and Murphy, K. 2017. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5382–5390.

Oh Song, H.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4004–4012.

Opitz, M.; Waltner, G.; Possegger, H.; and Bischof, H. 2018. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 8026–8037.

Qian, Q.; Shang, L.; Sun, B.; Hu, J.; Li, H.; and Jin, R. 2019. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE International Conference on Computer Vision*, 6450–6458.

Rolínek, M.; Musil, V.; Paulus, A.; Vlastelica, M.; Michaelis, C.; and Martius, G. 2020. Optimizing Rank-based Metrics with Blackbox Differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7620–7630.

Roth, K.; Brattoli, B.; and Ommer, B. 2019. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 8000–8009.

Roth, K.; Milbich, T.; and Ommer, B. 2020. PADS: Policy-Adapted Sampling for Visual Similarity Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6568–6577.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115(3): 211–252.

Sanakoyeu, A.; Tschernezki, V.; Buchler, U.; and Ommer, B. 2019. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 471–480.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 815–823.

Schütze, H.; Manning, C. D.; and Raghavan, P. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Sohn, K. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, 1857–1865.

Suh, Y.; Han, B.; Kim, W.; and Lee, K. M. 2019. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7251–7259.

Sun, Y.; Cheng, C.; Zhang, Y.; Zhang, C.; Zheng, L.; Wang, Z.; and Wei, Y. 2020. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6398–6407.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, X.; Han, X.; Huang, W.; Dong, D.; and Scott, M. R. 2019a. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5022–5030.

Wang, X.; Hua, Y.; Kodirov, E.; Hu, G.; Garnier, R.; and Robertson, N. M. 2019b. Ranked list loss for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5207–5216.

Wang, X.; Zhang, H.; Huang, W.; and Scott, M. R. 2020. Cross-Batch Memory for Embedding Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6388–6397.

Wu, C.-Y.; Manmatha, R.; Smola, A. J.; and Krahenbuhl, P. 2017. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2840–2848.

Xiao, T.; Li, S.; Wang, B.; Lin, L.; and Wang, X. 2017. Joint detection and identification feature learning for person search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3415–3424.

Yi, D.; Lei, Z.; Liao, S.; and Li, S. Z. 2014. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, 34–39. IEEE.

Zhai, X.; Oliver, A.; Kolesnikov, A.; and Beyer, L. 2019. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, 1476–1485.

Zhang, L.; Zhang, Y.; Gu, X.; Tang, J.; and Tian, Q. 2014. Scalable Similarity Search With Topology Preserving Hashing. *IEEE Transactions on Image Processing* 23(7): 3025–3039.

Zhang, R.; Isola, P.; and Efros, A. A. 2017. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1058–1067.

Zhao, Y.; Jin, Z.; Qi, G.-j.; Lu, H.; and Hua, X.-s. 2018. An adversarial approach to hard triplet generation. In *Proceedings of the European conference on computer vision (ECCV)*, 501–517.

Zheng, W.; Chen, Z.; Lu, J.; and Zhou, J. 2019. Hardness-aware deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 72–81.