MULTI-HIERARCHY PROXY STRUCTURE FOR DEEP METRIC LEARNING

Jian Wang[1], Xinyue Li[1,], Wei Song[1], Zhichao Zhang[2], Weiqi Guo[1,3]*

[1]Shanghai Ocean University, Shanghai, China
[2]Shanghai Jiao Tong University, Shanghai, China
[3]East Sea Oceanographic Engineering Investigation, Design and Research Institute, Shanghai, China

ABSTRACT

Mainstream methods for deep metric learning can be divided into pair-based and proxy-based methods. In recent years, proxy-based methods have attracted wide attention for their low training complexity and fast network convergence. Most proxy-based studies assign only one proxy per class to capture the features of the class, this leads to ignoring the hidden hierarchy and regular aggregation of features within the class. However, these details are meaningful for capturing features of the class. Therefore, we propose a multi-hierarchy proxy (MHP) structure to extract the hierarchical details and regular features hidden in the embedding space. At the same time, we design a layerwise merging similarity operator to reasonably measure the similarity between samples and classes. Our MHP method maintains the low time complexity of the proxy-based method and can be easily integrated into existing proxy-based losses. The effectiveness of our method is evaluated by extensive experiments on three public datasets and compared with state-of-the-art methods. The results show that the proposed MHP method can significantly improve the performance of proxy-based methods, reaching 69.8% on CUB-200-2011 and 87.4% on Cars-196 dataset at Recall@1.

***Index Terms*—** deep metric learning, proxy-based loss, multi-hierarchy proxy, image retrieval

1. INTRODUCTION

In recent years, as one of the most important methods of image retrieval, deep metric learning is developing rapidly and have been widely used in many fields such as industry[1, 2], medical treatment[3, 4], transportation[5, 6], and so on. Deep metric learning aims to train the network to learn a reasonable embedding space, which can effectively measure the similarity between samples. In this embedding space, similar samples are close to each other, while dissimilar samples are separated from each other. To achieve this, researchers first extracted the embeddings of samples from the deep neural network, and associated the embeddings into different forms to design a reasonable loss function to optimize the embedding space. The prevailing research strategy can be divided into two forms: pair-based methods and proxy-based methods.

The study of pair-based methods begins with the Siamese network. Samples of the same class are formed into positive pairs and samples of different classes are formed into negative pairs[7, 8]. These methods encourage positive samples to move closer and negative samples to move away. Triplet is another form of the pair-based method. Triplet methods[9, 10, 11] select a sample as an anchor point, then select the same class and different class samples of the anchor point to form a triplet. These methods encourage positive samples to approach the anchor point and negative samples to move away. Pair-based methods can exploit the rich data-data relationships, but with the increase of the number of sample pairs, high training complexity is a matter of concern.

The proxy-based methods are associating data points with classes. The key of these methods is how to use fewer proxies to capture features in one class and to effectively promote network convergence. Proxy-NCA loss[12] assigns one proxy to each class and associates each sample with proxies of all classes. Therefore, it can promote samples to proxy of the same classes and away from proxies of different classes through Neighborhood Component Analysis (NCA) loss[13]. Proxy-Anchor loss[14] follows the proxy assignment strategy of Proxy-NCA loss. It uses a weighted optimization method to adjust the optimization intensity based on the similarity between sample and proxy. SoftTriple loss[15] improves softmax loss to a proxy-based form by assigning multiple proxies to each class to reflect feature differences within the class. Manifold Proxy loss[16] extends N-pair loss[17] to a form of proxy-based method. Hierarchical proxy-based method[16] extracts a smaller number of cluster centroids to calculate the loss on existing proxies, and captures the association among different classes. Smooth Proxy-Anchor loss[18] adds a confidence module to combat the interference of noisy labels in the dataset.

Since the number of proxies is much smaller than the number of triplets in pair-based method, the training complexity is significantly reduced. But assigning one or very few proxies to each class is insufficient to capture the intra-class features. In addition, we find that in many realistic datasets, there are large feature differences within the class, and these differences are clustered regularly. For example, the shooting environment of bird images can be obviously divided into water, land, and sky. Under different environment birds have flying and stagnation states, and in these two states, juvenile and adult birds show different body shapes. These hidden hierarchies and regular aggregation of features within the class are obvious in most datasets and real-world scenarios in our life. This inspires us to explore the hierarchy of proxies.

In this work, we propose a Multi-Hierarchy Proxy method (MHP). Specifically, we design a special multi-hierarchy proxy structure, which assigns multiple layers of proxies to an embedding space, and proxies at different layers use an independent optimization method to extract the hierarchical information hidden in data. Based on the multi-hierarchy proxy structure, we design a sample-class similarity operator that employs layerwise merging to avoid feature missing caused by the gradient descent from high-layer to low-layer proxies, and generates a more reasonable metric of similarity.

Our contribution can be summarized as follows:

This work was supported by the National Natural Science Foundation of China (NSFC) grant (No.61972240), and the Program for the Capacity Development of Shanghai Local Colleges (No.20050501900).

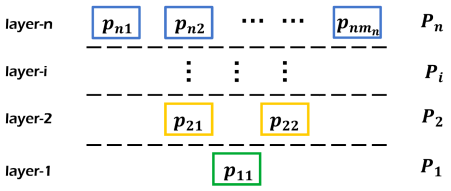

Fig. 1. Multi-Hierarchy proxy structure.

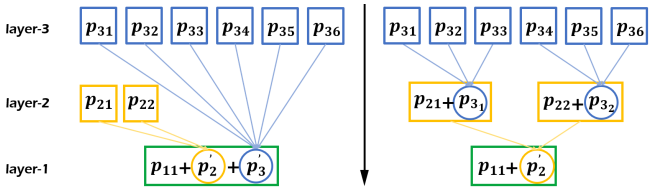

Fig. 2. Similarity operator based on the MHP structure. (a)Direct Merging. (b)Layerwise Merging.

1. We propose a multi-hierarchy proxy structure, which makes up for the deficiency of proxy-based methods in expressing the embedding space, and captures the details and hierarchical structural features of the class.

2. We propose a similarity operator between samples and classes, which can retain the embedding information extracted from the multi-hierarchy proxy structure.

3. We apply our MHP method to the current state-of-the-art proxy-based losses and conduct extensive experiments. The results show that our MHP method achieves new performance on three public datasets.

2. PROPOSED METHOD

2.1. Multi-Hierarchy Proxy Structure

The multi-hierarchy proxy can be described as a pyramid structure. Multiple proxies are assigned to each class to form an n-layer pyramid. Let $P = \{P_1, P_2, \ldots, P_i, \ldots, P_n\}$ represent the set of all proxies for one class, and P_i is the set of proxies at the i-th layer, $p_{i,j} \in P_i$ represents the j-th proxy at the i-th layer. Let m_i be the number of proxies at the i-th layer, and $m_1 < m_2, \ldots, < m_n$, the total number of proxies for this class is $N_p = \sum_{i=1}^{n} m_i$. The overall MHP structure is demonstrated in Fig.1.

In particular, proxies associate within layers and are independent between layers in the process of initialization and optimization, and these processes are performed end-to-end in the network.

The number of proxy layers and the number of proxies in each layer remain consistent in each class. So in the following section, we analyze for one class.

2.2. Similarity Operator

In loss calculation, we expect that the loss can accurately express the difference between the predicted and the actual results through multiple proxies at multiple levels. Therefore, our goal is to calculate the similarity between sample x and class c accurately, in other words, to represent the feature of class c accurately through multiple proxies and multiple layers assigned to one class.

$$S_{x,c} = \sum_{i=1}^{n} \mu^{i-1} S_{x,P_i^c}, \mu \in (0,1) \tag{1}$$

The similarity between sample x and class c is written as (1), μ is a hyperparameter used to control the impact of each proxy layer. P_i^c represents the set of all proxies at layer i of class c, and S_{x,P_i^c} represents the similarity between sample x and all proxies at layer i of class c.

It can be seen that the key to calculate the similarity $S_{x,c}$ between sample x and class c is how to calculate the similarity between sample x and each layer of proxies in class c, such as, the similarity between sample x and all the proxies at layer i of class c is S_{x,P_i^c}.

However, for the layer with more than one proxy, the similarity between sample x and layers of multiple proxies needs to be calculated in a special way, as (2), where $p_{i,j}^c$ represents the j-th proxy in the i-th layer of proxies of class c, and $S_{x,p_{i,j}^c}$ represents the similarity between sample x and the j-th proxy in the i-th layer proxy of class c. And we use the SoftMax function in this dimension as the weight for the similarity between sample x and all proxies in layer i of class c.

$$S_{x,P_i^c} = \sum_{j=1}^{m_i} S_{x,p_{i,j}^c} \cdot softmax(S_{x,p_{i,j}^c}) \tag{2}$$

Suppose we set 3 proxy layers for each class, i.e., $n = 3$. And the number of proxies at each layer is $\{1,2,6\}$, i.e., $m_1 = 1, m_2 = 2$, $m_3 = 6$. For direct merging, as shown in Fig.2(a), the similarity dimension between the sample and the second and third layer proxies is directly reduced to the first layer. At this time, the similarity of these three layers can be calculated. In the final process of accumulation, we added the weighting factor μ to control the proportion of the impact of similarity at each proxy layer.

However, this method has some disadvantages in the case of more layers of proxies or a large number of proxies in one layer. If the similarity between sample x and the high layer proxy is directly reduced to the first layer for summation, it will cause a large amount of feature missing and make the similarity calculation between samples and classes less accurate. Therefore, we merge the similarity dimension from the higher-layer proxy to the lower-layer proxy step by step to reduce feature missing.

As shown in Fig.2(b), there are 6 third-layer proxies and 2 second-layer proxies, so we first transform the similarity of the third-layer proxies from 6 dimensions to 2 dimensions to accommodate the second-layer proxies. Then the similarity with the two proxies of the second layer is summed up in proportion, and then merged to the first layer. This allows all proxies at each layer to be used to calculate the similarity between samples and classes.

In this way, in order to make the similarity dimension of high-layer proxies merge to low-layer proxies step by step, we require that the number of high-layer proxies must be an integer multiple of the number of low-layer proxies, that is, $\forall i,\ \exists m_i \in \mathbb{N}^+,\ satisfy\ N_i = m_i N_{i+1}$. We demonstrate the advantage of layerwise merging in Section 3.4.

2.3. Proxy-based Losses with Multi-Hierarchy

The proposed MHP method can be integrated with the existing proxy-based losses. In the following, we provide a guide for applying the MHP method plug and play to existing proxy-based losses. Through the above analysis and the formulas (1) and (2), our MHP method is shown in the form of optimization items as (3).

$$S_{x,c}^{MHP} = \sum_{i=1}^{n} \mu^{i-1} \sum_{j=1}^{m_i} S_{x,p_{i,j}^c} \cdot softmax(S_{x,p_{i,j}^c}) \quad (3)$$

MHP-Proxy-NCA Loss: Proxy-NCA loss [12] selects the anchor point x, the positive proxy y with the same label as the anchor, and the negative proxy set Z composed of all the remaining proxies to participate in the calculation of the loss. We can rewrite it according to multi-hierarchy proxy method as (4).

$$L_{NCA}^{MHP}(x, y, Z) = -\log\left(\frac{e^{S_{x,y}^{MP}(x,y)}}{\sum_{z \in Z} e^{S_{x,z}^{MP}(x,z)}}\right) \quad (4)$$

MHP-Proxy-Anchor Loss: Proxy-Anchor loss[14] marks the proxy of the class of sample x_p^+ in batch as a positive proxy p^+, and the rest as a negative proxy p^-. The proxy is used as an anchor to form a positive pair and a negative pair with the sample to calculate the loss. According to the multi-hierarchy strategy, the Proxy-Anchor loss can be modified as (5).

$$\begin{aligned} &L_{anchor}^{MHP}(X) = \\ &\frac{1}{|P^+|} \sum_{p \in P^+} \log(1 + \sum_{x \in X_p^+} e^{-\alpha(S_{x,p}^{MP}(x,p) - \delta)}) \\ &+ \frac{1}{|P^-|} \sum_{p \in P^-} \log(1 + \sum_{x \in X_p^-} e^{\alpha(S_{x,p}^{MP}(x,p) + \delta)}) \end{aligned} \quad (5)$$

MHP-SoftTriple Loss: SoftTriple loss[15] associates samples x with the centers y of all classes, uses SoftMax to calculate the similarity between samples and a certain class, and finally computes the cross-entropy loss. We change the optimization of SoftTriple loss from associating samples with single layer proxies to associating with multi-hierarchy proxies, as shown in (6).

$$L_{soft}^{MHP}(x) = -\log \frac{e^{\lambda(S_{x,y}^{MP}(x,y) - \delta)}}{e^{\lambda(S_{x,y}^{MP}(x,y) - \delta)} + \sum_{j \neq y_i} e^{\lambda S_{x,j}^{MP}(x,j)}} \quad (6)$$

3. EXPERIMENTS

3.1. Datasets

We performed experiments on three standard datasets[19, 20, 21]. There is no intersection between the training set and the test set.

The CUB-200-2011 dataset[19] contains 11,788 images of 200 bird species, of which the first 100 classes (5,864 images) were used for training and the remaining 100 classes (5,924 images) for testing.

The Cars196 dataset[20] contains 16,185 images of 196 cars, of which the first 100 classes (8,054 images) were used for training and the remaining 100 classes (8,131 images) for testing.

The SOP dataset[21] contains 120,053 images of 22,634 online products, of which the first 11,318 classes (59,551 images) were used for training and the remaining 11,316 classes (60,502 images) for testing.

3.2. Experiment Setting

Our experiments ran on a single Tesla P100 GPU with 16 GB RAM. We used the PyTorch framework to implement the multi-hierarchy structure. We used batch normalized (BN)-Inception[27] as the backbone feature extraction network and pre-trained on the ImageNet dataset[28]. For all datasets, the input images were first resized to 256×256 and then cropped to 224×224. Each model was trained for 40 epoches and the batch size was set to 150. Input batches were randomly sampled during training.

Number	Method	R@1	R@2	R@4	R@8
CUB-200-2011					
N_p=5±2	$\text{MHP}_{(1,4)}$	68.4	79.0	86.5	91.7
	$\text{MHP}_{(1,2,4)}$	69.4	79.2	86.1	91.6
N_p=10±2	$\text{MHP}_{(1,8)}$	68.1	78.2	86.1	91.7
	$\text{MHP}_{(1,3,6)}$	**69.8**	**79.8**	**87.1**	**92.1**
N_p=15±2	$\text{MHP}_{(1,4,8)}$	68.4	78.6	86.9	91.6
	$\text{MHP}_{(1,2,4,8)}$	69.1	78.9	86.2	**92.1**
Cars-196					
N_p=5±2	$\text{MHP}_{(1,4)}$	87.0	92.1	95.4	97.4
	$\text{MHP}_{(1,2,4)}$	86.8	92.1	95.4	97.3
N_p=10±2	$\text{MHP}_{(1,8)}$	86.6	92.2	95.3	97.4
	$\text{MHP}_{(1,3,6)}$	**87.4**	**92.5**	**95.4**	**97.7**
N_p=15±2	$\text{MHP}_{(1,4,8)}$	87.0	92.2	**95.4**	97.4
	$\text{MHP}_{(1,2,4,8)}$	86.7	92.2	95.3	97.3

Table 1. Impact of the number of proxies and proxy layers

3.3. Parameter of the Multi-Hierarchy Proxy Structure

In the multi-hierarchy proxy structure, increasing an appropriate number of layers can enhance the ability of the proxy to represent the embedding space.

To find the optimal number of proxy layers, we adjusted the proxy layer by fixing the same number of proxies for each class and the parameter μ in (1) to 0.5. The number of proxies in each class was fixed as {5,10,15}(±2), and different layers were designed for comparison. The results are shown in Table.1.

Experiments show that when the number of proxies is N_p=10(±2), the structure {1,3,6} achieves the best performance. With the increase of the number of samples and layers, the performance goes down. Therefore, we set the structure {1,3,6} in the following experiments.

3.4. Ablation Studies

In this section, we carried out ablation studies on two datasets: CUB-200-2011[19] and Cars-196[20], to demonstrate the effectiveness of the multi-hierarchy proxy structure and the similarity operator.

3.4.1. Validation of the Multi-Hierarchy Proxy Structure

To verify the effectiveness of our multi-hierarchy proxy structure, we compared our MHP-SoftTriple loss with the SoftTriple loss[15]. The SoftTriple method assigns 10 proxies per class. For the MHP-SoftTriple loss, we built two 3-layer proxy structures: the structure {1,3,6}(10 proxies per class) and the structure {1,2,4}(7 proxies per class). The results are shown in Table.3.

The results show that the 3-layer proxy structure {1,2,4} with only 7 proxies per class is far better than the original SoftTriple method with 10 proxies per class, and the structure {1,3,6} (10 proxies per class) achieves the best results.

Method		CUB-200-2011				Cars-196				SOP			
	Recall@K(%)	1	2	4	8	1	2	4	8	1	10	100	1000
Pair-based	Lifted Struct^{64}[21]	43.6	56.6	68.6	79.6	53.0	65.7	76.0	84.3	62.5	80.8	91.9	97.4
	N-pair^{64}[17]	51.0	63.3	74.3	83.2	71.1	79.7	86.5	91.6	67.7	83.8	93.0	97.8
	Multi-Similarity^{64}[22]	57.4	69.8	80.0	87.8	77.3	85.3	90.5	94.2	74.1	87.8	94.7	98.2
	ABIER^{512}[23]	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^{512}[24]	60.6	71.5	79.8	87.4	85.2	90.5	94.0	96.1	76.3	88.4	94.8	98.2
	Tuplet Margin^{512}[25]	62.5	73.9	83.0	89.4	86.3	92.3	**95.4**	97.3	78.0	**91.2**	**96.7**	**99.0**
	Multi-Similarity^{512}[22]	65.7	77.0	86.3	91.2	84.1	90.4	94.0	96.5	78.2	90.5	96.0	98.7
	Circle loss^{512}[26]	66.7	77.4	86.2	91.2	83.4	89.8	94.1	96.5	78.3	90.5	96.1	98.6
Proxy-based	Proxy-NCA^{64}[12]	49.2	61.9	67.9	72.4	73.2	82.4	86.4	87.8	73.7	-	-	-
	MHP + Proxy NCA^{64}	53.4	65.1	70.9	76.0	75.9	84.1	88.0	90.1	74.1	87.0	94.3	97.7
	SoftTriple^{512}[15]	65.4	76.4	84.5	90.4	84.5	90.7	94.5	96.9	78.3	90.3	95.9	-
	MHP + SoftTriple^{512}	67.6	77.6	85.6	91.7	85.2	91.3	94.8	97.2	78.9	90.7	96.1	98.5
	Proxy Anchor^{512}[14]	68.4	79.2	86.8	91.6	86.1	91.7	95.0	97.3	79.1	90.8	96.2	98.7
	MHP + Proxy Anchor^{512}	**69.8**	**79.8**	**87.1**	**92.1**	**87.4**	**92.5**	**95.4**	**97.7**	**79.7**	**91.2**	96.4	98.9

Table 2. Recall@K(%) performance on CUB-200-2011, Cars-196 and SOP. Superscript denotes embedding size.

Method	R@1	R@2	R@4	R@8
CUB-200-2011				
SoftTriple$_{(10)}$[15]	65.4	76.4	84.5	90.4
MHP-SoftTriple$_{(1,2,4)}$	66.4	77.3	85.5	91.7
MHP-SoftTriple$_{(1,3,6)}$	**67.6**	**77.6**	**85.6**	**91.7**
Cars-196				
SoftTriple$_{(10)}$[15]	84.5	90.7	94.5	96.9
MHP-SoftTriple$_{(1,2,4)}$	85.1	90.4	94.7	97.0
MHP-SoftTriple$_{(1,3,6)}$	**85.2**	**91.3**	**94.8**	**97.2**

Table 3. Impact of Multi-Hierarchy Structure

Method	R@1	R@2	R@4	R@8
CUB-200-2011				
Direct Merging$_{(1,2,4)}$	66.1	76.4	84.7	90.8
Layerwise Merging$_{(1,2,4)}$	**69.4**	**79.2**	**86.1**	**91.6**
Direct Merging$_{(1,3,6)}$	66.7	76.9	84.6	90.6
Layerwise Merging$_{(1,3,6)}$	**69.8**	**79.8**	**87.1**	**92.1**
Cars-196				
Direct Merging$_{(1,2,4)}$	83.7	90.1	93.8	96.4
Layerwise Merging$_{(1,2,4)}$	**86.8**	**92.1**	**95.4**	**97.3**
Direct Merging$_{(1,3,6)}$	84.1	89.9	93.7	96.6
Layerwise Merging$_{(1,3,6)}$	**87.4**	**92.5**	**95.4**	**97.7**

Table 4. Impact of the layerwise merging and direct merging

3.4.2. Validation of Similarity Operator

To verify the advantages of the similarity operator in the MHP method, we adopted the layerwise merging and direct merging strategies on the MHP method for comparison, respectively. Both methods were tested under the same proxy structures $\{1,2,4\}$ and $\{1,3,6\}$ based on Proxy-Anchor loss[14], as shown in Table.4.

Experiments show that, in our MHP method, regardless of the structure of the proxy, the layerwise merging strategy is obviously superior to the direct descent merging.

3.5. Time Complexity Analysis

In this part, we analyze the time complexity of the classical methods in deep metric learning.

Pair-based	Complexity	Proxy-based	Complexity
Contrastive[8]	O(N^2)	NCA[12]	O(CN)
Triplet[9]	O(N^3)	Anchor[14]	O(CN)
N-pair[17]	O(N^3)	SoftTriple[15]	O(CMN)
Lifted Struct[21]	O(N^3)	MHP (Ours)	O(CMN)

Table 5. Comparison of time complexity

Suppose the number of the sample in the dataset is N. In the pair-based losses, all samples need to calculate the similarity with each other, so the lowest time complexity is $O(N^2)$, while the time complexity of some triplet-based losses[9] is $O(N^3)$.

In the proxy-based losses, we only need to calculate the similarity between samples and classes, and the number of classes C is finite, so the lowest time complexity is $O(CN)$. In particular, for Proxy-NCA[12] and Proxy-Anchor[14], C is a constant. For our MHP method and SoftTriple loss[15] with the multi-proxy assignment, which need to assign M proxies to each class, the time complexity of these two methods is $O(CMN)$, where M is a constant. Details are shown in Table.5.

3.6. Comparison with Other Methods

We compared the most representative methods with our MHP method on three public datasets[19, 20, 21]. The results are summarized in Table.2. Specifically, our MHP method reaches 69.8% on the CUB-200-2011, 87.4% on Cars-196, and 79.7% on SOP at Recall@1.

4. CONCLUSION

In this work, we find that there are many hidden hierarchies and regular aggregation of features within the class in most datasets. Therefore, we propose a multi-hierarchy proxy method (MHP) for deep metric learning. In this method, we set up a multi-hierarchy proxy structure for each class. Furthermore, based on this structure, we design the corresponding sample-class similarity operator, which retains the detail feature of the class as much as possible. With simple modifications, our MHP method can significantly improve the performance of the current proxy-based losses.

5. REFERENCES

[1] Z. B. Xu, X. J. Li, H. Lin, Z. G. Wang, and T. Peng, "Fault diagnosis of rolling bearing based on modified deep metric learning method," *Shock and Vibration*, vol. 2021, 2021.

[2] Kuan Xu, Xudong Li, Hongzhi Jiang, and Huijie Zhao, "Deep metric learning on point sets for 3D industry elements recognition," in *Optical Sensing and Imaging Technologies and Applications*, 2018, vol. 10846, pp. 726–731.

[3] P. S. Yang, Y. P. Zhai, L. Li, H. R. Lv, J. G. Wang, C. Z. Zhu, and R. Jiang, "A deep metric learning approach for histopathological image retrieval," *Methods*, vol. 179, pp. 14–25, 2020.

[4] T. Iesmantas, A. Paulauskaite-Taraseviciene, and K. Sutiene, "Enhancing multi-tissue and multi-scale cell nuclei segmentation with deep metric learning," *Applied Sciences-Basel*, vol. 10, no. 2, 2020.

[5] L. Hu, J. E. Ma, and Y. T. Fang, "Defect recognition of insulators on catenary via multi-oriented detection and deep metric learning," *Proceedings of the 38th Chinese Control Conference (CCC)*, pp. 7522–7527, 2019.

[6] Jinghan Du, Minghua Hu, and Weining Zhang, "Decision support for aircraft taxi time based on deep metric learning," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.

[7] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, vol. 2, pp. 1735–1742.

[8] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 1, pp. 539–546 vol. 1.

[9] Elad Hoffer and Nir Ailon, "Deep metric learning using triplet network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9370, no. 1271, pp. 84–92, 2015.

[10] Ben Harwood, Vijay Kumar B.G., Gustavo Carneiro, Ian Reid, and Tom Drummond, "Smart mining for deep metric learning," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2840–2848.

[11] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai, "Triplet-center loss for multi-view 3d object retrieval," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1945–1954.

[12] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh, "No fuss distance metric learning using proxies," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 360–368.

[13] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov, "Neighbourhood components analysis," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, 2005, pp. 513–520.

[14] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak, "Proxy anchor loss for deep metric learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3235–3244.

[15] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Tacoma Tacoma, Hao Li, and Rong Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6449–6457.

[16] Nicolas Aziere and Sinisa Todorovic, "Ensemble deep manifold similarity learning using hard proxies," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7291–7299.

[17] Kihyuk Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 1857–1865.

[18] Carlos Roig, David Varas, Issey Masuda, Juan Carlos Riveiro, and Elisenda Bou-Balust, "Smooth Proxy-Anchor Loss for Noisy Metric Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei, "3d object representations for fine-grained categorization," in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 554–561.

[21] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese, "Deep metric learning via lifted structured feature embedding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4004–4012.

[22] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5017–5025.

[23] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof, "Deep metric learning with bier: Boosting independent embeddings robustly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 276–290, 2020.

[24] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon, "Attention-based ensemble for deep metric learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[25] Baosheng Yu and Dacheng Tao, "Deep metric learning with tuplet margin loss," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6489–6498.

[26] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei, "Circle loss: A unified perspective of pair similarity optimization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6397–6406.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine Learning*, vol. 37, pp. 448–456, 2015.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.