The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

Neighborhood-Adaptive Structure Augmented Metric Learning

Pandeng Li[1]*, Yan Li[2]*, Hongtao Xie[1]†, Lei Zhang[2]

[1]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[2]Kuaishou Technology, Beijing, China
lpd@mail.ustc.edu.cn, liyan@kuaishou.com, htxie@ustc.edu.cn, zhanglei06@kuaishou.com

Abstract

Most metric learning techniques typically focus on sample embedding learning, while implicitly assume a homogeneous local neighborhood around each sample, based on the metrics used in training (e.g., hypersphere for Euclidean distance or unit hyperspherical crown for cosine distance). As real-world data often lies on a low-dimensional manifold curved in a high-dimensional space, it is unlikely that everywhere of the manifold shares the same local structures in the input space. Besides, considering the non-linearity of neural networks, the local structure in the output embedding space may not be as homogeneous as assumed. Therefore, representing each sample simply with its embedding while ignoring its individual neighborhood structure would have limitations in Embedding-Based Retrieval (EBR). By exploiting the heterogeneity of local structures in the embedding space, we propose a Neighborhood-Adaptive Structure Augmented metric learning framework (NASA), where the neighborhood structure is realized as a structure embedding, and learned along with the sample embedding in a self-supervised manner. In this way, without any modifications, most indexing techniques can be used to support large-scale EBR with NASA embeddings. Experiments on six standard benchmarks with two kinds of embeddings, i.e., binary embeddings and real-valued embeddings, show that our method significantly improves and outperforms the state-of-the-art methods.

Introduction

With the recent success of deep learning, deep metric learning (DML) methods have demonstrated strong ability in various tasks (Ge et al. 2021; Liu et al. 2021; Peng et al. 2021; Lin et al. 2021; Wang et al. 2020b), such as semantic search (Huang et al. 2020; Li et al. 2021b; Min et al. 2020a,b) and face recognition (Li et al. 2021a). Most existing approaches (Roth, Brattoli, and Ommer 2019; Wu et al. 2017) take as input a sample (*e.g.,* an image or a document), use a trained neural network as an encoder and represent this sample with the output embedding. Currently, almost all of these methods focus on the encoder training, and various objectives are formulated in terms of pairwise similarities within a mini-batch (Wang et al. 2019) or similarities between samples and classification vectors (Deng et al. 2019). Sampling strategies (Wu et al. 2017), memory bank (Wang et al. 2020a) and examples generation (Ko and Gu 2020) are proposed to mine informative samples in the training process, and boost-like policies (Roth, Brattoli, and Ommer 2019) are used to improve the encoder. Generally, existing methods concentrate on how to learn a good embedding for each sample and represent each sample with its embedding.

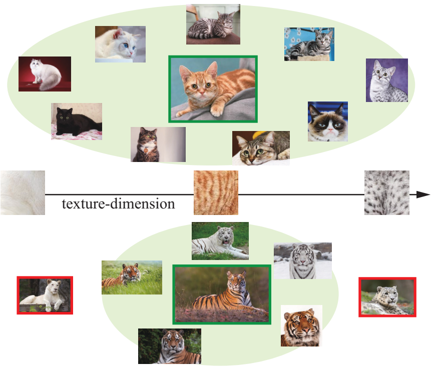


Figure 1: Qualitative illustration of adaptive neighborhood: neighbors of a cat can distribute in a wide range on the "*texture*" dimension, since cats can have very different texture patterns, while neighbors of a tiger should distribute in a narrow range, since all tigers have similar texture patterns.

The similarity of two samples are then measured by the proximity between their embeddings, using a distance metric predefined in encoder training, such as Euclidean distance (Wu et al. 2017), cosine similarity (Deng et al. 2019) and Hamming distance (Yang et al. 2018). Therefore, DML methods implicitly make an assumption about the manifold structure around each sample: *the local neighborhood of each sample is homogeneous and shares the same structures*. However, this assumption is relatively too ideal. As shown in Fig. 1, given an image of a cat, in the embedding space, its neighbors can distribute in a wide range on the "texture" dimension, since cats can have very different texture patterns. By contrast, for a tiger, its neighbors should distribute in a narrow range, since all kinds of tigers have similar textures. Therefore, on the "texture" dimension, the cat

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

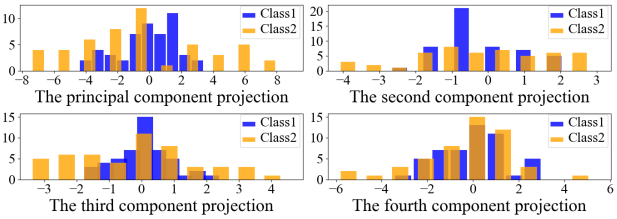


Figure 2: The Principal Component Analysis (PCA) (Abdi and Williams 2010) on CUB-200-2011 using Triplet loss.

and tiger should have different shapes. This observation also holds on practical datasets. We randomly select two classes from the test set of CUB-200-2011 (Wah et al. 2011), and project the DML embeddings onto the first-4 PCA components. Fig. 2 shows that different classes have quite different distributions on the same dimension, which is consistent with Fig. 1. As a result, it is desirable to have adaptive neighborhood structures for different samples.

In this work, we propose a Neighborhood-Adaptive Structure Augmented metric learning framework (NASA) to solve the above issues, where a sample is now represented by its *sample embedding* and local neighborhood *structure embedding*. The overall framework is illustrated in Fig. 3. As downstream tasks of metric learning can usually be cast to Embedding-Based Retrieval (EBR) (Huang et al. 2020), indexing techniques (Jégou, Douze, and Schmid 2011; Malkov and Yashunin 2020) are usually required. Since most indexing techniques only work with the sample embeddings, we formulate the learning procedure of NASA as two sub-tasks: (1) *sample embeddings* are optimized to provide good separability between samples, which work with indexing to efficiently filter out irrelevant results; and (2) *structure embeddings* are learned to capture adaptive local neighborhoods which provide fine-grained result lists, with respect to the retrieval and re-ranking stage of large-scale EBR (Huang et al. 2020). In this way, indexing techniques can be used to facilitate large-scale EBR with NASA without any modifications and additional memory cost.

Our main contributions are summarized as follows: 1) we demonstrate the importance of the local manifold structure of each sample in metric learning; 2) a novel framework, NASA, which captures the adaptive neighborhood structure of each sample, is proposed to learn a globally well-separated and locally discriminative embedding space; 3) results on six benchmark datasets with two kinds of widely-adopted embeddings (real-valued embedding and binary embedding) demonstrate the effectiveness of NASA. To our best knowledge, our method is the first work that tries to exploit the heterogeneity of local neighborhood structures in deep metric learning.

Related Work

Metric Learning. The objective of metric learning is to build new spaces of representations such that similar samples are pushed towards together and different samples are repelled against. Most deep metric learning works focus on the design of metric loss functions based on sample pairs. The earliest is Contrastive loss (Chopra, Hadsell, and LeCun 2005), which minimizes the distances of positive pairs and maximizes the distances of negative pairs. Then pair-based methods are expanded by triplet (Schroff, Kalenichenko, and Philbin 2015), n-pairs (Sohn 2016), and so on. The corresponding hard examples sampling (Wu et al. 2017) and weighting (Wang et al. 2019; Sun et al. 2020) strategies are also helpful to the pair-based losses, but the auxiliary strategies bring about the runtime cost and training instability (Kim et al. 2020). To overcome the shortcoming of pair-based methods, many alternative methods are proposed. One of direction is called proxy-based methods (Movshovitz-Attias et al. 2017; Teh, DeVries, and Taylor 2020). They compare samples to class-related and learnable proxies, but the parameters are tremendous when there are too many categories (Qian et al. 2019; Kim et al. 2020). Some works are not limited to inter-class discrimination, but mining intra-class (Fu et al. 2021) or multi-level relations (Kim and Park 2021) to regularize embedded networks. Recently, probabilistic embeddings have been proposed to measure the uncertainty in the embedding space. HIE (Oh et al. 2019) represents a sample by a Gaussian distribution with mean specifies the position and covariance reflects the uncertainty. In (Shi and Jain 2019), PFE uses a pretrained sample embedding as the mean and only learns the uncertainty (variance). By learning feature (mean) and uncertainty (variance) simultaneously, DUL (Chang et al. 2020) proposes to fully exploit the effect of uncertainty in feature learning, and achieves superior performance in face recognition.

Hashing Learning. Deep unsupervised hashing can be regarded as a special deep metric learning technology, which aims to learn discriminative binary embeddings without labels. DeepBit (Lin et al. 2016) is the first deep unsupervised method to learn hash functions, which mainly considers the binary code consistency between the original images and the rotated images. Generative models (Song et al. 2018; Dai et al. 2017) apply generative adversarial network (Goodfellow et al. 2014) or auto-encoder structure to learn binary codes through the minimum decoding error principle. However, these above methods do not explore semantic similarity well. Subsequently, some methods (Yang et al. 2018; Su et al. 2018) utilize the semantic information in deep features and construct similarity graphs to learn local structures between data. The credibility of the similarity graph structure determines the efficiency of model training. Recently, TBH (Shen et al. 2020) considers the advantages of the generative models and similar learning methods and uses the twin bottleneck structure to obtain the optimal performance.

Method

Given a dataset consist of n images $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ and its corresponding labels $\mathcal{Y} = \{y_i\}_{i=1}^n \in \{1, \cdots, C\}$ (if provided), the set of neighbors of $\boldsymbol{x}_i$ is denoted as $\mathrm{N}(\boldsymbol{x}_i)$. Deep metric learning (DML) aims to learn a feature encoder $f : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^D$ to map $\boldsymbol{x}_i$ into a D-dimensional embedding space $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$, where the distance between $\boldsymbol{x}_i, \boldsymbol{x}_j$ can be effectively measured by their embeddings

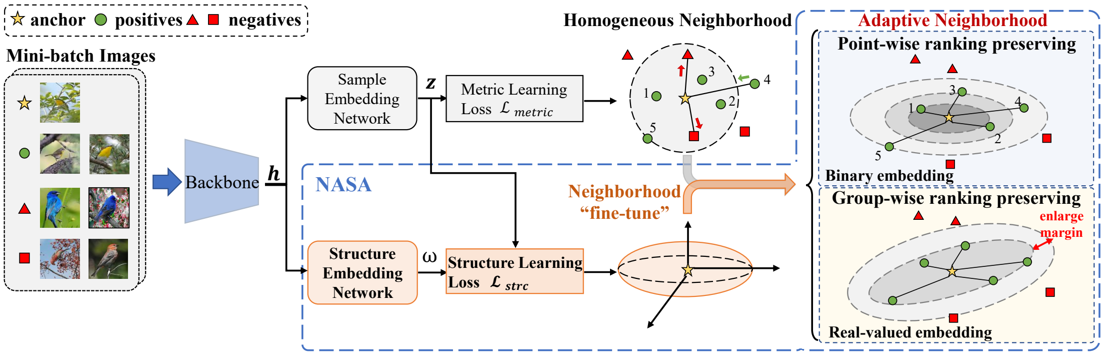


Figure 3: The pipeline of our proposed NASA. A backbone network and two head networks are used to learn both the sample embedding and structure embedding. $\mathcal{L}_{metric}$ is built using existing DML techniques in the embedding space with a homogeneous neighborhood, where positives are pushed together and negatives are repelled. $\mathcal{L}_{strc}$ is solved in the adaptive local neighborhood, which is derived from the structure embedding, to maintain the local data topology.

$\boldsymbol{z}_i, \boldsymbol{z}_j$: $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq d(\boldsymbol{z}_i, \boldsymbol{z}_j)$. $d(\cdot, \cdot)$ is a distance metric such as Hamming distance (Su et al. 2018), Euclidean distance (Radenović, Tolias, and Chum 2019) and cosine similarity (Deng et al. 2019). With a predefined metric, the local region around $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$ is defined as:

$$\mathrm{N}(\boldsymbol{z}_i) \triangleq \{\boldsymbol{z} : d(\boldsymbol{z}, \boldsymbol{z}_i) \leq \tau\}, \qquad (1)$$

where τ is a threshold. By mapping $\mathrm{N}(\boldsymbol{x}_i)$ into $\mathrm{N}(\boldsymbol{z}_i)$ and $\mathcal{X} - \mathrm{N}(\boldsymbol{x}_i)$ out of $\mathrm{N}(\boldsymbol{z}_i)$, DML learns an embedding space with small intra-class distance and large inter-class distance.

Adaptive Neighborhood

Neighborhood structure (1) in the embedding space is implicitly determined by the distance metric used in training. For most widely used metrics, the resulting neighborhood is homogeneous, *i.e.,* deviations on different dimensions contribute equally to the distance, and consistent, *i.e.,* same geometry structure for all samples. However, as shown in (Zhang et al. 2013), if a feature encoder can map $\boldsymbol{x}_i$ and $\mathrm{N}(\boldsymbol{x}_i)$ closer on some dimensions, these dimensions should contribute more in distance calculation, resulting in a heterogeneous and adaptive neighborhood. As a result, given $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$, its local neighborhood structure can be extracted from the posterior distribution of $\{\boldsymbol{z} = f(\boldsymbol{x}) : \boldsymbol{x} \in \mathrm{N}(\boldsymbol{x}_i)\}$. We denote this distribution as $p(\boldsymbol{z}|\boldsymbol{z}_i)$, and parameterize it as a D-dimensional Gaussian distribution with mean $\boldsymbol{z}_i$ and diagonal covariance $\Sigma_i = \text{diag}(\sigma_i^{(1)}, \cdots, \sigma_i^{(D)})$ (Oh et al. 2019; Shi and Jain 2019; Chang et al. 2020):

$$p(\boldsymbol{z}|\boldsymbol{z}_i) = \frac{\exp\left\{-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_i)^T \Sigma_i^{-1} (\boldsymbol{z} - \boldsymbol{z}_i)\right\}}{(2\pi)^{D/2} |\Sigma_i|^{1/2}}. \qquad (2)$$

Eq. (2) can be related to a distance metric as follows (Sebe, Lew, and Huijsmans 2000):

$$d(\boldsymbol{z}, \boldsymbol{z}_i) = \sum_{d=1}^{D} \left(\boldsymbol{z}^{(d)} - \boldsymbol{z}_i^{(d)}\right)^2 / \sigma_i^{(d)}. \qquad (3)$$

Eq (3) realizes an adaptive local neighborhood as deviations on different dimensions are weighted by $\sigma_i^{(d)}$ and contribute differently. As a sample is a neighbor of itself, Eq (2) can be interpreted as a sample $\boldsymbol{x}_i$ is now mapped to a probabilistic distribution in the embedding space (Oh et al. 2019; Shi and Jain 2019), with $\boldsymbol{\mu}_i = \boldsymbol{z}_i$ specifies the location and $\boldsymbol{\omega}_i = \{1/\sigma_i^{(1)}, ..., 1/\sigma_i^{(D)}\}$ reflects the uncertainty. In this work, we call $\boldsymbol{\mu}_i$ the sample embedding and $\boldsymbol{\omega}_i$ the structure embedding, and propose to learn these two kinds of embeddings by exploiting the adaptive neighborhood structures.

Sample Embedding Learning

For two samples $\boldsymbol{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$, $\boldsymbol{z}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, their distance can be extended as:

$$d(\boldsymbol{z}_1, \boldsymbol{z}_2) = \iint\limits_{\boldsymbol{z}'_1 \sim \mathcal{N}_1, \boldsymbol{z}'_2 \sim \mathcal{N}_2} d(\boldsymbol{z}'_1, \boldsymbol{z}'_2) p(\boldsymbol{z}'_1|\boldsymbol{\mu}_1) p(\boldsymbol{z}'_2|\boldsymbol{\mu}_2) d\boldsymbol{z}'_1 d\boldsymbol{z}'_2. \qquad (4)$$

(4) can be approximated via Monte-Carlo sampling, however, this would bring much computational cost. Instead, by using the Euclidean distance as a metric, we can reach a reasonable estimation of distance for any two samples, which supports embedding learning effectively. The expectation of the squared Euclidean distance can be estimated as:

$$\mathrm{E}\left[d^2(\boldsymbol{z}_1, \boldsymbol{z}_2)\right] = \iint\limits_{\boldsymbol{z}'_1, \boldsymbol{z}'_2} p(\boldsymbol{z}'_1) p(\boldsymbol{z}'_2) (\boldsymbol{z}'_1 - \boldsymbol{z}'_2)^T (\boldsymbol{z}'_1 - \boldsymbol{z}'_2) d\boldsymbol{z}'_1 d\boldsymbol{z}'_2. \qquad (5)$$

By factorizing $(\boldsymbol{z}'_1 - \boldsymbol{z}'_2)$ as $(\boldsymbol{z}'_1 - \boldsymbol{\mu}_1 + \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 + \boldsymbol{\mu}_2 - \boldsymbol{z}'_2)$ and expanding the inner product, (5) can be expressed as:

$$\mathrm{E}\left[d^2(\boldsymbol{z}_1, \boldsymbol{z}_2)\right] = \text{tr}(\Sigma_1) + \text{tr}(\Sigma_2) + \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2. \qquad (6)$$

Since $\mathrm{E}[d^2] \geq \mathrm{E}^2[d]$, (6) gives the upper bounds of the expectation of distance between two (probabilistic) embeddings $\boldsymbol{z}_1, \boldsymbol{z}_2$. One can directly use (6) to guide the training process, however this would increase the training cost substantially, since more neighbors are needed in a mini-batch to estimate Σ_i accurately. Instead, we use $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2$, *i.e.,* distance between sample embeddings, as the metric, which still ensures a well-separated embedding space as demonstrated in our experiments. Another advantage of this approximation is that, most DML techniques can now be plied to learn the sample embeddings. The pipeline of s ple embedding learning is illustrated in Fig 3. A sample bedding network is optimized using an existing DML t nique. The learned sample embedding has a homogene neighborhood, which is then locally “fine-tuned” by the lowing structure embedding.

Structure Embedding Learning

For a sample $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$, its structure embedding $\boldsymbol{\omega}$ learned from the manifold structure of its local neighborhood. As a manifold can be entirely characterized by the relative proximities between its subregions (Lee and Verleysen 2007), we use the ranking of neighbors to guide structure embedding learning. Specially, for $\boldsymbol{z}_i$ and its neighbors, $\boldsymbol{z} = f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathrm{N}(\boldsymbol{x}_i)$, by using (3) as adaptive distance, the embedding space is locally stretched or shrunk such that the rankings among neighbors are preserved in this adaptive local neighborhood.

Since it is not able to get the ranking information from current human-labeled signals, (*e.g.,* class label), we start with the help of self-supervised learning strategy. Based on whether we can infer confident ranking information, two kinds of self-supervised learning tasks are proposed.

Point-wise Ranking Learning. Many unsupervised hashing methods (Shen et al. 2020; Yang et al. 2019; Su et al. 2018) assume that the pairwise similarity between features $\boldsymbol{h}$ extracted by a pre-trained network can be directly used to guide the learning of binary embeddings $\boldsymbol{z} \in \{-1, 1\}^D$. Generally, pair samples with high similarity are regarded as high-confidence supervisory signals. In this case, we consider that the top-K similarity ranking information is beneficial to neighborhood structure learning.

In a mini-batch, for an anchor with binary embedding $\boldsymbol{z}_i$, we calculate its cosine similarity with other samples using the *pre-training features*, and get the top-K neighbors $\{\boldsymbol{n}_1, ..., \boldsymbol{n}_K\}$ according to the similarity in a descending order. With structure embedding $\boldsymbol{\omega}_i$, the query-sensitive Hamming distance (Zhang et al. 2013) between $\boldsymbol{z}_i$ and $\boldsymbol{n}_k$ is:

$$d_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k) = \sum_{d=1}^{D} \boldsymbol{\omega}_i^{(d)} \cdot \left(\boldsymbol{z}_i^{(d)} \oplus \boldsymbol{n}_k^{(d)}\right) = \boldsymbol{\omega}_i^T \left(\boldsymbol{z}_i \oplus \boldsymbol{n}_k\right). \tag{7}$$

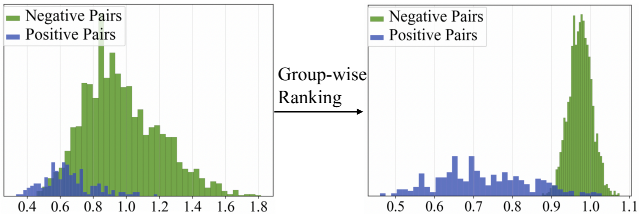
Structure embedding $\boldsymbol{\omega}_i$ is learned to make: $\forall\, p, q \in [1, K]$,

$$\text{if } p < q, \text{ then } d_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_p) < d_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_q).$$

Noticing that deep hashing techniques (Li et al. 2019; Yang et al. 2018) use the inner product of ℓ_2 normalized binary embedding as similarity metric in training, to make the order of magnitude of ranking preserving loss consistent with that of sample embedding learning loss, we map the adaptive distance (7) to adaptive similarity:

$$s_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k) = \frac{\sum_{d=1}^{D} \boldsymbol{\omega}_i^{(d)} - 2 d_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k)}{D} = \boldsymbol{\omega}_i^T \left(\tilde{\boldsymbol{z}}_i^T \tilde{\boldsymbol{n}}_k\right), \tag{8}$$

where $\tilde{\boldsymbol{z}}_i$ and $\tilde{\boldsymbol{n}}_k$ are the ℓ_2 normalized binary embeddings. Since $\sum_d \boldsymbol{\omega}_i^{(d)}$ is consistent for $\boldsymbol{z}_i$, preserving the order of $s_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k)$ is equivalent to preserving the order of $d_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k)$, and the point-wise ranking preserving task is:



(a) The initial distance (b) The adaptive distance

Figure 4: Visualization of distance distribution w/o and w/ group-wise ranking learning (CUB (Wah et al. 2011), Triplet loss, $512d$). The Triplet model has a large distribution overlap, whereas our model has a large group margin.

$$\mathcal{L}_{strc} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K-1} [s_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_{k+1}) - s_{strc}(\boldsymbol{z}_i, \boldsymbol{n}_k) + \beta]_+, \tag{9}$$

where N is the batch size, β is the margin and $[\cdot]_+$ is the hinge function.

Group-wise Ranking Learning. In most cases, we cannot infer confident point-wise ranking information as above, since images usually have large intra-class variances and subtle inter-class distinctions (Wah et al. 2011). As shown Fig. 4-a, for one thing, the initial distances of positive and negative pairs are too close and chaotic, thus the top-K point-wise ranking is confusing and cannot be directly used as supervised signals. For another, *most negatives have larger distances than most positives*. Therefore, for an anchor, we can have a confident “group ranking” between its *positive group* and *negative group*. By exploring local neighborhood structure, we propose a group-wise ranking strategy to separate the two group distributions, which ensures a large margin as well as a small overlap between positive and negative samples, and improves feature discrimination and generalization. The pairwise adaptive distance is defined as:

$$\begin{aligned} d_{strc}(\boldsymbol{z}_i, \boldsymbol{z}_j) &= \sqrt{\textstyle\sum_{d=1}^{D} \boldsymbol{\omega}_i^{(d)} \cdot (\boldsymbol{z}_i^{(d)} - \boldsymbol{z}_j^{(d)})^2} \\ &= \sqrt{\boldsymbol{\omega}_i^T \left(\boldsymbol{r}_{ij} \odot \boldsymbol{r}_{ij}\right)}, \end{aligned} \tag{10}$$

where $\boldsymbol{r}_{ij} = \boldsymbol{z}_i - \boldsymbol{z}_j$ and $\odot$ is Hadamard product. For $\boldsymbol{z}_i$, to separate its two group distributions with a large margin, we first set two distant centers for its positive and negative groups. Then, by squeezing each group towards its own center, the overlap is minimized as well. Since almost all samples in a batch are negatives of $\boldsymbol{z}_i$, the center of negative group, η_1, is set to mean of distances between $\boldsymbol{z}_i$ and other samples. Noticing that the overall distance distribution is bell-shaped (Fig. 4-a), the center of positive group is set to a relatively small value: $\eta_2 = \max(0.2\eta_1, \eta_1 - t \cdot \theta)$, θ is the standard deviation of the overall distance distribution and t controls the margin. The group-wise ranking learning task is formulated as:

$$\mathcal{L}_{strc} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq i} c_{ij}|d_{strc}(\boldsymbol{z}_i, \boldsymbol{z}_j) - \eta_1| + (1 - c_{ij})|d_{strc}(\boldsymbol{z}_i, \boldsymbol{z}_j) - \eta_2|, \tag{11}$$

c_{ij} is the soft assignment to different centers:

$$c_{ij} = \frac{\exp\left(-\alpha|d_{strc}(\boldsymbol{z}_i, \boldsymbol{z}_j) - \eta_1|\right)}{\sum_{m=1}^{2}\exp\left(-\alpha|d_{strc}(\boldsymbol{z}_i, \boldsymbol{z}_j) - \eta_m|\right)}, \tag{12}$$

where α is the scale factor. Each anchor can have its own η_1, η_2, but this may cause the learning process unstable due to the limit of batch-size. In our experiments, we find that using the same η_1, η_2 for all anchors in a mini-batch works well, therefore, η_1 and θ are set to the mean and standard deviation of all pairwise distances in a mini-batch.

Overall Learning

The overall training objective of NASA is as follows:

$$\mathcal{L} = \mathcal{L}_{metric} + \lambda\mathcal{L}_{strc}, \tag{13}$$

where λ relatively weights the losses. $\mathcal{L}_{metric}$ is an existing DML loss, and for binary embedding, $\mathcal{L}_{strc}$ is Eq (9), while for real-valued embedding, $\mathcal{L}_{strc}$ is Eq (11).

Discussion

NASA with indexing. NASA can be used to support large-scale EBR with indexing techniques straightforwardly. For a large-scale dataset, only the sample embeddings are extracted and organized using indexing (Jégou, Douze, and Schmid 2011), while the structure embeddings are skipped, adding no additional computational and memory costs. For a query, its sample embedding is processed by indexing to efficiently filter out irrelevant results, then if required, its structure embedding is employed to re-rank the left candidates based on (10) or (7). That's why only the structure embedding of the anchor is employed in training, as it ensures the consistency of distance metric in testing and training. Since (10) and (7) can be efficiently vectorized, and there are not many candidates, the time cost of re-ranking is negligible.

Asymmetric distance. The adaptive distance, (10) or (7), is asymmetric, *i.e.,* $d_{strc}(\boldsymbol{x}, \boldsymbol{y}) \neq d_{strc}(\boldsymbol{y}, \boldsymbol{x})$ which has its own advantage. Take the central tiger image and the rightmost cat image in Fig. 1 as a *query-recall* pair. If *query*="cat", the adaptive distance on the "texture" dimension should be small, since cats can have similar texture as a tiger. However, if *query*="tiger", the adaptive distance should be relatively large, since the cat has a different texture pattern, from which we can be sure that it is not a tiger. This asymmetry also makes sense in semantic information retrieval (Huang et al. 2020). A document "landmarks in Paris" is a good recall for query "landmarks in France", while "landmarks in France" is not a good recall for "landmarks in Paris". In the future, we would like to investigate whether the essence of adaptive neighborhood can be applied in semantic retrieval.

We clarify some key differences between NASA and several representative probabilistic embeddings (Oh et al. 2019; Shi and Jain 2019; Chang et al. 2020). To measure the similarity of two samples, HIE (Oh et al. 2019) samples $K = 8$ candidates from each distribution, and uses the average of the 64 candidate-pairs' similarities as the estimation, both in training and testing. PFE (Shi and Jain 2019) uses mutual likelihood score as the metric, which takes both the mean and variance into account. Both the above two methods suffer more computational complexity, and cannot directly work with indexing techniques. In DUL (Chang et al. 2020), the variance (Σ) in DUL measures the uncertainty of sample embedding, which should be accurately estimated, thus a KL-divergence regularization is added to avoid overfitting, while in NASA, it stems from the adaptive local neighborhood structure, and the accurate value is not strictly required, making the learning process simple.

MDR (Kim and Park 2021) proposes to regularize distances into multiple levels, which is similar to group separation in group-wise ranking learning. However, the mathematical principles behind MDR and NASA are quite different. MDR still assumes a homogeneous local neighborhood, and its distance separation comes for better generalization. In NASA, group separation is performed on adaptive distance, which is derived from a self-supervised learning task to realizes an adaptive neighborhood.

Experiments

We carry out extensive experiments to evaluate our model in both retrieval and ranking tasks. According to "Reality Check" (Shen et al. 2020; Musgrave, Belongie, and Lim 2020), we employ Mean Average Precision (MAP) as performance metrics for binary embeddings and use Recall@K (R@K), MAP and P@N scores for real-valued embeddings.

Datasets and Settings

Three datasets for binary embeddings. CIFAR-10 (Krizhevsky, Hinton et al. 2009) consists of 60,000 images in 10 categories, which is randomly divided into training and testing sets, with 5,000 and 1,000 images. NUS-WIDE (Chua et al. 2009) is a large-scale dataset with 269,648 web images in 81 concepts. We select 21 largest concepts with 186,577 images, split 10,500 samples for training and 2,100 samples for testing. FLICKR25K (Huiskes and Lew 2008) has 25,000 images collected from Flickr in 24 concepts. We randomly select 2,000 images for testing and 5,000 images for training.

Setting for binary embeddings. We organize experiments on four representative hashing methods (SGH (Dai et al. 2017), SSDH (Yang et al. 2018), GreedyHash (Su et al. 2018), TBH (Shen et al. 2020)). To be consistent with the four hashing methods, AlexNet (Krizhevsky, Sutskever, and Hinton 2012) or VGG-16 (Simonyan and Zisserman 2015) is adopted as our backbone network. We employ SGD algorithm with 0.9 momentum and fix the batch size as 128. The hyper-parameters setting is: $k = 8$, $\beta = 0.03$ and $\lambda = 1$.

Three datasets for real-valued embeddings. CUB-200-2011 (CUB) (Wah et al. 2011) has 11,788 images of 200 bird species. We use the first 100 species (5,864 images) for training and the rest 100 species (5,924 images) for test-

Method	Backbone	CIFAR-10			NUS-WIDE			FLICKR25K		
		32 bits	64 bits	128 bits	32 bits	64 bits	128bits	32 bits	64 bits	128 bits
SGH (Dai et al. 2017)	AlexNet	43.7	44.3	45.6	67.8	69.2	70.4	67.5	68.2	69.4
NASA [B] + SGH		45.8	46.8	48.1	68.7	70.5	71.1	69.3	70.0	70.7
NASA + SGH		**46.1**	**47.3**	**49.8**	**69.8**	**71.9**	**72.2**	**70.1**	**70.5**	**71.5**
SSDH (Yang et al. 2018)	VGG-16	27.4	29.3	30.1	63.4	64.1	65.2	75.6	75.7	76.0
NASA [B] + SSDH		30.2	33.9	37.7	64.6	68.3	69.1	75.9	76.6	78.8
NASA + SSDH		**32.4**	**35.3**	**38.2**	**65.8**	**70.2**	**70.5**	**77.1**	**77.5**	**79.2**
GreedyHash (Su et al. 2018)	AlexNet	36.1	40.2	43.1	73.1	75.0	76.5	66.7	70.3	70.8
NASA [B] + GreedyHash		39.4	43.6	46.0	76.5	78.4	79.6	73.0	74.2	74.8
NASA + GreedyHash		**40.8**	**44.2**	**46.3**	**77.3**	**78.7**	**79.7**	**73.5**	**74.3**	**74.8**
TBH (Shen et al. 2020)	AlexNet	51.4	53.2	54.2	72.5	73.5	73.7	70.1	71.5	71.1
NASA [B] + TBH		51.9	53.7	55.5	72.9	75.8	76.1	70.8	72.1	72.5
NASA + TBH		**52.2**	**54.3**	**55.8**	**74.0**	**76.3**	**76.6**	**71.5**	**73.1**	**73.6**

Table 1: The MAP scores for various bits on the three datasets. NASA [B] means that only binary sample embedding is used.

Method	CUB				Cars				SOP			
	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@10^2	R@10^3
Triplet^{128}	55.5	67.1	77.5	85.4	75.8	84.3	90.1	93.4	72.4	86.5	95.0	98.1
MDR^{128} + Triplet	65.1	76.4	84.2	90.7	84.7	90.5	94.0	95.9	77.1	89.5	95.6	98.5
NASA^{128} + Triplet	**65.5**	**76.5**	**84.6**	**90.9**	**85.8**	**91.4**	**94.8**	**96.9**	**77.4**	**89.8**	**95.7**	**98.6**
Margin^{128}	62.8	73.9	82.4	89.7	78.1	86.2	91.1	94.2	72.7	86.2	93.8	98.0
MDR^{128} + Margin	64.0	75.8	83.5	90.2	82.8	88.4	92.6	95.2	74.2	87.6	94.5	98.1
NASA^{128} + Margin	**65.6**	**76.5**	**84.4**	**90.9**	**83.1**	**89.2**	**93.1**	**95.7**	**74.4**	**87.8**	**94.6**	**98.2**
MS^{128}	61.7	72.8	82.7	90.2	81.1	89.0	93.4	96.4	76.5	89.3	95.1	98.4
MDR^{128} + MS	63.3	74.6	83.8	90.4	84.2	90.1	94.1	96.9	77.0	89.6	95.5	98.6
NASA^{128} + MS	**64.8**	**75.6**	**84.7**	**90.8**	**85.3**	**90.9**	**94.7**	**97.1**	**77.3**	**89.8**	**95.7**	**98.7**
Proxynca++^{128}	62.6	73.9	83.7	90.5	79.2	87.5	92.4	95.8	78.6	90.2	95.8	98.2
MDR^{128} + Proxynca++	64.4	75.7	85.1	91.1	82.8	89.2	93.2	96.3	79.4	90.9	96.3	98.5
NASA^{128} + Proxynca++	**66.0**	**77.1**	**86.0**	**92.1**	**84.0**	**90.6**	**94.3**	**96.7**	**80.0**	**91.4**	**96.5**	**98.8**

Table 2: Retrieval accuracy on three standard image datasets. Bold numbers indicate the best score within the same type of loss.

ing. Cars-196 (Cars) (Krause et al. 2013) has 16,185 images of 196 cars. We split the first 98 cars (8,054 images) for training and the rest 100 cars (8,131 images) for testing. Stanford Online Products (SOP) (Oh Song et al. 2016) contains 120,053 images of 22,634 online products. We use the first 11,318 products (59,551 images) for training and 11,316 products (60,502 images) for testing.

Setting for Real-valued embeddings. Triplet loss (Schroff, Kalenichenko, and Philbin 2015), Margin loss (Wu et al. 2017), MS loss (Wang et al. 2019) and Proxynca++ (Teh, DeVries, and Taylor 2020) are employed as baselines. For the first three losses, Inception with batch normalization (Ioffe and Szegedy 2015) is used as backbone and the input images are cropped to 224×224. The structure embedding network is a two-layer MLP with batch normalization and Softmax activation. We employ Adam optimizer with $1e^{-5}$ weight decay and fix the batch size as 128. For Proxynca++, we follow the settings in the original paper. For CUB, Cars and SOP, the balance factor λ are 4, 2 and 1, respectively. Hyper-parameters: $t = 3$ and $\alpha = 10$.

Test. We directly use sample embeddings to retrieve coarse-grained results. Then, to further prove the effectiveness of the structure embeddings (**NASA**), we re-rank the recalled top-n results, where $n = 2000, 10000, 10000$ for CIFAR-10, NUS-WIDE and FLICKR25K, $n = 32, 32, 2000$ for CUB, Cars and SOP. It is worth noting that our re-ranking only introduces little computational overhead due to a small n.

Results

Results on binary embeddings. The MAP scores on the standard datasets are shown in Table 1. As compared with four state-of-the-art (SOTA) methods, NASA achieves a substantial boost of average 3.33%, 2.95% and 2.72% MAP for CIFAR-10, NUS-WIDE, and FLICKR25K, respectively. Moreover, only using binary sample embeddings (NASA [B]) for retrieval also achieves decent improvements. This shows that benefiting from the point-wise ranking strategy, the neighborhood distribution center of the sample also improves the inter-class discrimination in embedding space.

Results on real-valued embeddings. Retrieval accuracy in Table 2 demonstrates the superiority of our strategy. Specifically, based on NASA, Triplet loss, Margin loss, MS loss and Proxynca++ obtain 0.8%–10.0% R@1 gains on three fine-grained datasets. The above results demonstrate the necessity of NASA, which mines the adaptive neighborhood information of samples to assist embedding learning, thereby improving the feature discrimination and generalization. Table 3 compares the proposed model with SOTA methods. Compared with other methods with ResNet50, NASA performs the best combined with Proxynca++ and achieves very competitive results. Besides, we test the sample embeddings (NASA[R] +Triplet) with k-reciprocal re-ranking (Zhong et al. 2017), which is inferior to ours. From Table 2 and 3, it can be clearly observed the following advantages: **(1)** NASA is a **general-purpose algorithm** that consistently improves

Method	Backbone	CUB			Cars			SOP		
		R@1	R@2	R@4	R@1	R@2	R@4	R@1	R@10	R@10^2
A-BIER^{512} (Opitz et al. 2018)	G	57.5	68.7	78.3	82.0	89.0	93.2	74.2	86.9	94.0
HDML^{512} (Zheng et al. 2019)	R	53.7	65.7	76.7	79.1	87.1	92.1	68.7	83.2	92.4
SoftTriple^{512} (Qian et al. 2019)	BN	65.4	76.4	84.5	84.5	90.7	94.5	78.3	90.3	95.9
Symm^{512} (Gu and Ko 2020)	G	55.9	67.6	78.3	79.1	87.1	92.1	73.2	86.7	94.8
Circle^{512} (Sun et al. 2020)	BN	66.7	77.4	86.2	83.4	89.8	94.1	78.3	90.5	96.1
Proxy Anchor^{512} (Kim et al. 2020)	BN	68.4	79.2	86.8	86.1	91.7	95.0	79.1	90.8	96.2
MDR^{512} (Kim and Park 2021)	BN	68.5	78.9	86.5	88.4	93.1	95.4	80.1	91.4	96.4
DCML-MDW^{512} (Zheng et al. 2021)	R	68.4	77.9	86.1	85.2	91.8	96.0	79.8	90.8	95.8
NASA[R]^{512}+ K-r (Zhong et al. 2017)	BN	68.3	78.5	86.1	88.5	92.8	94.6	77.9	90.1	95.9
NASA^{512} + Triplet	BN	68.8	79.3	86.9	**88.9**	**93.4**	**96.2**	78.5	90.6	96.2
NASA^{512} + Proxynca++	R	**70.2**	**80.4**	**88.0**	88.5	93.3	96.1	**81.2**	**92.1**	**97.0**

Table 3: Comparison with the SOTA methods. Backbone abbreviations: G–GoogleNet, BN–Inception with batch normalization, R–ResNet50. NASA [R] means that only real-valued sample embedding is used.

Method	CUB		Cars		SOP	
	MAP	P@N	MAP	P@N	MAP	P@N
T^{128}	29.3	31.2	29.2	31.2	38.4	39.6
M^{128}	31.9	33.4	32.4	33.8	42.7	41.6
MS^{128}	32.5	34.3	33.2	34.3	45.2	44.3
P^{128}	31.8	32.7	32.9	34.1	48.6	47.1
ProxyNCA^{128}	32.0	32.9	31.2	33.1	43.0	42.2
SoftTriple^{128}	31.4	33.0	32.2	33.6	-	-
Circle^{128}	32.2	33.8	32.8	33.9	43.5	42.8
NASA^{128} + T	34.4	35.4	**36.5**	**37.8**	45.5	44.7
NASA^{128} + M	34.6	33.4	34.1	35.4	43.1	42.4
NASA^{128} + MS	33.3	35.0	35.7	36.6	45.4	44.6
NASA^{128} + P	**35.0**	**36.3**	34.2	35.2	**52.7**	**50.9**

Table 4: The MAP and P@N scores on three datasets, where N is the number of database images in each category. Some abbreviations: T-Triplet, M-Margin, P-Proxynca++.

all baselines. (**2**) NASA is a very **effective algorithm**. A simple baseline (*e.g.,* Triplet loss) boosted by NASA can surpass many SOTAs. These results establish the importance of local neighborhood structure in DML.

To prove the effect on the ranking task, we examine NASA with MAP and P@N metrics in Table 4. Even if the intra-class variance of these datasets is large, our NASA model achieves consistent ranking improvements compared to baselines and other methods. Besides, incorporating structure embeddings into the retrieval process leads to very small time costs. For example, since the re-rank is only performed on top-32 images on CUB, total time costs are 1.88 ms and 1.97 ms per query for Triplet and Ours, respectively (unoptimized Python implementation).

Ablation Study

We choose Triplet loss to train models on CUB and Cars.
Hyper-parameters. In Fig. 5, properly adjusting the factor λ to balance the global and local embedding structure is beneficial to the performance. Besides, as the margin β or t grows, the performance increases at first and reaches the best results, then gradually decreases as a whole. A small margin may not fully exploit the underlying information, while a large margin may destroy the adaptive neighborhood structure.

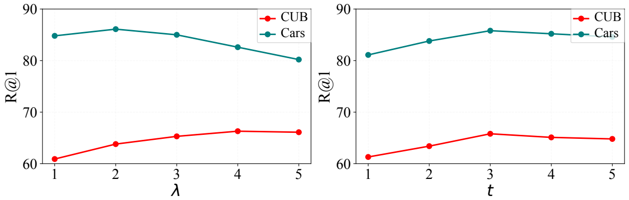


Figure 5: Accuracy in R@1 versus balance factor λ and margin factor t on CUB and Cars.

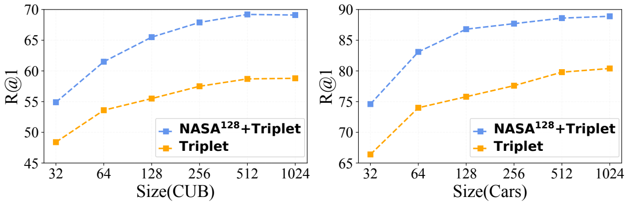


Figure 6: Accuracy in R@1 versus embedding size.

Embedding dimension. The trade-off between speed and accuracy is essential in large-scale retrieval, in which embedding size is the key factor. Therefore, we vary the embedding size and evaluate our model in Fig. 6. As the size grows, our model has a significant increase in accuracy, and when the size is greater than 128, the accuracy of Triplet loss has gradually become saturated.

Conclusion

We propose a novel Neighborhood-Adaptive Structure Augmented (NASA) metric learning framework to learn the adaptive structure embeddings in a simple self-supervised manner. Without any modifications, most existing indexing techniques can be used to support large-scale retrieval with NASA seamlessly. Extensive experiments show the superior retrieval performance of our method. We will explore more sophisticated tasks for structure embeddings learning.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (62121002, 62022076, U1936210), the Fundamental Research Funds for the Central Universities under Grant WK3480000011. We acknowledge the support of GPU cluster built by MCC Lab of Information Science and Technology Institution, USTC.

References

Abdi, H.; and Williams, L. J. 2010. Principal component analysis. *WIREs Comp Stats*.

Chang, J.; Lan, Z.; Cheng, C.; and Wei, Y. 2020. Data uncertainty learning in face recognition. In *CVPR*.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *CIVR*.

Dai, B.; Guo, R.; Kumar, S.; He, N.; and Song, L. 2017. Stochastic generative hashing. In *ICML*.

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *CVPR*.

Fu, Z.; Li, Y.; Mao, Z.; Wang, Q.; and Zhang, Y. 2021. Deep Metric Learning with Self-Supervised Ranking. In *AAAI*.

Ge, J.; Xie, H.; Min, S.; and Zhang, Y. 2021. Semantic-guided Reinforced Region Embedding for Generalized Zero-Shot Learning. In *AAAI*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. In *NeurIPS*.

Gu, G.; and Ko, B. 2020. Symmetrical Synthesis for Deep Metric Learning. In *AAAI*.

Huang, J.-T.; Sharma, A.; Sun, S.; Xia, L.; Zhang, D.; Pronin, P.; Padmanabhan, J.; Ottaviano, G.; and Yang, L. 2020. Embedding-Based Retrieval in Facebook Search. In *SIGKDD*.

Huiskes, M. J.; and Lew, M. S. 2008. The MIR flickr retrieval evaluation. In *MIR*.

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

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product Quantization for Nearest Neighbor Search. *TPAMI*.

Kim, S.; Kim, D.; Cho, M.; and Kwak, S. 2020. Proxy anchor loss for deep metric learning. In *CVPR*.

Kim, Y.; and Park, W. 2021. Multi-level Distance Regularization for Deep Metric Learning. In *AAAI*.

Ko, B.; and Gu, G. 2020. Embedding Expansion: Augmentation in Embedding Space for Deep Metric Learning. In *CVPR*.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *ICCV workshops*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Lee, J. A.; and Verleysen, M. 2007. *Nonlinear Dimensionality Reduction*. Springer–Verlag New York. ISBN 978-0-387-39350-6.

Li, J.; Xie, H.; Li, J.; Wang, Z.; and Zhang, Y. 2021a. Frequency-aware Discriminative Feature Learning Supervised by Single-Center Loss for Face Forgery Detection. In *CVPR*.

Li, P.; Xie, H.; Min, S.; Zha, Z.-J.; and Zhang, Y. 2021b. Online Residual Quantization via Streaming Data Correlation Preserving. *TMM*.

Li, S.; Chen, Z.; Lu, J.; Li, X.; and Zhou, J. 2019. Neighborhood preserving hashing for scalable video retrieval. In *ICCV*.

Lin, F.; Xie, H.; Li, Y.; and Zhang, Y. 2021. Query-Memory Re-Aggregation for Weakly-supervised Video Object Segmentation. In *AAAI*.

Lin, K.; Lu, J.; Chen, C.-S.; and Zhou, J. 2016. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*.

Liu, Z.; Qian, P.; Wang, X.; Zhu, L.; He, Q.; and Ji, S. 2021. Smart Contract Vulnerability Detection: From Pure Neural Network to Interpretable Graph Feature and Expert Pattern Fusion. In *IJCAI*.

Malkov, Y. A.; and Yashunin, D. A. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *TPAMI*.

Min, S.; Yao, H.; Xie, H.; Wang, C.; Zha, Z.-J.; and Zhang, Y. 2020a. Domain-aware visual bias eliminating for generalized zero-shot learning. In *CVPR*.

Min, S.; Yao, H.; Xie, H.; Zha, Z.-J.; and Zhang, Y. 2020b. Multi-objective matrix normalization for fine-grained visual recognition. *TIP*.

Movshovitz-Attias, Y.; Toshev, A.; Leung, T. K.; Ioffe, S.; and Singh, S. 2017. No fuss distance metric learning using proxies. In *ICCV*.

Musgrave, K.; Belongie, S.; and Lim, S.-N. 2020. A Metric Learning Reality Check. In *ECCV*.

Oh, S. J.; Murphy, K. P.; Pan, J.; Roth, J.; Schroff, F.; and Gallagher., A. C. 2019. Modeling uncertainty with hedged instance embeddings. In *ICLR*.

Oh Song, H.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep metric learning via lifted structured feature embedding. In *CVPR*.

Opitz, M.; Waltner, G.; Possegger, H.; and Bischof, H. 2018. Deep metric learning with bier: Boosting independent embeddings robustly. *TPAMI*.

Peng, H.; Zhang, R.; Dou, Y.; Yang, R.; Zhang, J.; and Yu, P. S. 2021. Reinforced Neighborhood Selection Guided Multi-Relational Graph Neural Networks. *TOIS*.

Qian, Q.; Shang, L.; Sun, B.; Hu, J.; Li, H.; and Jin, R. 2019. Softtriple loss: Deep metric learning without triplet sampling. In *ICCV*.

Radenović, F.; Tolias, G.; and Chum, O. 2019. Fine-Tuning CNN Image Retrieval with No Human Annotation. *TPAMI*.

Roth, K.; Brattoli, B.; and Ommer, B. 2019. Mic: Mining interclass characteristics for improved metric learning. In *ICCV*.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*.

Sebe, N.; Lew, M. S.; and Huijsmans, D. P. 2000. Toward improved ranking metrics. *TPAMI*.

Shen, Y.; Qin, J.; Chen, J.; Yu, M.; Liu, L.; Zhu, F.; Shen, F.; and Shao, L. 2020. Auto-Encoding Twin-Bottleneck Hashing. In *CVPR*.

Shi, Y.; and Jain, A. K. 2019. Probabilistic Face Embeddings. In *ICCV*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*.

Sohn, K. 2016. Improved deep metric learning with multi-class n-pair loss objective. *NeurIPS*.

Song, J.; He, T.; Gao, L.; Xu, X.; Hanjalic, A.; and Shen, H. T. 2018. Binary generative adversarial networks for image retrieval. In *AAAI*.

Su, S.; Zhang, C.; Han, K.; and Tian, Y. 2018. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *NeurIPS*.

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 *CVPR*.

Teh, E. W.; DeVries, T.; and Taylor, G. W. 2020. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *ECCV*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. Technical report.

Wang, X.; Han, X.; Huang, W.; Dong, D.; and Scott, M. R. 2019. Multi-Similarity Loss With General Pair Weighting for Deep Metric Learning. In *CVPR*.

Wang, X.; Zhang, H.; Huang, W.; and Scott, M. R. 2020a. Cross-Batch Memory for Embedding Learning. In *CVPR*.

Wang, Y.; Xie, H.; Zha, Z.-J.; Xing, M.; Fu, Z.; and Zhang, Y. 2020b. Contournet: Taking a further step toward accurate arbitrary-shaped scene text detection. In *CVPR*.

Wu, C.; Manmatha, R.; Smola, A. J.; and Krähenbühl, P. 2017. Sampling Matters in Deep Embedding Learning. In *ICCV*.

Yang, E.; Deng, C.; Liu, T.; Liu, W.; and Tao, D. 2018. Semantic structure-based unsupervised deep hashing. In *IJCAI*.

Yang, E.; Liu, T.; Deng, C.; Liu, W.; and Tao, D. 2019. Distillhash: Unsupervised deep hashing by distilling data pairs. In *CVPR*.

Zhang, L.; Zhang, Y.; Tang, J.; Lu, K.; and Tian, Q. 2013. Binary code ranking with weighted hamming distance. In *CVPR*.

Zheng, W.; Chen, Z.; Lu, J.; and Zhou, J. 2019. Hardness-Aware Deep Metric Learning. In *CVPR*.

Zheng, W.; Wang, C.; Lu, J.; and Zhou, J. 2021. Deep Compositional Metric Learning. In *CVPR*.

Zhong, Z.; Zheng, L.; Cao, D.; and Li, S. 2017. Re-ranking person re-identification with k-reciprocal encoding. In *CVPR*.