Spherical Feature Transform for Deep Metric Learning

Yuke Zhu*[1], Yan Bai *[2], and Yichen Wei[1]

[1] MEGVII Inc. {zhuyuke, weiyichen}@megvii.com
[2] Tongji Unversity, China yan.bai@tongji.edu.cn

Abstract. Data augmentation in feature space is effective to increase data diversity. Previous methods assume that different classes have the same covariance in their feature distributions. Thus, feature transform between different classes is performed via translation. However, this approach is no longer valid for recent deep metric learning scenarios, where feature normalization is widely adopted and all features lie on a hypersphere.
This work proposes a novel spherical feature transform approach. It relaxes the assumption of identical covariance between classes to an assumption of similar covariances of different classes on a hypersphere. Consequently, the feature transform is performed by a rotation that respects the spherical data distributions. We provide a simple and effective training method, and in depth analysis on the relation between the two different transforms. Comprehensive experiments on various deep metric learning benchmarks and different baselines verify that our method achieves consistent performance improvement and state-of-the-art results.

1 Introduction

It is crucial to have sufficient data diversity in deep metric learning. A common practice is to augment data in the image space. This is effective but has limited effect. Specifically, it is hard to generate variances in one class using the information in the other classes.

Directly augmenting data in the feature space has become a new trend [6, 12, 13,17,31,33,34]. Specifically, Yin .etal. [31] propose a simple method that requires no extra labeling and is easy to implement. It assumes that the example features in each class follow a Gaussian distribution, and the covariance between all classes is the same, thus shared. Each feature is the summation of the class-dependent mean and a class-independent variance. Thus, given existing features in one class, their variance parts can be transferred to generate *new* features in other classes, via a translation. This is illustrated in Fig. 1(a). It is shown effective in [31].

Recently, feature normalization is widely adopted in deep metric learning [4, 18,26–29]. In this case, all features lie on the surface of a hypersphere. The feature transfer approach [31] becomes inappropriate. First, a Gaussian distribution is

* Equal contribution.

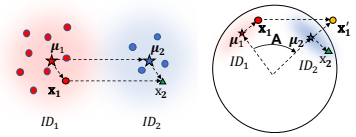


Fig. 1. Illustration of two feature transforms. (a) *translation transform* [31]. The feature of ID_1 and ID_2 are sampled from Gaussian distributions with mean value $\boldsymbol{\mu}_1$,$\boldsymbol{\mu}_2$ and identical covariance. To increase the intra-class variances of ID_2, feature $\mathbf{x}_2$ is generated by translating $\mathbf{x}_1$ by $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$. (b) Illustration of *translation transform* and SFT on a sphere. Directly translating $\mathbf{x}_1$ from ID_1 to ID_2 will result in $\mathbf{x}_1'$, which is out of the surface of the sphere. Our spherical feature transform performs a rotation, such that feature $\mathbf{x}_1$ of ID_1 is transferred to $\mathbf{x}_2$ of ID_2.

no longer correct. A proper spherical distribution should be used instead. Second, although each class can be approximated as a local Gaussian on the sphere, the assumption of identical covariance between classes is less valid. Last, feature translation would produce an invalid feature that is out of the surface of the hypersphere, as shown in Fig. 1(b). Therefore, both the prior and the feature transform should be adapted for the spherical case.

This work proposes *spherical feature transform* to resolve above problems. It assumes that distributions of features of different classes are spherical-homoscedastic [9]. This relaxes the previous assumption that identical covariance between classes. Instead, it assumes all classes have *similar* covariances, where the similarity is measured by equivalence of eigenvalues of the covariance matrices. Consequently, the transformation between two classes is a rotation that is characterized by the classes' means. This is illustrated in Fig. 1(b). Theoretical analysis reveals that our approach is a generalization of [31].

Our method is simple and general. It is validated on several deep metric learning tasks. Comprehensive experiments and ablation studies demonstrate its effectiveness.

2 Related Work

Feature augmentation is a relatively new topic. Some researchers [6, 21, 33, 34] adopt an adversarial approach to generate hard features from the observed negative samples utilizing the Generative Adversarial Networks (GAN) [7]. Their main focus is to generate hard negative features. While the structure of feature distributions is not considered. Also, the training process with GAN is usually complicated and unstable [1]. Dixit .etal. [5] propose a data augmentation method using attribute-guided feature descriptor for generation. Liu .etal. [13] propose to learn a pose manifold in the feature space and use it to synthesize pose-augmented features. However, these works need extra labeling for supervision.

Recently, Lin .etal. [12] utilize the variational inference to disentangle intra-class variance and leverages the distribution to generate discriminative samples to improve robustness. This work and ours share similar insight that the variances of different class can be regarded as similar. But their method is based on the assumption that the variances can be fully disentangled and can be modeled using a Gaussian. While our method makes no assumptions about this. In fact, we will show that when features are on a hypersphere, the intra-class variances can not be modeled using one distribution. The most similar work to ours is in [31]. This work also models the variances using a Gaussian. It proposes to transfer the variance part from one class to the other for feature augmentation. It will be detailedly introduced in Sec 3.1. However, both two works do not considering the widely adopted feature normalization techniques and its influence on feature distributions.

3 Proposed Approach

3.1 Review of Feature Transform

Feature transform is an approach for feature generation by transferring the intra-class variance from one class to the others. It is based on the assumption that features from each class follow a Gaussian distribution and the distributions of different classes have different mean values but shared covariances. Using this assumption, a feature $\mathbf{x}$ is represented by two parts:

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\sigma}, \tag{1}$$

where $\boldsymbol{\mu}$ is the mean value of the class that $\mathbf{x}$ belongs to. $\boldsymbol{\sigma}$ is the variance part sampled from a zero-mean Gaussian. $\boldsymbol{\mu}$ contains the information of identity of the class. $\boldsymbol{\sigma}$ contains the information of intra-class variance that is shared among classes.

Following this prior, Feature Transfer Learning (FTL) [31] is proposed to transfer the variance part from one class to the others for feature generation. Specifically, given a feature $\mathbf{x}_1$ with $\mathbf{x}_1 = \boldsymbol{\mu}_1 + \boldsymbol{\sigma}_1$ and the center of a target class $\boldsymbol{\mu}_2$. The feature generation is proceeded by $\tilde{\mathbf{x}}_2 = \boldsymbol{\mu}_2 + \boldsymbol{\sigma}_1$, where $\tilde{\mathbf{x}}_2$ is regarded as belonging to the target class but shares identical variance with $\mathbf{x}_1$. We illustrate this process in Fig. 1(a). The feature transform can also be written as

$$\tilde{\mathbf{x}}_2 = \mathbf{x}_1 + \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1. \tag{2}$$

It can be interpreted as translating the feature $\mathbf{x}_1$ by $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$. Thus, this method is referred to as *translation transform*.

3.2 Review of Spherical-homoscedasticity

Spherical-homoscedasticity is a property describing the relationship between a set of data distributions on the sphere, which we refer to as spherical distributions. It is proposed by Onur C .etal. [9].

The definition of spherical-homoscedasticity resorts to the Gaussian approximation. We first give the definition of Gaussian approximation and then give the definition of spherical-homoscedasticity.

Definition 1. *Suppose* $\mathbf{x}_i$ *is a sample from the spherical distribution. Then the Gaussian approximation is given as* $N(E(\mathbf{x}_i), Var(\mathbf{x_i}))$*, where* $E(.)$ *and* $Var(.)$ *are the functions for expectation and variances.*

Definition 2. *Suppose distribution* $N_1(\boldsymbol{\mu}, \Sigma)$ *is the Gaussian approximation of spherical distribution* F_1 *and* $\mathbf{A}$ *is an orthogonal matrix. Suppose* $\mathbf{A}$ *is spanned by* $\boldsymbol{\mu}$ *and one of the eigenvectors of* $\boldsymbol{\Sigma}$*. Suppose* $N_2\left(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}^T\Sigma\mathbf{A}\right)$ *is the Gaussian approximation of spherical distribution* F_2*. Then* N_1 *and* N_2 *(*F_1 *and* F_2*) are spherical-homoscedastic.*

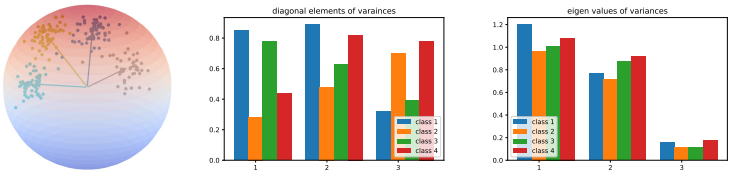
Spherical-homoscedasticity requires the covariances of distributions to have identical eigenvalues. Geometrically, this property indicates that distributions share identical shape. In other words, distributions can be transformed to be totally overlapped.

3.3 Spherical Feature Transform

Recently, *feature normalization* has been widely discussed [18,26,27] and adopted in DML frameworks [4,27–29]. This technique scales all the features to the same norm. Thus, the features are restricted to lie on the surface of a hypersphere. In this case, the feature transform in Eq. 2 is no longer valid. There are two reasons. First, the identical-variance prior is too restrictive for spherical distributions. In general(e.g. the two distributions in Fig. 1(b)), spherical distributions are unlikely to have the same covariance. Second, *translation transform* produces features lying out of the surface of the hypersphere. This breaks the manifold structure of the feature space as shown in Fig. 1(b). Therefore, both the identical-variance prior and the *translation transform* should be modified for the spherical case.

We propose a new approach. It relaxes the identical-variance prior to the prior of identical eigen values of variances, which is the spherical-homoscedasticity as defined in Sec. 3.2. This relaxation is validated Fig. 2. The experiment is performed on CUB dataset (see experiments for details). We choose four classes with sufficient number of samples so that their feature distributions can be faithfully estimated. We compare their covariance matrices and the eigenvalues. As shown in Fig. 2(b), their covariance matrices are significantly different, but the difference of the eigen values of these covariance matrices are much smaller (about 8% on average) as shown in Fig. 2(c). This shows that the identical-variance prior does not hold. And our assumption of identical eigenvalues of covariances is more valid. The similar observation is also found on other datasets in face recognition, vehicle recognition and etc.

Geometrically, our assumpion implies that a distribution can be transformed to overlap with another via an orthogonal rotation matrix as in the Definition 2. Thus, a feature vector in one class can be transformed to another class to generate augmented features. We denote the Gaussian approximation of two classes distributions as $N_1(\boldsymbol{\mu_1}, \Sigma_1)$ and $N_2(\boldsymbol{\mu_2}, \Sigma_2)$. Given a feature $\mathbf{x}_1$ sampled



(a) (b) (c)

Fig. 2. (a) Visualization of features on CUB dataset. Features are projected to 3D using PCA. (b) Diagonal elements of four classes' variances from CUB. The values from the same position on the diagonal are plotted together. They differ a lot. (c) Eigen values of four classes' variances from CUB. The eigen values from the same position on the eigen matrices are plotted together. They are much closer.

from N_1, we have:

$$\tilde{\mathbf{x}}_2 = \mathbf{A}\mathbf{x}_1, \tag{3}$$

where $\tilde{\mathbf{x}}_2$ is considered as belonging to the class of N_2. This method is called *Spherical Feature Transform (SFT)*.

However, we note that solving the orthogonal matrix $\mathbf{A}$ according to Definition 2. is non-trivial. A brute force approach would be complex. We propose a simpler and more elegant approach to calculate $\mathbf{A}$ without solving matrix equations. It is presented in the Proposition 1.

Proposition 1. *Suppose* $N_1(\boldsymbol{\mu_1}, \Sigma_1)$ *and* $N_2(\boldsymbol{\mu_2}, \Sigma_2)$ *are two Gaussian approximations of spherical distributions. If they are spherical-homoscedastic, then the rotation matrix between them is spanned by* $\boldsymbol{\mu}_1$ *and* $\boldsymbol{\mu}_2$.

The proof of Proposition 1 is left in the supplement. The rotation matrix $\mathbf{A}$ is calculated as following. First, we apply Schmidt orthogonalization to obtain $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$: $\mathbf{n}_1 = \boldsymbol{\mu}_1$, $\mathbf{n}_2 = \frac{\boldsymbol{\mu}_2 - (\boldsymbol{\mu}_2^T \mathbf{n}_1)\mathbf{n}_1}{\left\|\boldsymbol{\mu}_2 - (\boldsymbol{\mu}_2^T \mathbf{n}_1)\mathbf{n}_1\right\|_2}$. Then, we use Rodrigues rotation formula to calculate the rotation matrix:

$$\mathbf{A} = \mathbf{I} + (\mathbf{n}_2\mathbf{n}_1^T - \mathbf{n}_1\mathbf{n}_2^T)\sin(\alpha) + (\mathbf{n}_1\mathbf{n}_1^T + \mathbf{n}_2\mathbf{n}_2^T)(\cos(\alpha) - 1), \tag{4}$$

where $\mathbf{I}$ is the identity matrix and α is the rotation angle between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

3.4 Theoretical Analysis

We discuss the relation between proposed SFT in Eq.(3) and the *translation transform* in Eq.(2). In general, the two transforms are different. However, we show that a simple variant of the *translation transform* (for normalized features) under some special cases is a degenerated form of SFT. Actually, we use this variant as a baseline method in our experiment.

In *translation transform*, the variance part $\boldsymbol{\sigma}$ defined in Eq.(1) is assumed to have the same distribution among all the classes. Differently, we propose SFT by showing that this term should be orthogonal transformed when features are

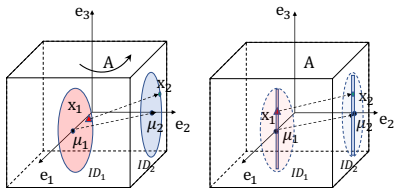


Fig. 3. Illustration of the degeneration from SFT to *translation transform* by taking a three-dimensional example. The $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ are three axis of the coordinate. The red and blue ellipses represents the distribution of ID_1 and ID_2. Suppose they are spherical-homoscedastic and the rotation matrix between them is $\mathbf{A}$ and $\mathbf{x}_2 = \mathbf{A}\mathbf{x}_1$ (a) In general $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$ is not equivalent to $\mathbf{x}_2 - \mathbf{x}_1$. (b) Special case: The intra-class variances are now encoded by the one-dimensional space spanned by $\mathbf{e}_3$ and the $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$ is equivalent to $\mathbf{x}_2 - \mathbf{x}_1$.

normed. We observed that, when well trained, the features sampled from $\boldsymbol{\sigma}$ are likely to lie in the invariant subspace of the orthogonal matrix $\mathbf{A}$, as defined in Definition 2. This observation is experimentally validated in Sec 4.2. We show that in this case SFT degenerates to *translation transform* defined in Eq. 2. With this condition $\mathbf{A}\boldsymbol{\sigma}_\mathbf{1} = \boldsymbol{\sigma}_\mathbf{1}$, Eq. (3) is simplified as

$$\tilde{\mathbf{x}}_2 = \mathbf{A}\mathbf{x}_1 = \mathbf{A}(\boldsymbol{\mu}_1 + \boldsymbol{\sigma}_1) = \boldsymbol{\mu}_2 + \boldsymbol{\sigma}_1. \tag{5}$$

The right side is the *translation transform* in Eq.(2).

This degeneration case is a bit hard to understand, especially in high dimensional space. For an intuitive illustration, we show an example in three-dimensional space. As shown in Fig. 3(a), in general, the result of SFT $\mathbf{x}_2 = \mathbf{A}\mathbf{x}_1$ is not equal to $\mathbf{x}_1 + \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$. While, some special features stay equal after rotation and translation, as shown in Fig. 3(b). In such a case, the direction of $\boldsymbol{\sigma}_1$ is parallel to the rotation axis of $\mathbf{A}$. That is, $\boldsymbol{\sigma}_1$ lie in the invariant subspace of $\mathbf{A}$.

Proposition 2. *The degeneration happens only before feature normalization.*
Proof Suppose feature $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\sigma}$ and its variance part $\boldsymbol{\sigma}$ lie in the invariant subspace of $\mathbf{A}$. As $\mathbf{A}$ is spanned by $\boldsymbol{\mu}$ and one of the other vector, $\boldsymbol{\mu}$ is orthogonal to the invariant subspace of $\mathbf{A}$. So $\boldsymbol{\mu}$ is orthogonal to σ. Then the norm of $\mathbf{x}$ is evaluated as:

$$\|\mathbf{x}\| = \|\boldsymbol{\mu} + \boldsymbol{\sigma}\| = \sqrt{(\boldsymbol{\mu} + \boldsymbol{\sigma})^\mathbf{T}(\boldsymbol{\mu} + \boldsymbol{\sigma})} = \sqrt{\boldsymbol{\mu}^\mathbf{T}\boldsymbol{\mu} + \boldsymbol{\sigma}^\mathbf{T}\boldsymbol{\sigma}} \tag{6}$$

As $\boldsymbol{\mu}$ is a constant for one class and the $\boldsymbol{\sigma}$ varies, the norm of $\mathbf{x}$ can not be a constant for each features of a class. In other words, the feature norms are not constant. □

Based on Proposition 2 , we can make a simple modification to the *translation transform* to make it able to produce valid features in spherical case. Specifically, we use Eq. 2 before feature normalization and then reproject them back to the hypersphere. This variant is referred to as the degenerated form of SFT.

However, the degenerated form will produce identical augmented features as SFT only when degeneration takes place. There are still features that won't obey the condition of degeneration. Directly applying the degenerated form on them may curse the augmentation process. We further investigate into whether there is an ideal case where the degeneration will always take place thus the degenerated form can be treated as an alternative of SFT. Considering the condition of degeneration, this special case should satisfy $\mathbf{A}\boldsymbol{\sigma} = \boldsymbol{\sigma}$ for any rotation matrix $\mathbf{A}$ and any $\boldsymbol{\sigma}$. The three-dimensional example in Fig. 3(b) gives a clear clue that this special case exist mathematically. Specifically, if the feature distributions are shrunk in the plane defined by $\{\mathbf{e_1}, \mathbf{e_2}\}$, then all $\boldsymbol{\sigma}$ will lie in the invariant subspace of $\mathbf{A}$. The exact mathematical description for such case is presented in Proposition 3

Proposition 3. *SFT degenerates to the* translation transform *iff for* $\forall$ *feature* $\mathbf{x}$ *with* $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\sigma}$, $\boldsymbol{\mu} \perp \boldsymbol{\sigma}$

The proof is presented in the supplement. Proposition 3 has revealed a extremely restrictive condition that the mean vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ lie in two orthogonal subspaces. Intuitively, this condition is hard to be satisfied. While surprisingly, it is found, although not clear why, but in general, that deep neural networks tend to learn an orthogonal subspaces for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. For revealing this phenomenon, we define a measure of how much the the subspaces of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are orthogonal. We first define two covariance matrices:

$$\mathbf{S}_c = \frac{1}{C}\sum_{i=1}^{C} {\boldsymbol{\mu}_i}^T \boldsymbol{\mu}_i, \mathbf{S}_w = \frac{1}{C}\sum_{i=1}^{C}\frac{1}{N_k}\sum_{k=0}^{N_k}\left(\mathbf{x}_k - \boldsymbol{\mu}_i\right)^T\left(\mathbf{x}_k - \boldsymbol{\mu}_i\right), \tag{7}$$

where y_k is the label of embedding $\mathbf{x_k}$. C is the number of classes. N_k is the number of samples for k-th class. Then, we estimate the eigenvalue space for $\mathbf{S}_c$ and denote them as $\mathbf{U}$, where $\mathbf{U} = \{\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_k}\}$ corresponds to the k largest eigenvalues. The subspace spanned by $\mathbf{U}$ will cover most energy of the mean vectors while the energy of σ will distribute over these components. We calculate the remaining energy percent of σ in this subspace by evaluating:

$$r_w = trace\left(\mathbf{U}^T\mathbf{S}_w\mathbf{U}\right)/trace(\mathbf{S}_w), \tag{8}$$

where $trace(.)$ is the sum of the diagonal elements. r_w measures that how much energy percent of $\boldsymbol{\sigma}$ is distributed over the subspace spanned by $\mathbf{U}$. r_w is between 0 and 1. If $r_w = 0$, then the subspaces for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are orthogonal. So the smaller r_w, the nearer of the state being orthogonal.

3.5 Training Scheme

Both the *translation transform* defined in Eq. 2 and SFT defined in Eq. 3 rely on the accurate estimation of the feature center of each class. We denote the feature centers as $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_C\}$ where C is the number of classes. In every mini-batch, we update them by:

$$\Delta\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m}\delta\left(y_i = j\right)\cdot\left(\boldsymbol{\mu}_j - \mathbf{x}_i\right)}{1 + \sum_{i=1}^{m}\delta\left(y_i = j\right)}, \tag{9}$$

where y_i is the label of feature $\mathbf{x}_i$ and m is the mini-batch size. $\delta(.)$ is the indicator function. For training, we propose two train schemes depending on the whether the training set is balanced.

Balanced train. When the dataset is balanced in the number of samples for each class, we will generate new features for every class. In specific, for a feature, we randomly choose a different class as target and transform the feature to that class. We do this for every feature in the mini-batch. After that, we get a new batch of features with different labels.

Unbalanced train. When the dataset is unbalanced in the number of samples for each class, we only generate new features for classes that are short of samples. In specific, we set a threshold for the number of samples and use it to separate the whole training data into head classes and tail classes. For any head features in a mini-batch, we randomly choose a tail class as target and transform it into the tail distributions.

For both training schemes, we get two batch of training data. Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ be the original features and $\mathbf{Y} = [y_1, \cdots, y_n]$ be the corresponding labels, where $y_i \in \{1, \cdots, C\}$. Let $\mathbf{X}_{gen} = [\mathbf{x}_{gen,1}, \cdots, \mathbf{x}_{gen,m}]$ be the generated batch and $\mathbf{Y}_{gen} = [y_{gen,1}, \cdots, y_{gen,m}]$ be the corresponding labels. As our augmentation method is applicable to any DML frameworks, we denote $J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y})$ as a general target function with $\boldsymbol{\theta}$ denoting the parameters to be optimized and $\mathbf{X}$, $\mathbf{Y}$ denoting the batch data and labels. Similar to DVML [12] and HDML [34], we also apply the metric learning losses on the original features besides the augmented features. It is because that the augmentation process relies on a well trained feature space. Omitting the original features or applying too much weight on the augmented features will curse the training process. It is shown in Sec 4.2. We formulate our losses as:

$$\min_{\boldsymbol{\theta}} J = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y}) + \lambda J(\boldsymbol{\theta}; \mathbf{X}_{gen}, \mathbf{Y}_{gen}), \tag{10}$$

where λ is a weighting factor controlling the balance between the original batch data and the generated batch data. The total training scheme for feature transform is illustrated in Algorithm 1.

Algorithm 1 Training with Feature Transform

Input: Training image set, network f, target function J, parameters λ and number of iteration numbers T.

Output: Parameters of network $\boldsymbol{\theta}$

```
Initialize θ
for iter = 1, ..., T do
    Sample mini-batch of m training images.
    Extract embeddings using f to get X with labels Y.
    Produce data {X_gen, Y_gen} using (3) or (2).
    Update geometric centers using (9).
    Optimize θ using (10).
end for
```

4 Experiments

Datasets and Metrics. We conduct experiments on two types of benchmark datasets: Metric Learning and Face Recognition. For metric learning, we experiment on three widely-used benchmarks to evaluate the our approach: (1)**Cars196** [11], (2)**CUB-200-2011** [25], (3)Stanford Online Products (**SOP**) [16]. To evaluate the performance of each method, we follow [6] to perform the K-means algorithm in the test set and report normalized mutual information (NMI) and F_1 metrics as well as Recall@K for retrieval task. For face recognition, we use a cleaned version of MS-Celeb-1M [8] as our training set that contains 3M facial images and 80920 classes. We present evaluation results on three face verification benchmarks: **LFW** [10], **YTF** [30] and **IJB-C** [15]. For LFW and YTF, we follow the unrestricted with labeled outside data protocol and report the performance of 6,000 face pairs on LFW and 5,000 video pairs on YTF. For IJB-C, we follow the 1:1 verification protocol to evaluate 19,557 positive matches and 15,638,932 negative matches and report the results of TARs at various FARs.

Implementation Details. For the metric learning task, we use GoogleNet [23](or GoogleNet-V2) pre-trained with ImageNet [3] as a backbone network and add a fully connected layer at the end to output the feature embedding. We use the same data preprocessing and augmentation as in Multi-Similarity Loss [29]. We set the embedding size to 512 and perform ℓ_2-normalization on the feature. We use the SGD optimizer with a weight decay of 1e-4 and train for 30,000 iterations. For learning rate, we set 1e-2 for Cars196 and SOP and 1e-3 for CUB-200-2011 as base learning rate for backbone and newly added layers 10x the base learning rate, and decay the learning rate by multiply 0.1 every 10,000 iterations. We set the batch size to be 60 made up of 20 classes and 3 images per class. The balanced train scheme is adopted when SFT is used. For face recognition, the CNN architecture used in our work is similar to [14]. We change the number of residual units to $[3, 4, 6, 3]$ to construct a 34-layer residual network. We preprocess all face images by MTCNN [32]. Then the 5 facial points are adopted to perform alignment to the face image. After that, we resize the cropped image to 112×112. Each pixel(in [0, 255]) in RGB images is normalized by subtracting 127.5 then being divided by 128. We use SGD optimizer with a weight decay of 5e-4 and train for 120K iterations. The learning rate is set to 0.1 initially and is divided by 10 at the 70K, 90K and 110K iterations. The unbalanced train scheme is adopted when SFT is used, where we set the classes that have less than 15 samples as tail classes.

Compared Methods. We compare our method to other feature generation methods, including HDML [34], DVML [12] and FTL [31]. These methods are introduced in Sec 2. They require no extra labeling and can be compared fairly on metric learning tasks. Also the degenerated SFT will be included for comparison. It is denoted as SFT-d in the results. The comparison is made on two traditional representative baseline losses, aka, triplet loss [19] and NPair loss [20] and two most recent baseline losses that achieved high results, aka, Ranked List

Table 1. Comparison on Cars196 and CUB-200-2011.

	Cars196					CUB-200-2011				
	R@1	R@2	R@4	MNI	F1	R@1	R@2	R@4	NMI	F1
GoogleNet										
Triplet	58.4	70.3	80.2	57.0	27.2	42.8	55.2	55.6	52.4	19.1
Triplet+HDML [34]	62.0	73.3	82.9	57.7	27.8	44.3	56.0	68.0	55.5	26.7
Triplet+DVML [12]	64.4	73.5	78.6	**60.5**	28.4	43.3	55.8	68.0	55.0	25.2
Triplet+FTL	60.1	71.5	80.5	57.9	25.0	46.8	59.2	70.2	57.3	24.3
Triplet+SFT-d	60.3	71.7	81.4	57.9	28.1	46.5	59.3	70.0	57.9	28.1
Triplet+SFT	**65.1**	**75.7**	**84.0**	58.1	**28.6**	**48.3**	**60.0**	**71.2**	**58.1**	**28.6**
NPair	72.8	82.3	88.5	61.3	29.4	53.5	64.9	72.3	60.4	27.8
NPair+HDML [34]	78.9	87.0	91.0	67.1	37.3	53.9	65.8	76.7	62.0	30.0
NPair+DVML [12]	**80.2**	85.6	91.9	66.1	34.8	54.2	66.2	77.3	62.0	31.5
NPair+FTL	73.1	82.2	88.6	60.0	27.4	54.0	66.0	77.0	61.9	29.7
NPair+SFT-d	76.2	85.0	90.9	64.2	33.1	54.5	67.0	**77.7**	62.0	30.1
NPair+SFT	79.4	**87.1**	**92.4**	**67.2**	**37.3**	**54.7**	**67.0**	77.5	**62.2**	**30.5**
GoogleNet-V2										
RLL [28]	74.2	83.2	89.0	62.2	32.9	59.6	71.0	80.5	64.3	32.9
RLL+DVML [12]	79.0	86.6	91.3	65.5	34.9	60.2	71.7	81.0	64.7	33.0
RLL + SFT-d	78.8	86.7	92.1	65.4	34.4	59.4	71.2	80.9	64.2	32.8
RLL + SFT	**80.2**	**88.1**	**92.8**	**66.1**	**35.3**	**60.3**	**71.8**	**81.1**	**64.9**	**33.6**
MS [29]	84.0	90.2	94.1	72.8	45.3	65.7	76.6	84.6	69.0	39.6
MS+DVML [12]	84.4	**90.8**	92.4	72.0	45.3	66.2	76.7	85.1	69.6	40.0
MS + SFT-d	83.8	90.4	94.6	73.1	45.3	66.1	76.8	85.2	70.0	**41.6**
MS + SFT	**84.5**	90.6	**94.6**	**73.2**	**45.8**	**66.8**	**77.5**	**85.8**	**70.3**	40.4

Loss (RLL) [28] and Multi-Similarity Loss (MS) [29]. Most of the comparison is made on GoogleNet [23] because almost all of the chosen competitors report their results on this backbone. For comparison with the SOTA, we also make some comparison on GoogleNet-V2. For fair comparison, we implement all of these methods and report the results from our experiments.

For FTL [31], the features are normed in our implementation as we found that feature normalization will outperform the original method greatly. The FTL differs from the degenerated form of SFT in that it requires a pre-training of the network and is applied in the fine-tuning stage while this is not needed in both SFT and the degenerated form. Also, FTL requires a decoder network and only transfers a part of the energy of $\boldsymbol{\sigma}$ using PCA. In our implementation, we follows them to use 95%.

4.1 Quantitative Results

Table 1, Table 3, Table 2 and Table 4 present the experimental results of SFT on three popular deep metric learning benchmarks and three face recognition benchmarks respectively.

By comparing with baseline methods, it is noticed that SFT can significantly improve the performance of them, especially on Cars196 and CUB-200-2011. For example, when coupled with NPair loss, SFT improves the baseline by 7 point on Cars196. SFT can also boost performance on higher baselines that reported by two most recent losses, Multi-Similarity loss and Ranked-List loss. While SFT is relatively less effective on SOP(Table 3). The reason is that the number of samples for each class in SOP is too small(about 5).

Table 3. Experimental results on Stanford Online Products(SOP). SFT is less effective on SOP as the number of samples for each class is only 5.

	SOP				
	R@1	R@10	R@100	NMI	F1
Triplet (ours)	70.8	85.5	93.8	88.2	28.0
Triplet + SFT-d	71.9	86.4	94.4	88.5	29.3
Triplet + SFT	72.3	86.5	94.5	88.6	29.9
RLL [28] (ours)	77.5	89.9	95.8	89.7	35.3
RLL + SFT-d	77.9	90.3	96.1	89.8	35.9
RLL + SFT	77.8	90.2	96.0	89.9	36.4
MS [29] (ours)	73.1	87.2	94.7	88.5	29.6
MS + SFT-d	73.5	87.5	94.9	88.6	29.8
MS + SFT	73.4	87.1	94.7	88.8	30.9

Table 4. Comparison with the state-of-art methods on IJB-C. The '-' denotes the corresponding results are not reported in the original paper.

Method	Training Data	IJB-C(TAR@FAR)		
		0.001%	0.01%	0.1%
Vggface2 [2]	3.3M	74.7	84.0	91.0
L2-Face [18]	3.3M	78.54	87.01	92.10
Arcface [4]	5.8M	-	92.10	-
L2-Face [18](ours)	3M	79.3	87.3	93.3
L2-Face(ours) + SFT-d	3M	79.4	87.9	93.3
L2-Face(ours) + SFT	3M	**80.6**	**88.2**	**93.6**
CosFace [27] (ours)	3M	85.67	92.11	95.4
CosFace (ours) + SFT-d	3M	86.85	**92.78**	**95.72**
CosFace (ours) + SFT	3M	**87.19**	92.63	95.6

To sum up, SFT performs better than HDML [34], FTL [31] and DVML [12]. For example, when coupled with NPair loss, SFT outperforms the HDML by 1.0 on Cars196.

On higher baselines, such as Multi-Similarity Loss, our SFT outperforms DVML by 0.7 on CUB. The degenerated form of SFT can be effective on most baseline methods. While averagely, it surpass the performance of SFT. Besides the metric learning losses, SFT can also be used together with softmax-based losses. This is mainly used in face recognition tasks. On LFW dataset and YTF dataset(Shown in Table 2), the performance of deep neural networks are nearly saturated, but we still report the performance for comparison with the other works. On IJB-C(Shown in Table 4), we provide a competitive baseline for both L2-Face [18] and CosFace [27], while it is observed that SFT can still boost the performance when compared with the baselines.

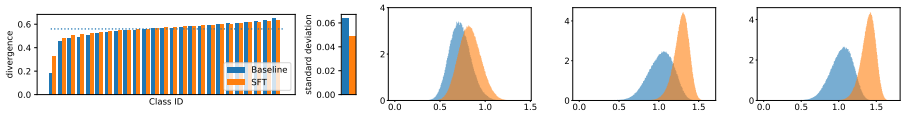
Table 2. Face verification (%) on the LFW and YTF datasets.

Method	Training Data	LFW	YTF
DeepFace+ [24]	4M	97.35	91.4
FaceNet [19]	200M	99.63	95.1
DeepID2+ [22]	300K	99.47	93.2
SphereFace [14]	0.5M	99.42	95.0
CosFace [27]	5M	99.73	97.6
ArcFace [4]	5.8M	99.83	98.02
L2-Face [18]	3.7M	99.78	96.08
L2-Face [18] (ours)	3M	99.45	96.0
L2-Face(ours) + SFT-d	3M	99.41	95.9
L2-Face(ours) + SFT	3M	99.50	96.5
CosFace [27] (ours)	3M	99.68	96.2
CosFace (ours) + SFT-d	3M	99.70	96.5
CosFace (ours) + SFT	3M	99.73	97.2

4.2 Ablation Study

In this part, we conduct the ablation study on Cars-196 with the ranked list loss. The conclusions from these experiments are also applicable to other datasets and loss functions.

Effect on Feature Distributions We find that SFT can make feature distributions more similar than the baseline method. This is consistent with our prior that feature distributions should be similar to each other. In other words, the SFT can make the eigenvalues of variances from different classes to be closer.



(a) (b)

Fig. 4. Effect of SFT on feature distributions. (a) **Left**: Divergences of each class in baseline and SFT. The blue dashed line represents the average divergence of baseline. **Right**: The standard deviation of divergences in baseline and SFT. (b) Histograms of positive(blue) and negative(orange) distance distributions on the Cars196 test set for(from left to right), initial state with pre-trained model, training with ranked list loss, training with ranked list loss together with SFT.

In specific, we compare the similarity by comparing the trace of the scattering matrix of each class. We refer to the trace of the matrix as the divergence. The scattering matrix of each class is defined as:

$$\mathbf{S}_{w,k} = \frac{1}{N_k} \sum_{k=0}^{N_k} (\mathbf{x}_k - \boldsymbol{\mu}_i)^T (\mathbf{x}_k - \boldsymbol{\mu}_i) . \tag{11}$$

Fig. 4(a) shows the divergences of each class, and the standard deviation of the divergences. The class IDs are sorted according to the divergences of the baseline. For clarity, one for every four values is chosen to shown in the histogram. It is observed that the divergences among classes are more balanced when SFT is applied. In general, the divergences below the average(blue dashed line) are increased and those above the average are decreased. The right part of Fig. 4 displays the standard deviations of the divergence values. It is consistent with the conclusion.

Furthermore, the distributions of pair distance are compared. It is shown in Figure 4(b). It is observed that the overlap between the positive parts and negative parts is reduced when SFT is applied. This indicates that SFT helps the network to learn a more discriminative feature space.

Effect on Unbalanced Datasets The face recognition datasets differ from DML datasets in that they are usually long-tailed. Among then, plenty of classes are in short of samples. These classes are usually called the tail classes. Experimentally, we find SFT can improve the performance of tail classes. In specific, we select all classes in MS-Celeb-1M that contains more than 100 samples to construct a mini-dataset. In total, we get 2,445 classes. Then, we random choose 1,500 classes to be the head classes and choose 50 samples each for training. For the remaining 945 classes, we treat them as the tail classes and choose 5 samples each for training. All the other samples are left for

Table 5. Effect of SFT on an unbalanced dataset. Head represents classes with rich samples. Tail represents classes in short of training samples. The results are the classification accuracy(%).

	Baseline	Balanced Train	Unbalanced Train
Head	95.61	**95.77**	95.49
Tail	73.91	78.35	**82.32**

testing. As the training set is much smaller than that of MS-Celeb-1M, we adopt a smaller network for training. In specific, we use a similar CNN architecture for training except that we change the number of residual units to $[1, 1, 1, 1]$. The results are shown in Table 5. We can see that the baseline method performs worst in the tail classes. But when SFT is applied, the performance of the tail classes is increased by a large margin. The best performance in tail classes is achieved by the unbalanced train scheme, which outperforms the baseline by 8.4%, while the performance drop in the head classes is negligible. In summary, the SFT can effectively improve the accuracy of the tail classes.

Impact of Center Estimation As the rotation matrix of SFT is estimated based on feature centers, the center estimation is essential. To evaluate the importance, we compare the image retrieval performance under three circumstances: (1) "Random", skip the center estimation step in line 6 of Algorithm 1. (2)"Pick", randomly pick one sample from the same class as center. (3)SFT, the standard SFT procedure. The results are shown in Table 6. The performances of SFT are almost the same as the baseline when Random or Pick is adopted. While only when SFT is adopted will the performance be improved by a noticeable number. This illustrate that the accurate estimation of class centers is crucial for feature transform. Moreover, it is noticed that even when the center estimation is not accurate, the feature transform will not harm the training too much. This suggests that training with feature transform is stable.

Table 6. Impact of center estimation.

	baseline	SFT	Random	Pick
R@1	74.2	**80.2**	74.4	74.6

Batch Size. The batch size is usually important in deep metric learning as it determines the number of positive pairs and negative pairs used for constructing target loss. While when implemented with our method, the number of positive pairs and negative pairs are enlarged. We then conduct experiments on different batch sizes to "fairly" compare the performance under an equal number of positive and negative pairs. The comparison results are shown in the left part of Fig. 5. It is observed that SFT can beat the baseline with the largest batch size 240 even evaluated under a small batch size 30. This suggests that the improvement when SFT is applied is not due to the increase of batch size.

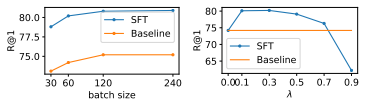


Fig. 5. Performance with different **Left**: batch size; **Right** λ.

Effect of λ. We conduct experiments to explore the influence of the weight factor. As shown in the Fig. 5, when increasing the λ, the performance of the method first increases and then decreases. When the λ is too large, the performance drops significantly. We blame the performance drop to that the gradients from the generated features will dominate the optimization process and infect the optimization of the regular ones. In practice, the optimal λ is data-dependent. We

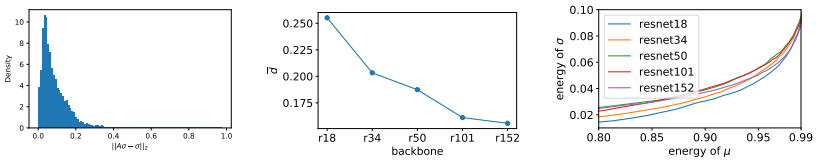


Fig. 6. Experiment on the condition of the degeneration. (a) The distribution of $\|\mathbf{A}\boldsymbol{\sigma} - \boldsymbol{\sigma}\|_\mathbf{2}$ sampled from features of backbone ResNet50. (b) The mean values of the distributions of $\|\mathbf{A}\boldsymbol{\sigma} - \boldsymbol{\sigma}\|_\mathbf{2}$ from different backbones. (c) s_r defined in Eq. 8 with respect to the energy of $\boldsymbol{\mu}$.

do not investigate into what is the optimal value of λ. In most of our experiments, the value is set to 0.2.

Discussion of Degeneration. In Sec 3.4, it is hypothesized that $\boldsymbol{\sigma}$ defined in Eq. 1 is likely to lie in the invariant subspace of $\mathbf{A}$. To investigate into whether the hypothesis holds, we evaluate the value of $d = \|\mathbf{A}\boldsymbol{\sigma} - \boldsymbol{\sigma}\|_\mathbf{2}$ on five versions of ResNet. The distribution of d on ResNet50 is shown in Fig. 6(a). It is noticed that a large number of d values are near zero. For these features, the augmented features by SFT and the degenerated form are close. For each backbone, d is evaluated 10000 times and the mean value is reported. The result is shown in Fig. 6(b). As observed, the hypothesis is more likely to hold in deeper networks. This implies that the degeneration of SFT will be more likely to happen when the network gets deeper.

Our experiment also reveals that the learned subspaces for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ tend to be orthogonal. This is presented in Fig. 6(c). For example, on the backbone of ResNet50, $\boldsymbol{\sigma}$ only distribute 10% energy on the subspace that covers 99% energy of $\boldsymbol{\mu}$. It means that, although the ideal condition in Proposition 3 can not be reached, the learned feature space tend to approach it. These experimental results support our analysis that the degeneration of SFT happens for most features. Considering the comparison shows that the SFT will outperform the degenerated form in most scenarios, the side effect of the degenerated form on features that won't degenerate should not be neglected.

5 Conclusion

In this paper, we propose Spherical Feature Transform (SFT) to generate new features from existing ones. The proposed SFT can effectively enrich the intra-class variances of both regular classes and under-represented ones. We have demonstrated the effectiveness of SFT by applying it to several most recent DML frameworks in three popular deep metric learning benchmark datasets and three face recognition benchmark datasets.

Acknowledgment. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700800.

References

1. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
2. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). pp. 67–74. IEEE (2018)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2019)
5. Dixit, M., Kwitt, R., Niethammer, M., Vasconcelos, N.: Aga: Attribute-guided augmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7455–7463 (2017)
6. Duan, Y., Zheng, W., Lin, X., Lu, J., Zhou, J.: Deep adversarial metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2780–2789 (2018)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
8. Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: European Conference on Computer Vision. pp. 87–102. Springer (2016)
9. Hamsici, O.C., Martinez, A.M.: Spherical-homoscedastic distributions: The equivalency of spherical and normal distributions in classification. Journal of Machine Learning Research **8**(Jul), 1583–1623 (2007)
10. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database forstudying face recognition in unconstrained environments (2008)
11. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 554–561 (2013)
12. Lin, X., Duan, Y., Dong, Q., Lu, J., Zhou, J.: Deep variational metric learning. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 689–704 (2018)
13. Liu, B., Wang, X., Dixit, M., Kwitt, R., Vasconcelos, N.: Feature space transfer for data augmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9090–9098 (2018)
14. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 212–220 (2017)
15. Maze, B., Adams, J., Duncan, J.A., Kalka, N., Miller, T., Otto, C., Jain, A.K., Niggel, W.T., Anderson, J., Cheney, J., et al.: Iarpa janus benchmark-c: Face dataset and protocol. In: 2018 International Conference on Biometrics (ICB). pp. 158–165. IEEE (2018)
16. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4004–4012 (2016)
17. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)

18. Ranjan, R., Castillo, C.D., Chellappa, R.: L2-constrained softmax loss for discriminative face verification. arXiv preprint arXiv:1703.09507 (2017)
19. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
20. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Advances in Neural Information Processing Systems. pp. 1857–1865 (2016)
21. Sohn, K., Liu, S., Zhong, G., Yu, X., Yang, M.H., Chandraker, M.: Unsupervised domain adaptation for face recognition in unlabeled videos. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3210–3218 (2017)
22. Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2892–2900 (2015)
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015), http://arxiv.org/abs/1409.4842
24. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1701–1708 (2014)
25. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
26. Wang, F., Xiang, X., Cheng, J., Yuille, A.L.: Normface: l 2 hypersphere embedding for face verification. In: Proceedings of the 25th ACM international conference on Multimedia. pp. 1041–1049. ACM (2017)
27. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5265–5274 (2018)
28. Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R., Robertson, N.M.: Ranked list loss for deep metric learning. arXiv preprint arXiv:1903.03238 (2019)
29. Wang, X., Han, X., Huang, W., Dong, D., Scott, M.R.: Multi-similarity loss with general pair weighting for deep metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5022–5030 (2019)
30. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity. IEEE (2011)
31. Yin, X., Yu, X., Sohn, K., Liu, X., Chandraker, M.: Feature transfer learning for face recognition with under-represented data. In: In Proceeding of IEEE Computer Vision and Pattern Recognition. Long Beach, CA (June 2019)
32. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters **23**(10), 1499–1503 (2016)
33. Zhao, Y., Jin, Z., Qi, G.j., Lu, H., Hua, X.s.: An adversarial approach to hard triplet generation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 501–517 (2018)
34. Zheng, W., Chen, Z., Lu, J., Zhou, J.: Hardness-aware deep metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 72–81 (2019)