Correcting the Triplet Selection Bias for Triplet Loss

Baosheng Yu[1], Tongliang Liu[1], Mingming Gong[2,3], Changxing Ding[4], and Dacheng Tao[1]

[1] UBTECH Sydney AI Centre and SIT, FEIT, The University of Sydney
[2] Department of Biomedical Informatics, University of Pittsburgh
[3] Department of Philosophy, Carnegie Mellon University
[4] School of Electronic and Information Engineering, South China University of Technology
bayu0826@uni.sydney.edu.au, tongliang.liu@sydney.edu.au, mig73@pitt.edu.com, chxding@scut.edu.cn, dacheng.tao@sydney.edu.au

Abstract. Triplet loss, popular for metric learning, has made a great success in many computer vision tasks, such as fine-grained image classification, image retrieval, and face recognition. Considering that the number of triplets grows cubically with the size of training data, triplet selection is thus indispensable for efficiently training with triplet loss. However, in practice, the training is usually very sensitive to the selection of triplets, e.g., it almost does not converge with randomly selected triplets and selecting the hardest triplets also leads to bad local minima. We argue that the bias in the selection of triplets degrades the performance of learning with triplet loss. In this paper, we propose a new variant of triplet loss, which tries to reduce the bias in triplet selection by adaptively correcting the distribution shift on the selected triplets. We refer to this new triplet loss as adapted triplet loss. We conduct a number of experiments on MNIST and Fashion-MNIST for image classification, and on CARS196, CUB200-2011, and Stanford Online Products for image retrieval. The experimental results demonstrate the effectiveness of the proposed method.

Keywords: Triplet Loss · Selection Bias · Domain Adaptation

1 Introduction

Deep metric learning aims to learn a similarity or distance metric which enjoys a small intra-class variation and a large inter-class variation [42]. Triplet loss is a popular loss function for deep metric learning and has made a great success in many computer vision tasks, such as fine-grained image classification [39], image retrieval [17, 22], person re-identification [6, 14], and face recognition [34, 31]. Recently, deep metric learning approaches employing triplet loss have attracted a lot of attention due to their efficiency for dealing with enormous of labels, e.g., the extreme multi-label classification problem [32]. More specifically, for conventional classification approaches, the number of parameters will increase linearly with

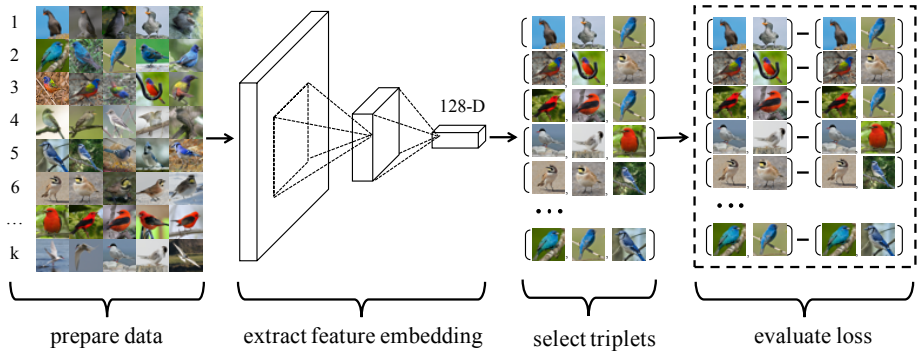


Fig. 1: The pipeline of triplet loss based deep metric learning. In the first stage, a mini-batch is sampled from the training data, which usually contains k identities with several images per identity. Deep neural networks then are used to learn a feature embedding, e.g., a 128-D feature vector. In the third stage, a subset of triplets are selected using some triplet selection methods. Lastly, the loss is evaluated using the selected triplets.

the number of labels, and it is impractical to learn an N-way softmax classifier with millions of labels [29]. However, with triplet loss, deep metric learning is able to efficiently deal with an extreme multi-label classification problem by learning a compact embedding, which is known as the large margin nearest neighbor (LMNN) classification [42]. As a result, deep metric learning exploiting triplet loss is very efficient for applications with enormous labels, e.g., the number of objects in image retrieval [17], the number of identities in face recognition [34] and person re-identification [14].

To learn a discriminative feature embedding, triplet loss maximizes the margin between the intra-class distance and the inter-class distance. As a result, for each triplet (x^a, x^p, x^n), where x^a is called the anchor point, x^p is called the positive point having the same label with x^a, and x^n is called the negative point having a different label, the intra-class distance $d(x^a, x^p)$ will be smaller than the inter-class distance $d(x^a, x^n)$ in the learned embedding space. As the number of triplets grows cubically with the size of training data, triplet selection thus is indispensable for efficiently training with triplet loss. Specifically, triplet selection usually works in an online manner, i.e., triplets are constructed within each mini-batch [34], and we describe a typical pipeline of deep metric learning using triplet loss in Fig. 1.

However, the performance of triplet loss is heavily influenced by triplet selection methods [6, 14], i.e., training with randomly selected triplets almost does not converge while training with the hardest triplets often leads to a bad local solution [34]. To ensure fast convergence, it is crucial to select "good" hard triplets [34] and a variety of triplet selection methods have been designed in different applications [39, 17, 34, 14]. Although selecting hard triplets leads to fast

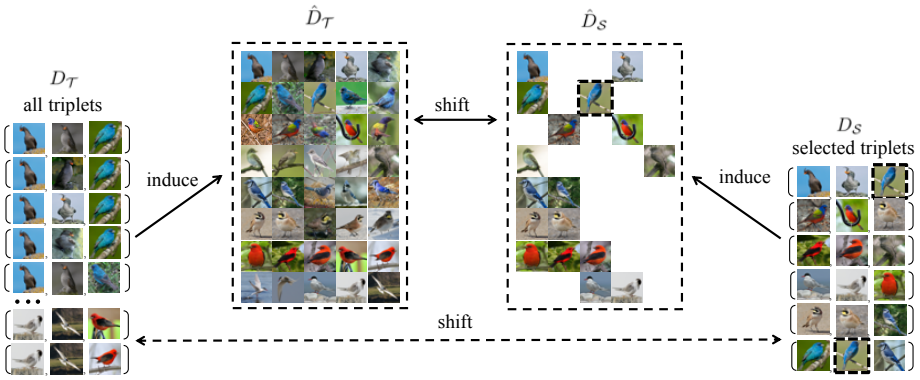


Fig. 2: An example illustrating the distribution shift in triplet selection. In online triplet selection, all triplets $D_{\mathcal{T}}$ are constructed from each mini-batch and will induce a dataset $\hat{D}_{\mathcal{T}}$. For the selected triplets $D_{\mathcal{S}}$, they also induce a dataset $\hat{D}_{\mathcal{S}}$. We evaluate the distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ using the distribution shift between $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$.

convergence, it has the risk of introducing a selection bias, which is an essential problem for learning. A triplet selection method thus needs to balance the trade-off between mining hard triplets and introducing selection bias. In contrast to struggling with this trade-off by carefully selecting triplets, we address this problem by directly minimizing the selection bias. More specifically, let $D_{\mathcal{T}}$ denote all possible triplets and $D_{\mathcal{S}}$ indicate the subset of selected triplets from $D_{\mathcal{T}}$. If the triplet selection is unbiased, $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ then share the same distribution. Otherwise, we can correct the bias in triplet selection by minimizing the distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$.

The problem of distribution shift falls within the scope of domain adaption [3, 16], which arises when learning a predictor from the source domain $\mathcal{S}$ while the target domain $\mathcal{T}$ changes. In learning with triplet loss, the model is trained using selected triplets $D_{\mathcal{S}}$ while the target is to learn a model using all possible triplets $D_{\mathcal{T}}$. To measure the distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$, we define a set of triplet-induced data, i.e., given a set of triplets, e.g., $D_{\mathcal{S}}$, the triplet-induced data $\hat{D}_{\mathcal{S}}$ is defined as follows:

$$\hat{D}_{\mathcal{S}} = \{(x_i^a, y_i^a), (x_i^p, y_i^p), (x_i^n, y_i^n) |\ \forall (x_i^a, x_i^p, x_i^n) \in D_{\mathcal{S}}\}, \tag{1}$$

where y_i are the corresponding labels of x_i. The induced data $\hat{D}_{\mathcal{T}}$ can be defined similarly. We give an example of the distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ in Fig. 2. To deal with the problem of distribution shift, distribution matching approaches learn a domain invariant representation and have been widely employed [3, 2, 30]. Due to triplet loss often involves lots of labels and inspired by the methods in [48, 11], we try to minimize the distribution shift between $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$ by learning a conditional invariant representation $\Phi(X)$, i.e., $P^{\mathcal{S}}(\Phi(X)|Y) \approx P^{\mathcal{T}}(\Phi(X)|Y)$, where X and Y stand the random variables for

data and label, respectively. More specifically, we propose a distribution matching loss function by employing Maximum Mean Discrepancy (MMD) [16], which measures the difference between $P^{\mathcal{S}}(\Phi(X)|Y)$ and $P^{\mathcal{T}}(\Phi(X)|Y)$. As a result, we learn a discriminative and conditional invariant embedding by jointly training with the triplet loss and the distribution matching loss.

In this paper, we first introduce the problem of triplet selection bias for learning with triplet loss. We then address this problem by reducing distribution shift between the triplet-induced data $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$. As the proposed distribution matching loss adaptively corrects the distribution shift, we refer to this new variant of triplet loss as adapted triplet loss. Lastly, we conduct a number of experiments on MNIST [23] and Fashion-MNIST [45] for image classification, on CARS196 [20], CUB200-2011 [38], and Stanford Online Products [29] for image retrieval. The experimental results demonstrate the effectiveness of the proposed method.

2 Related Work

Deep Metric Learning and Triplet Loss. Many problems in machine learning and computer vision depend heavily on learning a distance metric [42]. Inspired by the great success of deep learning [21], deep neural networks have been widely used to learn a discriminative feature embedding [39, 15]. Deep metric learning employing triplet loss raises a lot of attention due to its impressive performance on FaceNet [34] for face verification and recognition. After that, triplet loss has been widely used to learn a discriminative embedding for a variety of applications, such as image classification [39] and image retrieval [17, 22, 49, 12, 47]. A majority of applications for triplet loss lies in visual object recognition, such as action recognition [33], vehicle recognition [26], place recognition [1], 3d pose recognition [43], face recognition [34, 31, 9], and person re-identification [10, 46, 6, 25, 4, 14].

Triplet Selection Methods. Triplet selection is the key for the success of triplet loss and a variety of triplet selection methods have been used in different applications [39, 15, 34, 31, 40, 7]. More specifically, in the deep ranking model proposed by [39], triplets are selected according to the pair-wise relevance score. In [40], the triplets are selected using the top k triplets in each mini-batch based on the margin $d(x^a, x^p) - d(x^a, x^n)$. In [15], it selects only hard triplets, i.e., $d(x^a, x^p) < d(x^a, x^n)$, while [34, 31] select semi-hard triplets which violate the triplet constraint, i.e., $d(x^a, x^p) + \alpha < d(x^a, x^n)$, where α is a positive scalar. Unlike [34], which defines semi-hard triplet using moderate negatives, [35] select semi-hard triplets based on moderate positives. [7] proposes an online hard negative mining method for triplet selection to boost the performance on triplet loss. In [14], it proposes a batch-hard triplet selection method, i.e., it first select a set of hard anchor-positive pairs, and it then select hardest negatives within the mini-batch. Recently, [44] proposes a weighted sampling method to address the sampling matters in deep metric learning.

Domain Adaptation. Domain adaptation methods can be divided into four categories due to different assumptions about how the distribution shifts across domains. (1) Covariate shift [16] assumes the marginal distribution $P(X)$ changes across domains while the conditional distribution $P(Y|X)$ stays the same. (2) Model shift [41] assumes that both $P(X)$ and $P(Y|X)$ independently change across domains. (3) Target shift [48] assumes that the marginal distribution $P(Y)$ shifts wile $P(X|Y)$ stays the same. (4) Generalized target shift [11, 27, 24] assumes that both $P(Y)$ and $P(X|Y)$ independently change. Since triplet loss is widely used for extreme multi-label classification problems, we model the triplet selection bias by the change of $P(X|Y)$ in this paper.

3 Formulation

In this section, we first introduce triplet loss for deep metric learning and a widely used triplet selection method, i.e., semi-hard triplets [34]. We then formulate the problem of triplet selection bias as the distribution shift problem on triplet induced data. To minimize the distribution shift, we propose a distribution matching loss, which jointly works with the triplet loss to adaptively correct the distribution shift. As a result, we refer to this new triplet loss as adapted triplet loss.

3.1 Triplet Loss for Deep Metric Learning

Let X, Y denote two random variables, which indicate data and label, respectively. Let D denote a set of training data sampled from $P(X, Y)$, i.e., $D = \{(x_i, y_i)|\ (x_i, y_i) \sim P(X, Y)\}$. Metric learning aims to learn a distance function that assigns small (or large) distance to a pair of similar (or dissimilar) examples. A widely used distance metric, i.e., the Mahalanobis distance, is defined as follows:

$$d_K^2(x_i, x_j) = (x_i - x_j)^\top K(x_i - x_j), \tag{2}$$

where K is a symmetric positive semi-definite matrix. As K can be decomposed as $K = L^\top L$, we then have

$$d_K^2(x_i, x_j) = \|L(x_i - x_j)\|_2^2 = \|x_i' - x_j'\|_2^2, \tag{3}$$

where $x_i' = Lx_i$ and $x_j' = Lx_j$. Inspired by this, deep metric learning uses deep neural networks to learn a feature embedding $x' = \Phi(x)$, which generalizes the linear transformation $x' = Lx$ to a non-linear transformation $\Phi(x)$. That is, the learned distance metric is

$$d_K^2(x_i, x_j) = ||\Phi(x_i) - \Phi(x_j)||_2^2. \tag{4}$$

To learn a discriminative feature embedding $\Phi(x)$, i.e., intra-class distance is smaller than inter-class distance [42], triplet loss is defined as follows:

$$\mathcal{L}_{triplet}^* = \sum_{(x^a, x^p, x^n) \in D_\mathcal{T}} [d_K^2(x^a, x^p) - d_K^2(x^a, x^n) + \alpha]_+, \tag{5}$$

where $[\cdot]_+ = \max(0, \cdot)$, $\alpha \geq 0$ is the margin, and $D_{\mathcal{T}}$ is a set of triplets constructed from the original training data D, i.e.,

$$D_{\mathcal{T}} = \{(x^a, x^p, x^n) |\ y^a = y^p \text{ and } y^a \neq y^n\}. \tag{6}$$

3.2 Triplet Selection Bias

Let $D_{\mathcal{T}}$ denote all possible triplets constructed within a mini-batch and $D_{\mathcal{S}}$ denote the subset of selected triplets, i.e., $D_{\mathcal{S}} \subseteq D_{\mathcal{T}}$. More specifically, given a mini-batch of training data with k identities and c images per identity, there will be $k(k-1)c^2(c-1)$ possible triplets in total. As the number of triplets grows cubically with the number of training data, triplet loss usually is evaluated using only a selected subset of the total triplets. A typical triplet selection methods used in [34], which is referred to as semi-hard triplet selection, can be described as follows: it uses all possible anchor-positive pairs, i.e., $kc(c-1)$ pairs in total. For each anchor-positive pair (x^a, x^p), a semi-hard negative x^n then is randomly selected from all negatives under the constraint

$$d_K^2(x^a, x^p) \leq d_K^2(x^a, x^n) < d_K^2(x^a, x^p) + \alpha. \tag{7}$$

That is, triplet loss is evaluated on $D_{\mathcal{S}}$, i.e.,

$$\mathcal{L}_{triplet} = \sum_{(x^a, x^p, x^n) \in D_{\mathcal{S}}} [d_K^2(x^a, x^p) - d_K^2(x^a, x^n) + \alpha]_+, \tag{8}$$

As a result, there will be always a distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$. To measure the distribution shift between $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$, we define the triplet-induced data $\hat{D}_{\mathcal{S}}$ for $D_{\mathcal{S}}$ as follows:

$$\hat{D}_{\mathcal{S}} = \{(x^a, y^a), (x^p, y^p), (x^n, y^n) |\ \forall (x^a, x^p, x^n) \in D_{\mathcal{S}}\}. \tag{9}$$

Similarly, we also define $\hat{D}_{\mathcal{T}}$ as the data induced by $D_{\mathcal{T}}$. If $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ shares the same distribution $P(x^a, x^p, x^n)$, we then have, $\forall x \in \hat{D}_{\mathcal{S}}$,

$$P^{\mathcal{S}}(x) = \sum_{i \in \{a,p,n\}} P(x^i) * \mathbf{1}\{x = x^i\} = P^{\mathcal{T}}(x). \tag{10}$$

where $P^{\mathcal{S}}(x)$ and $P^{\mathcal{T}}(x)$ are the probability density functions for $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$ respectively. That is, there will be no distribution shift between the triplet-induced data $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$. As a result, we use the difference between $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$ as a measure the distribution shift in triplet loss.

3.3 Adapted Triplet Loss

To correct the triplet selection bias, we thus try to minimize the distribution shift between $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$ during learning the representation. Let $x \in X$ denote an input data and $\Phi(x)$ denote the representation learned by deep neural networks,

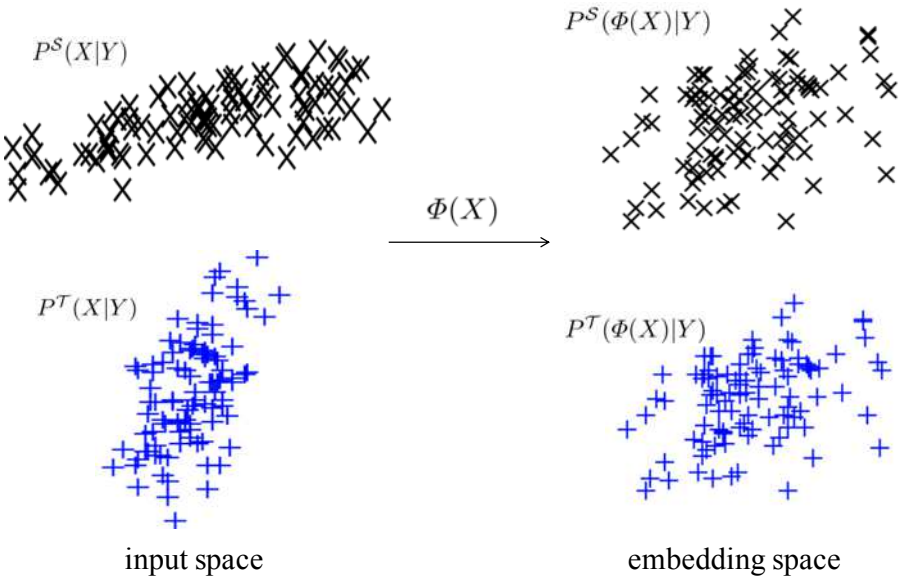


Fig. 3: An example illustrating the conditional invariant representation. There is a distribution shift between the source domain and the target domain in the input space, i.e., $P^{\mathcal{S}}(X|Y) \neq P^{\mathcal{T}}(X|Y)$. By learning a conditional invariant representation $\Phi(x)$, both source domain and target domain shares similar distribution in the embedding space, i.e., $P^{\mathcal{S}}(\Phi(X)|Y) = P^{\mathcal{T}}(\Phi(X)|Y)$. That is, the embedding $\Phi(x)$ generalizes well on the target domain while it is learned on source domain. Intuitively, the source domain consists of the selected triplets $D_{\mathcal{S}}$ while the target domain consists of all triplets $D_{\mathcal{T}}$. That is, we learn a conditional invariant embedding using selected triplets and it will generalize well on all triplets.

i.e., a dimension fixed feature vector. Inspired by [48, 11], we learn a shared conditional invariant representation between $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$, i.e.,

$$P^{\mathcal{S}}(\Phi(X)|Y) = P^{\mathcal{T}}(\Phi(X)|Y). \tag{11}$$

See Fig. 3 for an example of the conditional invariant representation. Maximum Mean Discrepancy (MMD) has been widely used to estimate the difference between two distributions [16] and we thus use the conditional mean feature embedding to estimate the difference between $P^{\mathcal{S}}(\Phi(X)|Y)$ and $P^{\mathcal{T}}(\Phi(X)|Y)$. As a result, the distribution matching loss can be defined as follows:

$$\mathcal{L}_{match} = \sum_{y} \|\Phi_y^{\mathcal{S}} - \Phi_y^{\mathcal{T}}\|_2^2, \tag{12}$$

where $\Phi_y^{\mathcal{S}}$ and $\Phi_y^{\mathcal{T}}$ are class-specific mean feature embeddings on $\hat{D}_{\mathcal{S}}$ and $\hat{D}_{\mathcal{T}}$ respectively, i.e.,

$$\Phi_y^{\mathcal{S}} = \sum_{(X,Y=y)\in\hat{D}_{\mathcal{S}}} P^{\mathcal{S}}(\Phi(X)|Y) * \Phi(X) \tag{13}$$

and

$$\Phi_y^{\mathcal{T}} = \sum_{(X,Y=y)\in\hat{D}_{\mathcal{T}}} P^{\mathcal{T}}(\Phi(X)|Y) * \Phi(X). \tag{14}$$

To correct the distribution shift in learning with triplet loss, we thus learn a discriminative and conditional invariant feature embedding by jointly minimizing the triplet loss as well as the distribution matching loss, i.e.,

$$\mathcal{L} = \mathcal{L}_{triplet} + \lambda * \mathcal{L}_{match}, \tag{15}$$

where λ is a trade-off parameter. Considering that this new variant of triplet loss adaptively corrects the triplet selection bias, we refer to it as adapted triplet loss.

3.4 Semi-supervised Adapted Triplet Loss

Unlabeled data are usually very helpful for domain adaptation. We believe that the unlabeled data will also be helpful for correcting the triplet selection bias. To scale the adapted triplet loss for exploiting large scale unlabeled data, we extend it for the semi-supervised setting.

Given a set of labeled data D_1 and a set of unlabeled data D_2. Let $D_{\mathcal{T}_1}$ denote the all triplets constructed from D_1 and $D_{\mathcal{S}}$ denote the subset of selected triplets, i.e., $D_{\mathcal{S}} \subseteq D_{\mathcal{T}_1}$. Let $D_{\mathcal{T}_2}$ be the latent triplets constructed using the unlabeled data D_2, which is actually unavailable since we do not know the latent labels of D_2. Different from the supervised setting, in which we learn a conditional invariant representation among $D_{\mathcal{S}}$ and $D_{\mathcal{T}_1}$, we consider how to learn a conditional invariant representation between $D_{\mathcal{S}}$, $D_{\mathcal{T}_1}$, and $D_{\mathcal{T}_2}$, i.e.,

$$P^{\mathcal{S}}(\Phi(X)|Y) = P^{\mathcal{T}_1}(\Phi(X)|Y) = P^{\mathcal{T}_2}(\Phi(X)|Y). \tag{16}$$

Given the target $P^{\mathcal{S}}(\Phi(X)|Y) = P^{\mathcal{T}_2}(\Phi(X)|Y)$, we then have

$$\sum_y P^{\mathcal{T}_2}(\Phi(X)|Y)P^{\mathcal{T}_2}(Y) = \sum_y P^{\mathcal{S}}(\Phi(X)|Y)P^{\mathcal{T}_2}(Y). \tag{17}$$

That is, if we know the class ratio $P^{\mathcal{T}_2}(Y)$ for triplet-induced data $\hat{D}_{\mathcal{T}_2}$, we are able to estimate the difference between $P^{\mathcal{S}}(\Phi(X)|Y)$ and $P^{\mathcal{T}_2}(\Phi(X)|Y)$. Inspired by [18], we estimate the class ratio $P^{\mathcal{T}_2}(Y)$ by converting it into an optimization problem, i.e.,

$$\theta^{\mathcal{T}_2} = \arg\min_{\theta} \| \sum_y \theta_y^{\mathcal{T}_2} * \Phi_y^{\mathcal{S}} - \Phi^{\mathcal{T}_2} \|_2^2, \ s.t. \sum_y \theta_y = 1, \tag{18}$$

where $\Phi^{\mathcal{T}_2} = \mathbb{E}_{P_X^{\mathcal{T}_2}}[\Phi(X)]$ and $\theta_y^{\mathcal{T}_2} = P^{\mathcal{T}_2}(Y = y)$. This optimization problem can be solved as follows:

$$\theta_{1:|Y|-1}^{\mathcal{T}_2} = (A^\top A)^{-1} A^\top B, \text{ and } \theta_0^{\mathcal{T}_2} = 1 - \sum_{y=1}^{|Y|-1} \theta_y^{\mathcal{T}_2}, \tag{19}$$

where $A = [\Phi_1^{\mathcal{S}} - \Phi_0^{\mathcal{S}}, \ldots, \Phi_{|Y|-1}^{\mathcal{S}} - \Phi_0^{\mathcal{S}}]$ and $B = \Phi^{\mathcal{T}_2}$. We can then define the distribution matching loss in a semi-supervised manner, i.e.,

$$\mathcal{L}_{semi\text{-}match} = \| \sum_y \theta_y^{\mathcal{T}_2} * \Phi_y^{\mathcal{S}} - \Phi^{\mathcal{T}_2} \|_2^2. \tag{20}$$

4 Experiment

In this section, we evaluate the proposed adapted triplet loss function on image classification and retrieval. For image classification, we use MNIST [23] and Fashion-MNIST [45] datasets. The MNIST dataset contains 60,000 training examples and 10,000 test examples, in which all examples are 28×28 grayscale images of handwritten digits. The Fashion-MNIST dataset contains a set of 28×28 grayscale article images and shares the same structure with the MNIST dataset, i.e., 60,000 training examples and 10,000 test examples. For image retrieval, we use three popular datasets, CARS196 [20], CUB200-2011 [38], and Stanford Online Products [29]. The CARS196 dataset contains 16,185 images of 196 different car models, the CUB200-2011 dataset contains 11,788 images of 200 different species of birds, and the Stanford Online Products contains 120,053 images of 22,634 different products.

4.1 Implementation Details

We implement the proposed method using Caffe [19]. Following [34], we always use a L2 normalization layer before the triplet loss layer. We use the margin $\alpha = 0.2$ in all experiments. We train our models using the stochastic gradient descent (SGD) algorithm with momentum 0.9 and weight decay 2e-5. For experiments on MNIST and Fashion-MNIST datasets, we learn 64-D feature embeddings using a modified LeNet [23]. More specifically, we use 3×3 filters in all convolutional layers and replace all activation layers with the PReLU [13] layer. The batch size is set to 256, which is large enough for both MNIST and Fashion-MNIST datasets, i.e., we are able to select enough triplets from each mini-batch. We use the learning rate 0.001 and the maximum iterations are set to 20k and 50k for MNIST and Fashion-MNIST, respectively.

For experiments on CARS196, CUB200-2011, and Stanford Online Products, we use GoogLeNet [37] as our base network and all layers except the last fully connected layer are initialized from the model trained on ImageNet [8]. The last fully connected layer is changed to learn 128-D feature embeddings and is initialized with random weights. All training images are resized to 256×256 and

randomly cropped to 224×224. We use a learning rate 0.0005 with the batch size 120 and the maximum training iterations are set to 15k iterations on CARS196, 20k iterations on CUB200-2011, and 50k iterations on Stanford Online Products datasets. To ensure enough triplets in each mini-batch, we prepare the training data using a similar method with [34], i.e., each mini-batch is randomly sampled from 20 classes with 6 images per class.

4.2 Experiment on Image Classification

In this subsection, we describe the experimental results on MNIST and Fashion-MNIST datasets. To demonstrate the effectiveness of the proposed method, we compare the classification accuracy of models trained using the original triplet loss function (baseline) and the adapted triplet loss function. The evaluation metric can be described as follows: to learn a fixed dimensional feature embedding $\Phi(x)$, we train our models using the original triplet loss function and the adapted triplet loss function respectively.

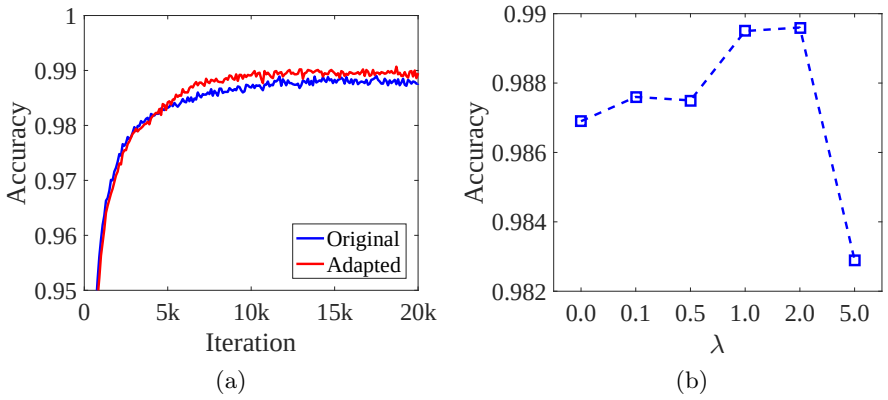


Fig. 4: Results on MNIST dataset. In figure (a), we use $\lambda = 2.0$ for adapted triplet loss and compare its performance with the original triplet loss for every 100 iterations. In figure (b), we compare the test accuracy for using different λ.

For testing, we first evaluate the conditional mean embedding $\mathbb{E}[\Phi(x)|y]$, i.e., the mean point in embedding space, for each class y using the training data. For each input x in test set, we then assign it to a class $\hat{y}$ according to the nearest mean point, i.e.,

$$\hat{y} = \arg\min_{y} \|\Phi(x) - \mathbb{E}[(\Phi(x)|y)]\|_2^2 \tag{21}$$

We demonstrate the results on MNIST dataset in Fig. 4. More specifically, we find that: in figure (a), the adapted triplet loss brings improvement after 5k iterations. Possible explanations for this improvement can be described as follows: for

the original triplet loss, the gradient might be dominated by the noise triplets or the hard triplets from some specific classes while the distribution matching loss can adaptively corrects the triplet selection bias between selected triplets and all possible triplets. That is, the adapted triplet loss will generate more balanced gradients for each iteration. Another reason is that the distribution matching loss acts as a regularizer for the original triplet loss, which reduces the risk of overfitting. We evaluate the performance for the adapted triplet loss using different loss weight λ, i.e., $\lambda = 0,\ 0.1,\ 0.5,\ 1.0,\ 2.0,\ 5.0$ in figure (b). Specifically, we use $\lambda = 0$ for the original triplet loss, which is a special case of the adapted triplet loss. We find that a trade-off on λ are required for using adapted triplet loss to learn a discriminative and conditional invariant embedding. Furthermore, we demonstrate similar results on Fashion-MNIST in Fig. 5.

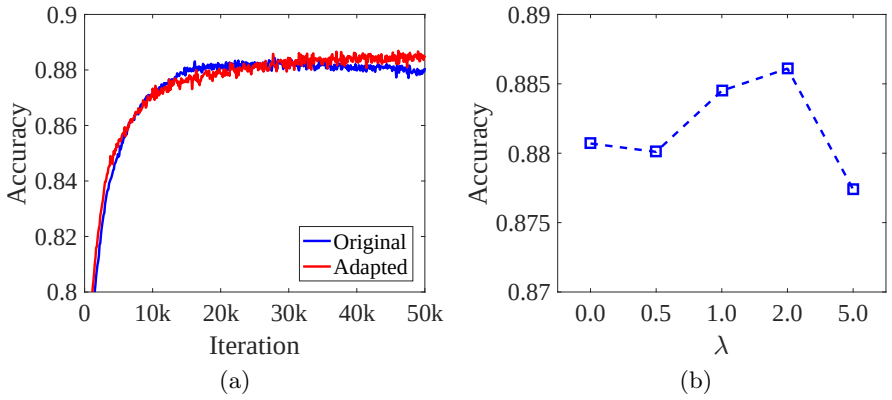
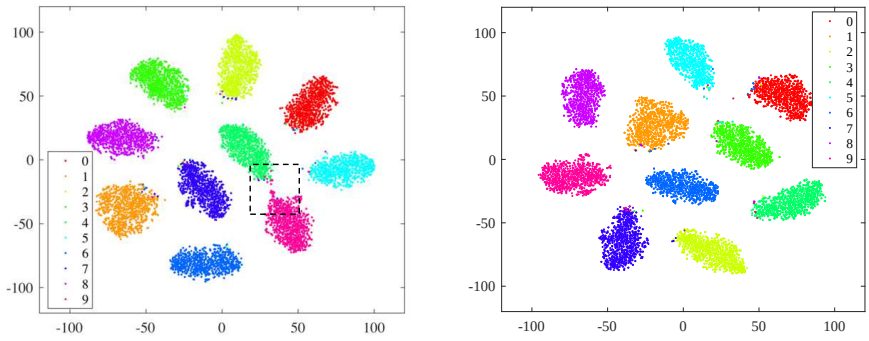


Fig. 5: Results on Fashion-MNIST dataset. In figure (a), we use $\lambda = 2.0$. For the original triplet loss, the test accuracy is reduced after $40k$ iterations, while the adapted triplet loss does not suffer from the problem of overfitting.

To demonstrate the feature embeddings learned by both the original triplet loss and the adapted triplet loss, we use t-SNE [28], which has been widely used for the visualization of high dimensional data, to convert embeddings into 2D space. In Fig. 6, we show the embeddings learned by the adapted triplet loss. Comparing with the embeddings learned by the original triplet loss, we find that the embedding learned by the adapted triplet loss forms uniform margins between different classes, while the embedding learned by the original triplet loss fails to keep a clear margins between some classes.

4.3 Experiment on Image Retrieval

In this subsection, we evaluate the adapted triplet loss on image retrieval. For CARS196, CUB200-2011, and Stanford Online Products datasets, we use similar



(a) Original (b) Adapated

Fig. 6: An example for the visualization of feature embeddings learned by the adapted triplet loss as well as the baseline model, i.e., the original triplet loss. We use the model trained on the training set of MNIST and show the learned embeddings on the test set. For the embedding learned by the original triplet loss, the margin between the two classes in the dash-line rectangle area is not large enough.

train/test splits with [29]. More specifically, for CARS196 dataset, we use all 8054 images from the first 98 classes as the training data and the rest as test data (8131 images); For CUB200-2011 dataset, we use the data from the first 100 classes as the training data (5864 images) and the rest 5924 images for test; For Stanford Online Products dataset, we use the standard train/test split provided in the dataset, i.e., 59,551 images of the first 11,318 classes for training and the rest 60,502 images of 11,316 classes for testing.

For all experiments on image retrieval, we use the standard Recall@K metric, i.e., the same protocol used in [29]. More specifically, the Recall@K metric can be described as follows: given a query image and its K nearest neighbors, if at least one example hit the query image, i.e., with the same label, the recall rate is equal to 1, otherwise the recall rate is 0. We then report the mean recall rate on all query images. For CARS196, CUB200-2011, and Stanford Online Products datasets, we train all models using only the training split and use all test images as the query images to evaluate the recall rate.

We demonstrate the recall rate on CARS196, CUB200-2011, and Stanford Online Products datasets in Table. 1. We can see that the adapted triplet loss outperforms the baseline with all different K values. The maximum improvement usually appears at $K = 1$, which is the most valuable component for image retrieval. Another observation is that the adapted triplet loss usually recalls more positive neighbors. Furthermore, we demonstrate the retrieval results on CARS196 and CUB200-2011 datasets in Fig. 7. More specifically, we select four

Loss Function	R@1	R@2	R@3	R@4	R@5	R@10	R@20
Original ($\lambda = 0$)	0.7781	0.8582	0.8903	0.9105	0.9217	0.9523	0.9716
Adapted ($\lambda = 0.001$)	0.7858	0.8587	0.8921	0.9094	0.9228	0.9525	0.9715
Adapted ($\lambda = 0.005$)	0.7912	**0.8666**	**0.8966**	0.9133	0.9250	0.9535	0.9716
Adapted ($\lambda = 0.010$)	**0.7917**	0.8627	0.8939	**0.9135**	**0.9237**	**0.9570**	**0.9745**
Adapted ($\lambda = 0.050$)	0.7917	0.8627	0.8939	0.9135	0.9237	0.9570	0.9745
Adapted ($\lambda = 0.100$)	0.7631	0.8463	0.8774	0.8996	0.9130	0.9449	0.9665

(a) CARS196

Loss function	R@1	R@2	R@3	R@4	R@5	R@10	R@20
Original ($\lambda = 0$)	0.4450	0.5724	0.6435	0.6913	0.7275	0.8207	0.8893
Adapted ($\lambda = 0.005$)	0.4439	0.5763	0.6464	0.6884	0.7250	0.8253	0.8964
Adapted ($\lambda = 0.010$)	**0.4660**	**0.5861**	**0.6555**	**0.6997**	**0.7343**	**0.8288**	**0.9024**
Adapted ($\lambda = 0.100$)	0.4512	0.5768	0.6475	0.6904	0.7230	0.8160	0.8902
Adapted ($\lambda = 0.500$)	0.4483	0.5682	0.6381	0.6879	0.7245	0.8114	0.8890

(b) CUB200-2011

Loss function	R@1	R@2	R@3	R@4	R@5	R@10	R@100
Original ($\lambda = 0$)	0.6274	0.6865	0.7170	0.7384	0.7524	0.7955	0.9050
Adapted ($\lambda = 0.010$)	**0.6303**	**0.6882**	**0.7206**	**0.7416**	**0.7550**	**0.7982**	**0.9071**
Adapted ($\lambda = 0.050$)	0.6303	0.6876	0.7191	0.7386	0.7530	0.7964	0.9063
Adapted ($\lambda = 0.100$)	0.6297	0.6874	0.7183	0.7378	0.7526	0.7957	0.9044

(c) Stanford Online Products

Table 1: Recall rate on CARS196, CUB200-2011, and Stanford Online Products datasets. For the adapted triplet loss, we train multiple models on all datasets using different λ, i.e., we use $\lambda = 0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1$ on CARS196, $\lambda = 0.005,\ 0.01,\ 0.1,\ 0.5$ on CUB200-2011, and $\lambda = 0.01,\ 0.05,\ 0.1$ on Stanford Online Products. For the original triplet loss, we use the adapted triplet loss with $\lambda = 0$.

query images and 10 retrieval results for each query image using the adapted triplet loss and the original triplet loss respectively.

5 Conclusion

In this paper, we address the problem of triplet selection bias for triplet loss by using a domain adaption method. We propose an adapted triplet loss, which adaptively corrects the selection bias for the original triplet loss. Considering that the selection bias is common in deep metric learning, the proposed method can be extended to a variety of loss functions, e.g., pair-based [36], triplet-based[29], and quadruplet-based [5] loss functions, which will be the subject of future study.

(a) CARS196

(b) CUB200-2011

Fig. 7: Retrieval results on CARS196 and CUB200-2011. The first column is the query image. For each query image, the first row contains 10 nearest neighbors for the original triplet loss; The second row contains 10 nearest neighbors for the adapted triplet loss. We highlight false positive examples with a white/black cross (best view in color).

6 Acknowledgement

Baosheng Yu, Tongliang Liu, and Dacheng Tao were partially supported by Australian Research Council Projects FL-170100117, DP-180103424, LP-150100671. Changxing Ding was partially supported by the National Natural Science Foundation of China (Grant No.: 61702193) and Science and Technology Program of Guangzhou (Grant No.: 201804010272).

References

1. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. In: CVPR (2016)
2. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine Learning **79**(1), 151–175 (2010)
3. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al.: Analysis of representations for domain adaptation. In: NIPS. vol. 19, p. 137 (2007)
4. Chen, W., Chen, X., Zhang, J., Huang, K.: Beyond triplet loss: a deep quadruplet network for person re-identification. In: CVPR (2017)
5. Chen, W., Chen, X., Zhang, J., Huang, K.: Beyond triplet loss: a deep quadruplet network for person re-identification. In: CVPR. vol. 2 (2017)
6. Cheng, D., Gong, Y., Zhou, S., Wang, J., Zheng, N.: Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In: CVPR. pp. 1335–1344 (2016)
7. Cui, Y., Zhou, F., Lin, Y., Belongie, S.: Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In: CVPR. pp. 1153–1162 (2016)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. IEEE (2009)
9. Ding, C., Tao, D.: Trunk-branch ensemble convolutional neural networks for video-based face recognition. IEEE T-PAMI (2017)
10. Ding, S., Lin, L., Wang, G., Chao, H.: Deep feature learning with relative distance comparison for person re-identification. Pattern Recognition **48**(10), 2993–3003 (2015)
11. Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., Schölkopf, B.: Domain adaptation with conditional transferable components. In: ICML. pp. 2839–2848 (2016)
12. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. arXiv preprint arXiv:1604.01325 (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV. pp. 1026–1034 (2015)
14. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017)
15. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. arXiv preprint arXiv:1412.6622 (2014)
16. Huang, J., Gretton, A., Borgwardt, K.M., Schölkopf, B., Smola, A.J.: Correcting sample selection bias by unlabeled data. In: NIPS. pp. 601–608 (2007)
17. Huang, J., Feris, R.S., Chen, Q., Yan, S.: Cross-domain image retrieval with a dual attribute-aware ranking network. In: ICCV. pp. 1062–1070 (2015)
18. Iyer, A., Nath, S., Sarawagi, S.: Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In: ICML. pp. 530–538 (2014)
19. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
20. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 3dRR(Workshop) (2013)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012)

22. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: CVPR. pp. 3270–3278 (2015)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
24. Li, Y., Gong, M., Tian, X., Liu, T., Tao, D.: Domain generalization via conditional invariant representations. In: AAAI (2018)
25. Liu, H., Feng, J., Qi, M., Jiang, J., Yan, S.: End-to-end comparative attention networks for person re-identification. IEEE T-IP (2017)
26. Liu, H., Tian, Y., Yang, Y., Pang, L., Huang, T.: Deep relative distance learning: Tell the difference between similar vehicles. In: CVPR (2016)
27. Liu, T., Yang, Q., Tao, D.: Understanding how feature structure transfers in transfer learning. In: IJCAI. pp. 2365–2371 (2017)
28. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. JMLR **9**(Nov), 2579–2605 (2008)
29. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: CVPR. pp. 4004–4012 (2016)
30. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. IEEE T-NN **22**(2), 199–210 (2011)
31. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: BMVC. vol. 1, p. 6 (2015)
32. Prabhu, Y., Varma, M.: Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In: SIGKDD. pp. 263–272. ACM (2014)
33. Ramanathan, V., Li, C., Deng, J., Han, W., Li, Z., Gu, K., Song, Y., Bengio, S., Rosenberg, C., Fei-Fei, L.: Learning semantic relationships for better action retrieval in images. In: CVPR (2015)
34. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR. pp. 815–823 (2015)
35. Shi, H., Yang, Y., Zhu, X., Liao, S., Lei, Z., Zheng, W., Li, S.Z.: Embedding deep metric for person re-identification: A study against large variations. In: ECCV. pp. 732–748. Springer (2016)
36. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: NIPS. pp. 1857–1865 (2016)
37. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions. CVPR (2015)
38. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
39. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: CVPR. pp. 1386–1393 (2014)
40. Wang, L., Li, Y., Lazebnik, S.: Learning deep structure-preserving image-text embeddings. In: CVPR (2016)
41. Wang, X., Huang, T.K., Schneider, J.: Active transfer learning under model shift. In: ICML. pp. 1305–1313 (2014)
42. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. JMLR **10**(Feb), 207–244 (2009)
43. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: CVPR. pp. 3109–3118 (2015)
44. Wu, C.Y., Manmatha, R., Smola, A.J., Krahenbuhl, P.: Sampling matters in deep embedding learning. In: CVPR. pp. 2840–2848 (2017)

45. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
46. Xiao, T., Li, H., Ouyang, W., Wang, X.: Learning deep feature representations with domain guided dropout for person re-identification. In: CVPR. pp. 1249–1258 (2016)
47. Yuan, Y., Yang, K., Zhang, C.: Hard-aware deeply cascaded embedding. In: ICCV. pp. 814–823. IEEE (2017)
48. Zhang, K., Schölkopf, B., Muandet, K., Wang, Z.: Domain adaptation under target and conditional shift. In: ICML. pp. 819–827 (2013)
49. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: CVPR. pp. 5955–5964 (2016)