Pattern Recognition Letters 150 (2021) 49–56

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Dynamic sampling for deep metric learning

Chang-Hui Liang, Wan-Lei Zhao[1],*, Run-Qing Chen

Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen 361005, Fujian, China

A R T I C L E I N F O

Article history:
Received 10 August 2020
Revised 4 March 2021
Accepted 23 June 2021
Available online 21 July 2021

Keywords:
Deep metric learning
Training sample mining
Fashion search

A B S T R A C T

Deep metric learning maps visually similar images onto nearby locations and visually dissimilar images apart from each other in an embedding manifold. The learning process is mainly based on the supplied image negative and positive training pairs. In this paper, a dynamic sampling strategy is proposed to organize the training pairs in an easy-to-hard order to feed into the network. It allows the network to learn general boundaries between categories from the easy training pairs at its early stages and finalize the details of the model mainly relying on the hard training samples in the later. Compared to the existing training sample mining approaches, the hard samples are mined with little harm to the learned general model. This dynamic sampling strategy is formulated as two simple terms that are compatible with various loss functions. Consistent performance boost is observed when it is integrated with several popular loss functions on fashion search and fine-grained image search.

1. Introduction

Distance metric learning (usually referred to as metric learning), aims at constructing a task-specific distance measure based on given data. The learned distance metric is then used to support various tasks such as classification, clustering, and retrieval. Conventionally, the metric learning is designed to learn a matrix for the parametric *Mahalanobis* distance. Such that the similar contents are close to each other under the learned *Mahalanobis* distance, while the distance between the dissimilar contents is large

Due to the great success of deep learning in many computer vision tasks in recent years, it has been gradually introduced to metric learning, which is widely known as deep metric learning. Instead of learning the distance metric directly, deep metric learning learns feature embedding from the raw data. For instance, given images x_a, x_b and x_c, x_a and x_b are from the same category while x_c is distinct from them. The deep metric learning learns a non-linear mapping function $\mathcal{F}(\cdot)$ that embeds x_a, x_b and x_c to the new feature space. In this embedding space, $\mathcal{F}(x_a)$ and $\mathcal{F}(x_b)$ are close to each other, and $\mathcal{F}(x_c)$ is dissimilar to both of them under a pre-defined distance metric $m(\cdot,\cdot)$.

Owing to the seminal learning framework from [1], deep metric learning has been successfully adopted in various tasks such as online fashion search [14], face recognition [1,22], person re-identification [30], and fine-grained image search [18,23,27], etc. In general, the embedding space is learned on image pairs/triplets driven by loss functions. Namely, the training images are organized into positive pairs (images from the same category) and negative pairs (images from different categories). The loss function is designed to distill all the pair-based category information into a single loss value. The training process aims to build an embedding space by minimizing this loss. Such that the pairwise relations reconstructed in the embedding space coincide well with that of being supplied to the training. The general framework of deep metric learning is shown in Figure 1. Since the number of pairwise relations is quadratic to the size of training image set, it is computationally expensive to enumerate all the pairwise relations of the training set. As a consequence, the definition of loss function along with the ushered-in pair-sampling strategy becomes critical.

In the literature, a series of loss functions have been proposed one after another. *Contrastive loss* [5] and *triplet loss* [8] are the two most popular loss functions. However, both of them fail to make full use of the pairwise relations in a mini-batch. In addition, it is widely observed that the large portion of image pairs are easy training samples. Hard training samples, which take up a small portion, are more decisive to the category boundaries. Due to the lack of strategy to mine on these hard training samples, deep metric learning based solely on *contrastive loss* and *triplet loss* converges slowly. To alleviate this issue, *N-Pair loss* [23], *lifted structure loss* [18], and *multiple similarity loss* [27] consider more pairwise relationships within one mini-batch. This leads to a much faster convergence pace and better discriminativeness of the learned em-

* Corresponding author.
E-mail address: wlzhao@xmu.edu.cn (W.-L. Zhao).
[1] This paper has been submitted to "***Pattern Recognition Letters***"

https://doi.org/10.1016/j.patrec.2021.06.027
0167-8655/© 2021 Elsevier B.V. All rights reserved.

bedding space. The performance is further boosted by the mining of the hard negatives in [3,18,19,23,27].

In this paper, the training samples selection is novelly abstracted into two weighting terms and integrated with the loss function. It allows the training process to learn the embedding space in an order, namely from relatively easy samples to the harder. Therefore, the hard concepts (carried by hard samples) are learned without overwriting the learned general (easy) concepts. Moreover, this scheme is generic in the sense it could be integrated with various loss functions. Its effectiveness is confirmed on fashion search and fine-grained image search when it is integrated with *lifted structure loss* [18], *multiple similarity loss* [27], *triplet loss* [8], and *BD-loss* [30].

The remainder of this paper is organized as follows. Section 2 reviews the most representative loss functions in deep metric learning. Our dynamic sampling strategy is presented in Section 3. The comparative study over the representative loss functions and the proposed sampling strategy is presented in Section 4. Section 5 concludes the paper.

2. Related work

In this section, several representative loss functions and the enhancement strategies over them in deep metric learning are reviewed. In order to facilitate our later discussions, several concepts are defined. Given a pair of images $\{x_i, x_j\}$, the distance between them is given as

$$s_{i,j} = m(\mathcal{F}(x_i), \mathcal{F}(x_j)), \quad (1)$$

where $m(\cdot,\cdot)$ is a pre-defined distance measure. It could be *Cosine* similarity or *Euclidean* distance, etc. For clarity, the following discussion is made based on *Cosine* similarity by default. Correspondingly, the label for this image pair is given as $y_{i,j}$. Positive pair is given as $y_{i,j} = 1$, which indicates x_i and x_j come from the same category. While $y_{i,j}$ equals to 0 when they come from different categories.

Contrastive loss [5] encodes the similarities from both positive pairs and negative pairs in one loss function. Basically, it regularizes the similarities between positive pairs to be larger than the similarities from negative pairs with a constant margin λ. Namely,

$$\mathcal{L}_c = \frac{1}{m}\sum_{(i,j)}^{m/2}\Big(-y_{i,j}s_{i,j} + (1-y_{i,j})[0, s_{i,j} - \lambda]_+ \Big), \quad (2)$$

where m is the number of anchor-positive pairs in one training batch. $[\cdot]_+$ in Eq. (2) is the hinge loss. The minimization on $\mathcal{L}_c$ tends to converge as long as the distance between one pair satisfies with the margin λ.

Binomial deviance loss (*BD-loss*) [30] can be viewed as a soft version of *contrastive loss*. Its loss function is given as

$$\mathcal{L}_b = \sum_{i=1}^{m}\Bigg(\frac{1}{P}\sum_{y_{a,b}=1}\log\left[1 + e^{\alpha(\lambda - s_{a,b})}\right] + \frac{1}{N}\sum_{y_{c,d}=0}\log\left[1 + e^{\beta(s_{c,d}-\lambda)}\right]\Bigg), \quad (3)$$

where P and N are the numbers of positive pairs and negative pairs respectively. α and β in Eq. (3) are the scaling factors. According to recent studies [15,30], it shows superior performance on several challenging tasks.

In order to enhance the discriminativeness of the embedding space, *triplet loss* (defined in Eq. (4)) is designed to maximize the similarity between an anchor-positive pair $\{x_a, x_p\}$ in contrast to the similarity from anchor to a negative sample x_n.

$$\mathcal{L}_t = \frac{3}{2m}\sum_{i=1}^{m/2}[s_{a,n}^{(i)} - s_{a,p}^{(i)} + \lambda]_+ \quad (4)$$

Compared to *contrastive loss*, the gap between positive and negative pairs is defined in terms of relative similarity.

Recently, several efforts [3,18,19,23,27,29] have been made to mine on the hard training samples to further boost its performance. In order to make full use of the samples inside one mini-batch, *lifted structure loss* (see Eq. (5)) [18] is proposed. The loss function is designed to consider all the positive and negative pairs in one mini-batch.

$$\mathcal{L}_f = \sum_{i=1}^{m}\Big[\log\sum_{y_{i,j}=1} e^{\lambda - s_{i,j}} + \log\sum_{y_{i,j}=0} e^{s_{i,j}}\Big]_+ \quad (5)$$

The positive and negative pairs are treated equally in this loss function. As a result, the training batch is over-dominated by the negative pairs, which makes it easily stuck in local optima.

Multiple similarity loss [27] is proposed to consider the similarities from three types of negative pairs. The loss function is given in Eq. (6).

$$\mathcal{L}_m = \frac{1}{m}\sum_{i=1}^{m}\Bigg(\frac{1}{\alpha}\log[1 + \sum_{k\in\mathcal{P}_i} e^{-\alpha(s_{i,k}-\lambda)}] + \frac{1}{\beta}\log[1 + \sum_{k\in\mathcal{N}_i} e^{\beta(s_{i,k}-\lambda)}]\Bigg) \quad (6)$$

where α and β are the scaling parameters. In the above loss function, $\mathcal{P}_i$ and $\mathcal{N}_i$ are the positive and negative image sets respectively. According to [27], similar performance as *BD-loss* is reported.

In the literature, there are many efforts have been taken to enhance the performance by utilizing proxy points [10,16,24]. Encouraging performance is also achieved by augmenting the training set with synthesized hard negatives [2,13].

In our solution, the training samples selection is novelly abstracted to weighting terms and integrated with the loss function. It allows the training process to learn the embedding space in an order, namely from relatively easy samples to the harder. Therefore, the hard concepts (carried by hard samples) are learned without overwriting the learned general (easy) concept. Moreover, this scheme is generic in the sense it could be integrated with various loss functions. Its effectiveness is confirmed on fashion search and fine-grained image search when it is integrated with *lifted structure loss, multiple similarity loss, triplet loss*, and *BD-loss*.

3. Dynamic training pair selection

In this section, our strategies that are designed to boost the performance of deep metric learning are presented. In general, most of the deep metric learning approaches are defined based on the training image pairs. The variations across different approaches mainly lie in the selection of training pairs and the definition of loss function. In this section, we first present the strategies we used in the training pair selection. Based on the strategies, several popular loss functions, that are integrated with the proposed dynamic sampling terms, are presented.

3.1. Heuristics in pairs mining

As witnessed in many research works [3,18,19,27], the easy training samples take a large portion in a mini-batch, however they are less informative than the hard training pairs. Similar as other works [28], hard thresholds are set to filter out these easy training samples. To achieve that, two similarity thresholds τ_p and τ_n, namely one for easy positives and another for easy negative pairs, are introduced. Given the similarity between a positive pair $\{x_i, x_j\}$ is $s_{i,j}$, it will not be considered in the training if $s_{i,j} \geqslant \tau_p$. Similarly, a negative pair $\{x_k, x_m\}$ is not considered in the training as $s_{k,m} \leqslant \tau_n$. In our implementation, τ_p and τ_n are set to *0.9* and

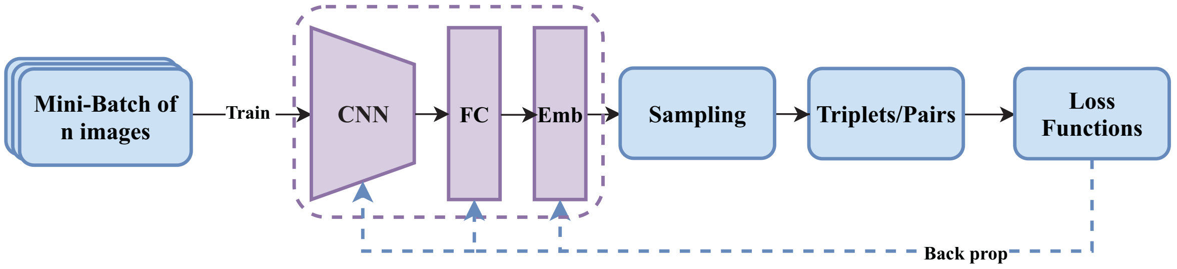

Fig. 1. The general framework of deep metric learning. The training images are organized into mini-batches. Images are then forward to a pre-trained ConvNets. The d-dimensional feature is produced by the 'Emb' layer. The fully-connected layer (FC) between ConvNets and 'Emb' is optional. The distances between image pairs are aggregated into a single loss value by a pre-defined loss function. The embedding space is optimized by iteratively minimizing this function loss.

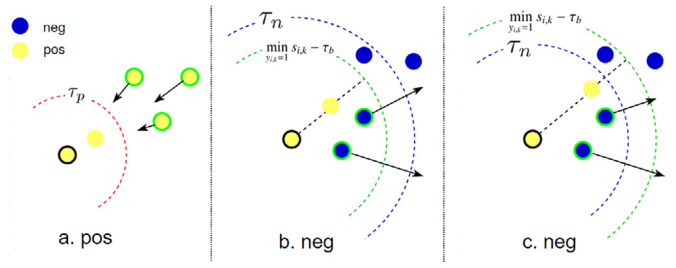

Fig. 2. The illustration of the role of three thresholds. The dashed circles in red, blue, and green represent boundaries regularized by τ_p, τ_n and Inequation 7 respectively. In figure (a), the positive samples which meet with $s_{i,k} < \tau_p$ will be selected. For negative pairs, they could be in either cases illustrated in figure (b) or figure (c). Under figure (b) and (c) cases, the negative pair $\{i, j\}$ that $s_{i,j} > \tau_n$ and $s_{i,j} > \min_{y_{i,k}} s_{i,k} - \tau_b$ will be selected. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

0.1 respectively based on an ablation analysis (Section 4.3). As the training continues, the boundary between positives and negatives becomes clearer. One could imagine growing number of negative and positive pairs is filtered by the thresholds and will no longer join in the training.

In addition to τ_p and τ_n, similar as [27], a flexible margin is set for negative pairs to separate them from the most remote positive pairs. Namely, given a positive pair $\{x_i, x_k\}$ and a negative pair $\{x_i, x_j\}$, $\{x_i, x_j\}$ will be selected to join in the training when the following inequation holds.

$$s_{i,j} > \min_{y_{i,k}=1} s_{i,k} - \tau_b, \tag{7}$$

where τ_b is the lower bound similarity of a positive sample to the anchor.

The roles that these three thresholds take are illustrated in Fig. 2. On the one hand, threshold τ_p prevents the training from pushing the positives as close as possible. On the other hand, threshold τ_n prevents the negative pairs from being pulled too far away. These two thresholds together prevent the structure of the learned embedding space from collapsing due to overfitting. According to our observation, very few training samples could pass through these two thresholds at the early training stage. As the training continues for several rounds, more and more negative pairs and positives are well separated in the embedding space. They are, therefore, set aside by these two thresholds. The training gets focus more and more on harder training samples. The flexible threshold given by Inequation 7 takes similar effect. As one could imagine, $\min_{y_{i,k}=1} s_{i,k}$ is relatively small at the early training stage. As the embedding space evolves to a better structure, $\min_{y_{i,k}=1} s_{i,k}$ grows bigger. This in turn thresholds out more and more relatively easy training samples.

3.2. Dynamic metric learning loss

Based on the above heuristics, only relatively hard pairs are joined in the training. Among these relatively hard training samples, the degree of hardness still varies from one training pair to another. In the ideal scenario, it is expected that the training samples are organized sequentially according to the degree of hardness. Samples with low degree of hardness are fed to the training at the early stages. As the training model evolves, harder training samples are fed to the training process since they become more critical to define the category boundaries. Intuitively, one has to prepare a group of image pairs for training with increasing degree of hardness each time. Although it sounds plausible, it is hard to operate as the hardness degree of one image pair varies along with the evolving embedding space. In the following, a novel weighting strategy based on image pair similarity $s_{i,j}$ is proposed. It regularizes the importance of a training pair according to its hardness in the training. The easy training samples are assigned with higher importance at the early training stages and the importance of hard training pairs grows as the training epoch increases.

In the existing loss functions, the similarity between one pair of image $s_{a,b}$ is mainly designed to aggregate the degree of penalty into the loss function. In our design, it is additionally used to indicate the hardness of a pair in one round of training. Specifically, for a positive pair $\{a, b\}$, the value $\tau_p - s_{a,b}$ basically indicates the hardness degree. The larger this value is, the harder the positive pair is. The value $s_{c,d} - \tau_n$ has the similar efficacy for a negative pair $\{c, d\}$. Let's take *BD-loss* as an example. We show how the training samples are re-weighted according to their degree of hardness. Given E_t is the number of total epochs we need to train our model, the current number of epochs that the training has been undertaken is given as E_c $(1 \le E_c \le E_t)$. Two terms $\frac{2E_c}{E_t}(\tau_p - s_{a,b})^2$ and $\frac{2E_c}{E_t}(s_{c,d} - \tau_n)^2$, one for positive and one for negative, are introduced to *BD-loss*. The *BD-loss* function is rewritten as

$$\begin{aligned}\mathcal{L}_b^* = \sum_{i=1}^{m} \Bigg\{ \frac{1}{P} \sum_{y_{a,b}=1} \log \left[1 + \mathrm{e}^{\alpha[(\lambda - \mathrm{s}_{a,b}) + \frac{2\mathrm{E_c}}{\mathrm{E_t}}(\tau_\mathrm{p} - \mathrm{s}_{a,b})^2]} \right] + \\ \frac{1}{N} \sum_{y_{c,d}=0} \log \left[1 + \mathrm{e}^{\beta[(\mathrm{s}_{c,d} - \lambda) + \frac{2\mathrm{E_c}}{\mathrm{E_t}}(\mathrm{s}_{c,d} - \tau_\mathrm{n})^2]} \right] \Bigg\},\end{aligned} \tag{8}$$

where λ acts as the minimum margin between positives and negatives for all the loss functions discussed in the paper. Apparently, these two terms are impacted by both E_c and the image pair similarity s. The larger the gap between s and the corresponding bound (either τ_p or τ_n) is, the higher these two terms are. This basically indicates the degree of hardness for a training pair (either positive or negative). Hard pairs tend to hold high weights. The terms are also controlled by the number of current epochs. The more number of epochs the training is undertaken, the higher of impact these two terms have on the overall loss $\mathcal{L}_b$. This leads the learning pro-

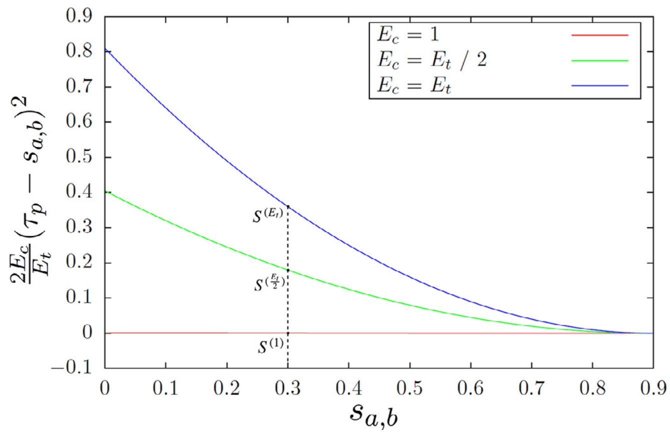

Fig. 3. The illustration of re-weighting term for positive pairs. The term is given as the function of similarity $s_{a,b}$. This term assigns different weights to different positive pairs according to their mutual similarities. The weights vary as the epoch E_c grows. The figure shows the re-weighting curves when training is at 1st, $E_t/2$th and E_tth epoch. As E_c grows, re-weighting is biased towards hard training pairs (whose $s_{a,b}$ is low). This leads to the higher contribution to the final loss.

cess to focusing on these hard training pairs more and more as E_c grows bigger.

Similarly, for *lifted structure loss* [18], *triplet loss* [8] and *multiple similarity loss* [27], they are rewritten as Eqs. (9)–(11) when the above two terms are integrated.

$$\mathcal{L}_f^* = \sum_{i=1}^{m} \Big\{ \log \sum_{y_{a,b}=1} e^{[(\lambda - s_{a,b}) + \frac{2E_c}{E_t}(\tau_p - s_{a,b})^2]} + \log \sum_{y_{c,d}=0} e^{[s_{c,d} + \frac{2E_c}{E_t}(s_{c,d} - \tau_n)^2]} \Big\}_+ \tag{9}$$

$$\mathcal{L}_t^* = \frac{3}{2m} \sum_{i=1}^{m/2} \Big\{ \frac{1}{P} \sum_{y_{a,b}=1} [-s_{a,b} + \frac{2E_c}{E_t}(\tau_p - s_{a,b})^2] + \frac{1}{N} \sum_{y_{c,d}=0} [s_{c,d} + \frac{2E_c}{E_t}(s_{c,d} - \tau_n)^2] + \lambda \Big\}_+ \tag{10}$$

$$\mathcal{L}_m^* = \frac{1}{m} \sum_{i=1}^{m} \Big\{ \frac{1}{\alpha} \log[1 + \sum_{a \in \mathcal{P}_i} e^{[-\alpha(s_{i,a} - \lambda) + \frac{2E_c}{E_t}(\tau_p - s_{i,a})^2]}] + \frac{1}{\beta} \log[1 + \sum_{b \in \mathcal{N}_i} e^{[\beta(s_{i,b} - \lambda) + \frac{2E_c}{E_t}(s_{i,c} - \tau_n)^2]}] \Big\} \tag{11}$$

In Eqs. (9)–(11), terms $\frac{2E_c}{E_t}(\tau_p - s_{a,b})^2$ and $\frac{2E_c}{E_t}(s_{c,d} - \tau_n)^2$ play a similar role as they do with *BD-loss*. Fig. 3 shows the weights produced by term $\frac{2E_c}{E_t}(\tau_p - s_{a,b})^2$ for positive pair with respect to $s_{a,b}$. The similarity of a positive pair $s_{a,b}$ is in the range of $[0.0, 0.9]$ after thresholding by τ_p. Low $s_{a,b}$ indicates the positive pair is close to the category boundary, namely it is a hard positive pair. As shown in the figure, the weights we assign to all the pairs are equally low at the first training epoch. The impact of this term on the loss function is therefore minor. Since the easy pairs take a large portion in one mini-batch, the training is actually biased towards easy pairs at the early stages. As epoch E_c grows, higher weights are assigned to pairs with lower similarities (as shown by the green curve in Fig. 3). The bias towards hard positives is more significant as epoch E_c grows even bigger, which in turn leads to the higher contribution from these hard positives to the loss function. Similar thing happens to the term for negative pairs. As a consequence, the optimization on the above models is tuned to focusing on the hard pairs gradually as the epoch grows.

As will be revealed in the experiment, the simple modification on the popular loss functions leads to considerable performance enhancement on different tasks. The performance improvement over the original model could be as high as more than *100%*.

The advantages of such modification are three folds. Firstly, the training samples are fed to the model from easy to hard automatically as the training epoch grows. This allows the training process to focus on relatively easy samples at the early stage and hard samples at its later stage. Secondly, this training sample selection rule is integrated with the loss function, no extra complexity is induced. Moreover, it is a generic strategy as it is compatible with different types of loss functions.

3.3. Implementation details

Inception [9] and ResNet-50 [6] (pre-trained on ImageNet [20]) are adopted respectively as the backbone network for our deep metric learning. Each training image is first fed into the network to generate a fixed-length feature vector, which maps the image into the embedding space. Thereafter, the images are organized into mini-batches. In each training iteration, *25* classes are randomly selected. Five images are randomly selected from each of these classes. This results in *125* images in one mini-batch. Thereafter, the image pairs which pass through the thresholds τ_p, τ_n and satisfy with Inequation 7 are selected to compute loss based on the revised function. For instance, Eq. (8) is employed for *BD-loss*. The computed loss for one mini-batch is back-propagated to optimize the network. α, λ, and β in Eq. (8) are set to *2, 0.5*, and *40* respectively. The above training process loops for E_t rounds.

For *lifted structure loss, triplet loss*, and *multiple similarity loss*, the training process remains largely the same. The only difference lies in the loss function. For *lifted structure loss, triplet loss*, and *multiple similarity loss*, Eqs. (9)–(11) are employed respectively. λ in Eq. (9) is set to *1.0*. λ in Eq. (10) is set to *0.5*. α, λ, and β in Eq. (11) are set to *2, 0.5*, and *50* respectively. All the codes are implemented with PyTorch and are publicly available on GitHub[2].

4. Experiments

In this section, the effectiveness of the proposed dynamic hard training sample mining strategy is studied when it is integrated with four popular loss functions on two deep learning backbones, namely Inception [9] and ResNet-50 [6]. The loss functions we consider are *BD-loss* (BD), *triplet loss* (TP), *multiple similarity loss* (MS), and *lifted structure loss* (LF). They are denoted as BD*, TP*, MS*, and LF* respectively when the proposed dynamic sampling strategy is integrated. Their behavior is comprehensively studied on two different tasks, namely fashion search and fine-grained image search. Their performance is compared to state-of-the-art approaches on each task.

4.1. Experiment design and setup

We noticed the experimental flaws pointed out by Musgrave et al. [17] in the current literature of deep metric learning. In our experiment design, four principles are held. 1. No sophisticated image augmentation is adopted. Images from all the datasets are resized to 256×256 and then randomly cropped to 224×224. the way of data augmentation follows with the configurations in [18]. Namely, the random cropping and random horizontal flips are employed during training and single cropping is employed during testing; 2. The *Adam* optimizer is adopted in the training for all the loss functions we consider. The output feature size is fixed to *512* for all the pull-out runs; 3. In order to alleviate overfitting, the parameters in Batch-Norm are frozen; 4. The provided training set

[2] https://github.com/CH-Liang/DSDML

Table 1
Summary over four datasets used in the evaluation.

Datasets (#Images/#Class)	#Train	#Validation	#Test
In-shop [14]	19,412/3,000	6,470/997	26,830/3,985
Deepfashion2 [4]	127,149/11,538	42,435/6,621	28,556/3,031
CUB200 [25]	4,360/74	1,504/26	5,924/100
Cars-196 [12]	6,018/73	2,036/25	8,131/98

Table 2
Ablation analysis of *BD-loss* on *In-shop* test set. $\tau_b = 0.2$, $\tau_p = .9$ and $\tau_b = 0.1$.

Recall@	Dim.	1	10	20	30	40	50
BD	512	85.4	95.3	96.8	97.4	97.8	98.1
BD+T	512	86.6	95.9	97.3	97.8	98.2	98.4
BD+W	512	87.4	97.0	98.1	98.4	98.6	98.7
*BD**	512	**87.6**	**97.3**	**98.3**	**98.6**	**98.8**	**98.9**

are divided into two. *25%* training samples are set aside for validation and the other *75%* are used in training. Above experiment design is to guarantee that the observed performance improvements are from the proposed dynamic sampling instead of the underlying tricks. The results from other relevant works are cited for reference. However, these results are not literally comparable to ours as the above principles are not necessarily held in the referred paper.

Following the convention in the literature, we report our performance in terms of Recall@K. To be line with the evaluation convention on different benchmarks, difference series of Ks are taken on different datasets. All the experiments are carried out on a server with NVIDIA GTX *1080* Ti GPU setup.

4.2. Datasets and evaluation protocols

For fashion search, datasets In-Shop (also known as Deepfashion) [14] and Deepfashion2 [4] are adopted in the evaluation. Dataset *In-Shop* is a collection of fashion product images crawled from online shopping websites. For dataset *Deepfashion2*, it is comprised of images both from online shops and users. Considerable portion of the images are directly collected from *Deepfashion*. For *Deepfashion2*, the retrieval is defined as user-to-shop query. Namely, the images uploaded by the users are treated as queries. The same product images crawled from online shopping websites are treated as the search targets. It is more challenging than *In-Shop* task as it is a cross-domain search problem. Since the test set for *Deepfashion2* is not released yet, the validation set is treated as the candidate dataset for search evaluation.

For fine-grained image search, *CUB-200-2011* [25] and *CARS-196* [12] are adopted, both of which are the most popular evaluation benchmarks in deep metric learning. The details about how each dataset is divided into training/validation/testing set are shown in Tab. 1.

4.3. Ablation study

In the ablation study, we first investigate the contributions from flexible threshold (given by Eq. (7)), the dynamic sampling and their combination. Afterwards, the choices of three similarity thresholds τ_b, τ_p, and τ_n are verified. All the experiments in this part are carried out with the backbone of Inception.

In the first experiment, the *BD-loss* that is integrated with simple thresholding with τ_p, τ_n and In Eq. (7) is given as “BD+T”. While *BD-loss* integrated with two re-weighting terms only is given as “BD+W”. *BD-loss* integrated with both is given as BD*. The results of these three runs on *In-shop* are shown in Table 2. The result from *BD-loss* is treated as the comparison baseline.

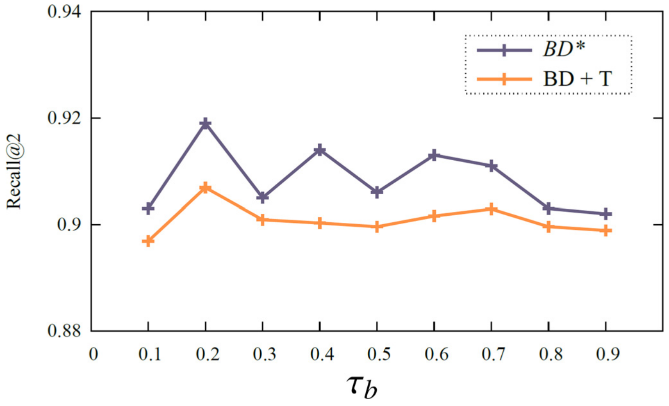

Fig. 4. The performance trends of BD+T and BD* on *CUB200* validation set as τ_b varies. In the experiment, τ_p and τ_n are fixed to 0.9 and 0.1 respectively.

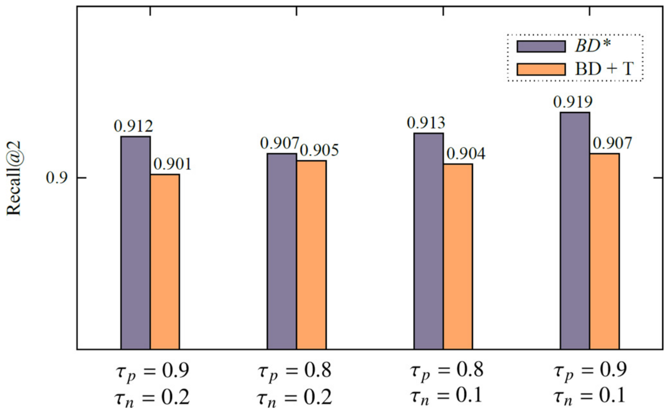

Fig. 5. The performance trend of BD+T and BD* on *CUB200* validation set as τ_p and τ_n vary. τ_b is fixed to 0.2.

As shown in the table, both schemes achieve consistent improvement. On average, the dynamic sampling brings more considerable improvement than simple thresholding. The best performance is observed when two schemes are integrated as a whole. This basically indicates they are complementary to each other. In the following experiments, all the four loss functions we study here are integrated with these two enhancement schemes.

In the above experiment, three similarity thresholds, namely τ_b, τ_p, and τ_n are fixed to 0.2, 0.9 and 0.1 respectively. In this experiment, we are going to verify whether the settings for these three thresholds are appropriate. The study is carried out on *CUB-200* dataset. The *BD-loss* is adopted. The parameter τ_b is verified first. In this experiment τ_p and τ_n are fixed to 0.9 and 0.1 respectively. In the second study, τ_b is fixed to 0.2, while different couplings of τ_p and τ_n are tested.

As shown in Fig. 4, the search performance of “BD+T” and “BD*” fluctuates as τ_b varies from *0.1* to *0.9*. In general, relatively higher performance is observed as τ_b is set to low similarity value. The highest performance is reached for both “BD+T” and “BD*” as $\tau_b = 0.2$. As a result, τ_b is fixed to 0.2 in the rest of experiments. The performance trend of “BD+T” and “BD*” on *CUB-200* is shown in Fig. 5 as different couplings of τ_p and τ_n are tested. As shown in the figure, the best performance is reached both for “BD+T” and “BD*” as τ_p and τ_n are set to 0.9 and 0.1 respectively. In the rest of our experiments, τ_p and τ_n are fixed to this setting.

Table 3
Comparison with the state-of-the-art approaches on *In-Shop*.

Recall@		Dim.	1	10	20	30	40	50
FashionNet‡ [14]		4,096	53.0	73.0	76.0	77.0	79.0	80.0
Divide‡ [21]		128	85.7	95.5	96.9	97.5	–	98.0
ABE‡ [11]		512	87.3	96.7	97.9	98.2	98.5	98.7
MIC‡ [19]		128	**88.2**	97.0	–	98.0	–	98.8
Inception	BD	512	85.4	95.3	96.8	97.4	97.8	98.1
	BD*	512	87.6	**97.3**	**98.3**	**98.6**	**98.8**	**98.9**
	LF	512	38.1	67.2	74.9	78.9	81.4	83.4
	LF*	512	86.7	96.3	97.6	98.1	98.4	98.5
	MS	512	85.8	95.7	97.2	97.8	98.1	98.4
	MS*	512	88.0	97.0	98.1	98.5	98.7	**98.9**
	TP	512	81.2	93.7	95.7	96.6	97.2	97.6
	TP*	512	83.4	94.5	96.2	97.0	97.4	97.8
Resnet-50	BD	512	80.7	92.5	94.4	95.5	96.1	96.7
	BD*	512	81.3	93.6	95.4	96.2	96.7	97.1
	LF	512	28.8	59.0	67.6	72.3	75.5	77.9
	LF*	512	78.5	92.6	94.6	95.6	96.2	96.7
	MS	512	80.8	91.9	94.3	95.5	96.2	96.6
	MS*	512	81.9	92.9	94.7	95.7	96.3	96.7
	TP	512	76.1	90.5	93.0	94.2	94.9	95.5
	TP*	512	78.2	91.4	93.5	94.4	95.1	95.6

‡ results are cited from the referred paper.

Table 4
Comparison with the state-of-the-art approaches on *Deepfashion2*

Recall@		Dim.	1	5	10	20
Match R-CNN‡ [4]		256	26.8	–	57.4	66.5
Angular‡ [26]		128	32.4	47.9	55.3	62.3
Hard Triplet‡ [7]		128	32.4	48.9	56.0	63.2
N-Pair‡ [23]		128	32.8	50.1	57.9	64.8
MIC [19]		512	38.1	52.1	59.3	66.3
Divide [21]		512	39.4	54.4	61.5	68.5
Inception	BD	512	39.6	54.4	61.8	69.3
	BD*	512	40.2	56.1	62.9	69.5
	LF	512	15.3	26.8	33.9	41.6
	LF*	512	40.6	57.1	65.2	71.2
	MS	512	40.5	56.5	63.7	70.8
	MS*	512	**42.2**	**58.4**	**65.7**	**72.3**
	TP	512	37.4	52.4	60.3	67.5
	TP*	512	39.3	55.2	62.9	70.4
Resnet-50	BD	512	31.3	45.7	52.5	60.2
	BD*	512	32.4	47.0	53.9	61.0
	LF	512	11.2	21.8	27.8	35.0
	LF*	512	33.5	48.6	56.6	65.1
	MS	512	34.5	48.9	56.1	63.5
	MS*	512	35.8	51.0	58.4	66.2
	TP	512	30.1	45.4	53.1	61.3
	TP*	512	35.3	50.3	57.5	65.0

‡ results are cited from the referred paper.

Table 5
Comparison with the state-of-the-art approaches on *CUB200*.

Recall@		Dim.	1	2	4	8	16	32
ABE‡ [11]		512	60.6	71.5	79.8	87.4	–	–
Divide‡ [21]		128	65.9	76.6	84.4	**90.6**	–	–
MIC‡ [19]		128	**66.1**	**76.8**	**85.6**	–	–	–
Inception	BD	512	61.6	71.8	81.8	88.6	93.2	96.3
	BD*	512	61.8	73.3	83.0	89.6	94.0	96.9
	LF	512	46.2	58.7	70.1	80.1	87.2	92.6
	LF*	512	58.4	69.5	78.6	86.3	91.6	95.2
	MS	512	62.6	73.5	82.1	88.9	93.2	96.5
	MS*	512	63.1	74.2	83.0	90.0	**94.3**	**97.1**
	TP	512	51.0	62.3	72.7	81.9	87.9	92.4
	TP*	512	54.6	66.8	77.2	84.7	90.4	94.4
Resnet-50	BD	512	28.4	39.3	49.8	62.3	73.8	83.2
	BD*	512	40.2	52.2	63.7	74.5	83.2	90.6
	LF	512	15.2	22.6	32.1	43.8	57.9	71.3
	LF*	512	32.9	44.7	57.1	68.5	79.1	87.0
	MS	512	44.0	57.1	68.4	78.6	86.0	95.4
	MS*	512	45.1	58.2	69.8	79.8	86.6	95.8
	TP	512	25.5	36.6	48.9	61.5	72.7	82.8
	TP*	512	28.5	39.1	50.9	63.1	74.4	84.3

‡ results are cited from the referred paper.

4.4. Fashion search

In this section, the effectiveness of the proposed enhancement strategy on four loss functions is studied in fashion search task. Representative approaches in the literature on this task are considered in the study. FashionNet [14] and Match R-CNN [4] are treated as the comparison baselines for *In-Shop* and *Deepfashion2* respectively. They are proposed along with these two benchmarks. The fashion search is treated as a sub-task under the multi-task learning framework. In addition, recent deep metric learning approaches Divide and Conquer (Divide) [21], Mining Interclass Characteristics (MIC) [19], *Angular loss* (Angular) [26], *Batch hard triplet loss* (Hard Triplet) [7] and *N-Pair loss* (N-Pair) [23] are also considered in the comparison. Among these approaches, MIC learns auxiliary encoder for the visual attributes, which induces extra computational costs. Attention-based Ensemble (ABE) [11] is the representative approach of ensemble deep metric learning. Due to the difference in training settings, the results from these papers are not literally comparable to ours. The results are cited for reference only.

The performance on *In-Shop* and *Deepfashion2* is shown on Tables 3 and 4 respectively. As shown in Tables 3 and 4, all the four loss functions that are integrated with dynamic sampling terms demonstrate considerable performance improvement. The improvement ranges from *10-100%* for different loss functions. Being integrated with the dynamic sampling, the performance from all loss functions with both backbones becomes competitive to or even outperforms the most effective approach in the literature, in particular on the challenging dataset *Deepfashion2*. Although ResNet-50 turns out to be more powerful than Inception in many scenarios, Inception shows better performance in our implementation. When no additional training tricks are involved, the features from pre-trained ResNet-50 lack of discriminativeness between samples within the same category (*e.g.*, clothes). According to our observation, Inception is more effective in dealing with fine-grained search problems. An interesting observation is that the performance difference between different loss functions becomes much smaller when being all supported by the dynamic sampling.

The search result samples from MS and MS* on *In-Shop* are shown in Fig. 6. As shown in the figure, the false positives returned by MS are visually very similar to the queries. They are the hard samples. When being integrated with the weighting terms, the fine-grained details carried by these hard samples can be well learned by MS* at its later stages. As shown in the figure, the MS* turns out to be robust to severe variations in fine-grained texture, view point, and color.

4.5. Fine-grained image search

In this section, the performance of the enhanced loss functions is evaluated on *CUB-200-2011* and *CARS-196* for fine-grained image search. Three state-of-the-art deep metric learning approaches Divide [21], ABE [11] and MIC [19] are considered in the comparison. The performance results on these two datasets are shown in Tables 5 and 6 respectively.

The modified loss functions demonstrate consistent improvement over the original ones. The improvement is significant on *lifted structure loss*. Similar as the fashion search task, the performance gap becomes minor among different loss functions when dynamic sampling is adopted. Due to the small scale of *CUB-200-2011* and *CARS-196*, the impact of hyper-parameters becomes greater which leads to significant performance drops with ResNet-50 backbone. Overall, BD* and MS* perform competitively well

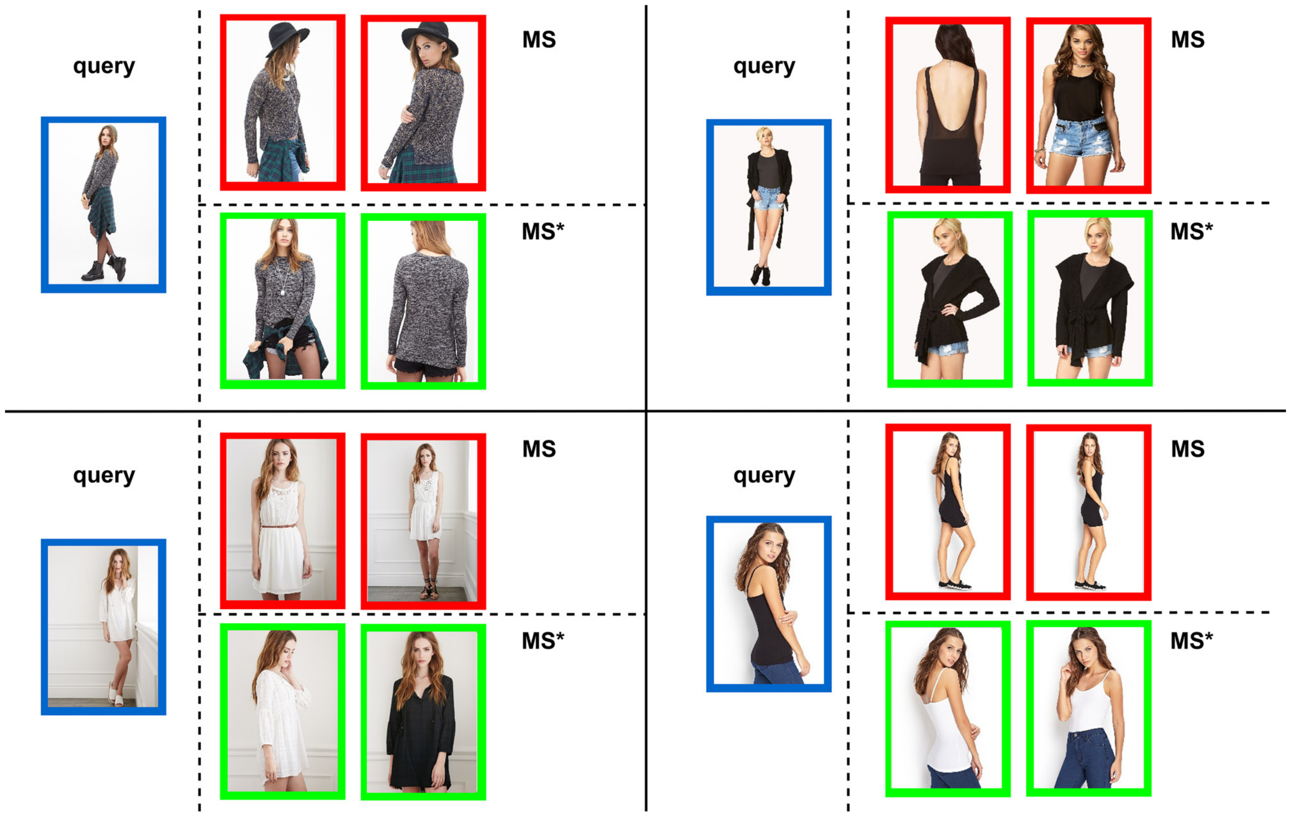

Fig. 6. Top-2 search results on *In-Shop*. The queries are shown on the left and followed by retrieved top-2 results from MS and MS* with Inception as the backbone. Our approach improves the performance significantly and shows stable performance even under severe transformations.

Table 6
Comparison with the state-of-the-art approaches on *Cars-196*.

Recall@		Dim.	1	2	4	8	16	32
ABE‡ [11]		512	**85.2**	90.5	94.0	96.1	–	–
Divide‡ [21]		128	84.6	**90.7**	**94.1**	**96.5**	–	–
MIC‡ [19]		128	82.6	89.1	93.2	–	–	–
Inception	BD	512	74.3	83.0	88.9	93.2	96.3	98.1
	BD*	512	75.7	84.4	90.6	94.8	**97.2**	**98.8**
	LF	512	36.1	47.6	60.7	71.8	81.5	88.5
	LF*	512	69.0	78.9	86.0	91.6	95.2	97.6
	MS	512	77.0	84.3	89.8	93.5	96.2	98.0
	MS*	512	78.0	85.6	90.8	94.3	96.7	98.4
	TP	512	51.1	62.4	72.6	80.2	86.7	91.8
	TP*	512	62.2	72.2	80.5	87.0	92.1	95.5
Resnet-50	BD	512	55.3	66.1	75.2	83.2	88.9	93.2
	BD*	512	56.6	67.2	76.6	83.8	89.6	93.7
	LF	512	22.6	32.3	43.3	55.9	68.4	79.6
	LF*	512	53.2	65.4	75.1	82.7	88.8	93.4
	MS	512	61.7	72.2	80.8	87.8	92.9	96.2
	MS*	512	65.8	75.7	83.2	89.1	93.5	96.2
	TP	512	45.1	56.6	66.7	75.9	83.6	89.2
	TP*	512	46.2	57.9	68.3	77.2	84.7	90.4

‡ results are cited from the referred paper.

with the state-of-the-art approaches, namely Divide and MIC. Compared to Divide and MIC, our approach is much lightweight. Approach Divide requires to learn several sub-embedding spaces. Similarly, extra computation is required in MIC to train the auxiliary encoder for the latent visual attributes. In contrast, considerable improvement from our approach is achieved by injecting two re-weighting terms into the loss function. No sophisticated modification on the training process or the network architecture is required.

Overall, *BD-loss* and *multiple similarity loss* show considerably superior performance over other loss functions on two different tasks. With the proposed dynamic sampling strategy, both models show performance boost on two tasks and across different parameter settings. In particular, the performance from the enhanced *BD-loss* and *multiple similarity loss* is better than or close to state-of-the-art approaches on two tasks.

5. Conclusion

We have presented a simple but effective dynamic sampling strategy to boost the performance of deep metric learning. In our solution, the dynamic sampling is formulated as two terms that are compatible with various loss functions. These two re-weighting terms dynamically tune the impact that a training pair contribute to the loss function. This allows the network to learn the concepts from easy to hard, which is comparable to the cognitive process of human beings. With this dynamic sampling strategy, the performance gap among different loss functions becomes minor. Consistent improvement on two tasks is observed with all the loss functions on two popular backbones.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by National Natural Science Foundation of China under grants 61572408 and 61972326, and the grants of Xiamen University 20720180074.

References

[1] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, 2005, pp. 539–546.
[2] Y. Duan, W. Zheng, X. Lin, J. Lu, J. Zhou, Deep adversarial metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2780–2789.
[3] W. Ge, Deep metric learning with hierarchical triplet loss, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 269–285.
[4] Y. Ge, R. Zhang, X. Wang, X. Tang, P. Luo, Deepfashion2: a versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5337–5345.
[5] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), IEEE, 2006, pp. 1735–1742.
[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[7] A. Hermans, L. Beyer, B. Leibe, In defense of the triplet loss for person re-identification, arXiv preprint arXiv:1703.07737
[8] E. Hoffer, N. Ailon, Deep metric learning using triplet network, in: International Workshop on Similarity-Based Pattern Recognition, Springer, 2015, pp. 84–92.
[9] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[10] S. Kim, D. Kim, M. Cho, S. Kwak, Proxy anchor loss for deep metric learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3238–3247.
[11] W. Kim, B. Goyal, K. Chawla, J. Lee, K. Kwon, Attention-based ensemble for deep metric learning, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 736–751.
[12] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3d object representations for fine-grained categorization, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2013, pp. 554–561.
[13] X. Lin, Y. Duan, Q. Dong, J. Lu, J. Zhou, Deep variational metric learning, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 689–704.
[14] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, Deepfashion: Powering robust clothes recognition and retrieval with rich annotations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1096–1104.
[15] D. Manandhar, M. Bastan, K.H. Yap, Semantic granularity metric learning for visual search, J. Vis. Commun. Image Represent. 72 (2020) 102871.
[16] Y. Movshovitz-Attias, A. Toshev, T.K. Leung, S. Ioffe, S. Singh, No fuss distance metric learning using proxies, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 360–368.
[17] K. Musgrave, S. Belongie, S.N. Lim, A metric learning reality check, in: European Conference on Computer Vision, Springer, 2020, pp. 681–699.
[18] H. Oh Song, Y. Xiang, S. Jegelka, S. Savarese, Deep metric learning via lifted structured feature embedding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4004–4012.
[19] K. Roth, B. Brattoli, B. Ommer, Mic: mining interclass characteristics for improved metric learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8000–8009.
[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.
[21] A. Sanakoyeu, V. Tschernezki, U. Buchler, B. Ommer, Divide and conquer the embedding space for metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 471–480.
[22] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
[23] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, in: Advances in Neural Information Processing Systems, 2016, pp. 1857–1865.
[24] E.W. Teh, T. DeVries, G.W. Taylor, Proxynca++: revisiting and revitalizing proxy neighborhood component analysis, in: European Conference on Computer Vision (ECCV), Springer, 2020.
[25] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200-2011 dataset, 2011,
[26] J. Wang, F. Zhou, S. Wen, X. Liu, Y. Lin, Deep metric learning with angular loss, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2593–2601.
[27] X. Wang, X. Han, W. Huang, D. Dong, M.R. Scott, Multi-similarity loss with general pair weighting for deep metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5022–5030.
[28] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, N.M. Robertson, Ranked list loss for deep metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5207–5216.
[29] C.Y. Wu, R. Manmatha, A.J. Smola, P. Krahenbuhl, Sampling matters in deep embedding learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2840–2848.
[30] D. Yi, Z. Lei, S.Z. Li, Deep metric learning for practical person re-identification, arXiv preprint arXiv:1407.4979