arXiv:1904.06627v2 [cs.CV] 11 May 2019

Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning

Xun Wang, Xintong Han, Weilin Huang*, Dengke Dong, Matthew R. Scott
Malong Technologies, Shenzhen, China
Shenzhen Malong Artificial Intelligence Research Center, Shenzhen, China
{xunwang,xinhan,whuang,dongdk,mscott}@malong.com

Abstract

A family of loss functions built on pair-based computation have been proposed in the literature which provide a myriad of solutions for deep metric learning. In this paper, we provide a general weighting framework for understanding recent pair-based loss functions. Our contributions are three-fold: (1) we establish a General Pair Weighting (GPW) framework, which casts the sampling problem of deep metric learning into a unified view of pair weighting through gradient analysis, providing a powerful tool for understanding recent pair-based loss functions; (2) we show that with GPW, various existing pair-based methods can be compared and discussed comprehensively, with clear differences and key limitations identified; (3) we propose a new loss called multi-similarity loss (MS loss) under the GPW, which is implemented in two iterative steps (i.e., mining and weighting). This allows it to fully consider three similarities for pair weighting, providing a more principled approach for collecting and weighting informative pairs. Finally, the proposed MS loss obtains new state-of-the-art performance on four image retrieval benchmarks, where it outperforms the most recent approaches, such as ABE[14] and HTL [4], by a large margin, e.g., $60.6\% \rightarrow 65.7\%$ *on CUB200, and* $80.9\% \rightarrow 88.0\%$ *on In-Shop Clothes Retrieval dataset at Recall@1. Code is available at* https://github.com/MalongTech/research-ms-loss

1. Introduction

Metric learning aims to learn an embedding space, where the embedded vectors of similar samples are encouraged to be closer, while dissimilar ones are pushed apart from each other [22, 23, 39]. With recent great success of deep neural networks in computer vision, deep metric learning has attracted increasing attention, and has been applied to various tasks, including image retrieval [37, 8, 5], face recognition [36], zero-shot learning [42, 1, 15], visual tracking [19, 31] and person re-identification [41, 9].

Corresponding author: whuang@malong.com

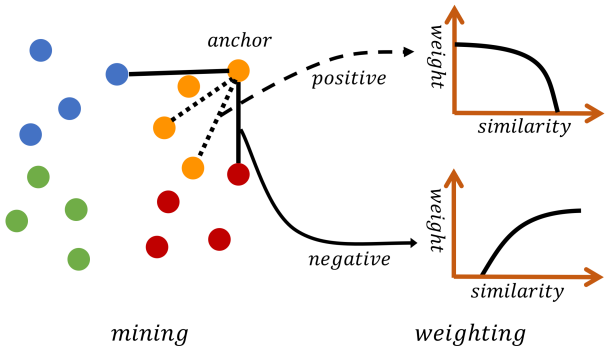

Figure 1. The proposed multi-similarity loss is able to jointly measure the self-similarity and relative similarities of a pair, which allows it collect informative pairs by implementing iterative pair mining and weighting.

Many recent deep metric learning approaches are built on pairs of samples. Formally, their loss functions can be expressed in terms of pairwise cosine similarities in the embedding space[1]. We refer to this group of methods as *pair-based* deep metric learning; and this family includes contrastive loss [6], triplet loss [10], triplet-center loss [8], quadruplet loss [18], lifted structure loss [25], N-pairs loss [29], binomial deviance loss [40], histogram loss [32], angular loss [34], distance weighted margin-based loss [38], hierarchical triplet loss (HTL) [4], *etc*. For these pair-based methods, training samples are constructed into pairs, triplets or quadruplets, resulting a polynomial growth of training pairs which are highly redundant and less informative. This gives rise to a key issue for pair-based methods, where training with random sampling can be overwhelmed by redundant pairs, leading to slow convergence and model degeneration with inferior performance.

Recent efforts have been devoted to improving sampling

[1]For simplicity, we use a cosine similarity instead of Euclidean distance, by assuming an embedding vector is L_2 normalized.

schemes for pair-based metric learning techniques. For example, Chopra *et al.* [3] introduced a contrastive loss which discards negative pairs whose similarities are smaller than a given threshold. In triplet loss [10], a negative pair is sampled by using a margin computed from the similarity of a randomly selected positive pair. Alternatively, lifted structure loss [25] and N-pairs loss [29] introduced new weighting schemes by designing a smooth weighting function to assign a larger weight to a more informative pair. Though driven by different motivations, these methods share a common goal of learning from more informative pairs. Thus, sampling such informative pairs is the key to pair-based deep metric learning, while precisely identifying these pairs is particularly challenging, especially for the negative pairs whose number is quadratic to the size of dataset.

In this work, we cast the sampling problem of deep metric learning into a general pair weighting formulation. We investigated various weighting schemes of recent pair-based loss functions, attempted to understand their insights more deeply, and identify key limitations of these approaches. We observed that a key factor that impacts pair weighting is to compute multiple types of similarities for a pair, which can be defined as self-similarity and relative similarity, where the relative similarity is heavily dependent on the other pairs. Furthermore, we found that most existing methods only explore this factor partially, which limits their capability considerably. For example, contrastive loss [6] and binomial deviance loss [40] only consider the cosine similarity of a pair, while triplet loss [10] and lifted structure loss [25] mainly focus on the relative similarity. We propose a multi-similarity loss which fully considers multiple similarities during sample weighting. The major contributions of this paper are summarized as follows.

- We establish a General Pair Weighting (GPW) framework, which formulates deep metric learning into a unified view of pair weighting. It provides a general formulation for understanding and explaining various pair-based loss functions through gradient analysis.

- We analyze the key factor that impacts pair weighting with GPW, where various pair-based methods can be analyzed comprehensively, with main differences and key limitations clearly identified. This allows us to define three types of similarities for a pair: a self-similarity and two relative similarities. The relative similarities are computed by comparing to other pairs, which are of great importance to existing pair-based methods.

- We propose a new multi-similarity (MS) loss, which is implemented using two iterative steps with sampling and weighting, as shown in Fig. 1. MS loss considers both self-similarity and relative similarities which enables the model to collect and weight informative pairs more efficiently and accurately, leading to boosts in performance.

- MS loss is evaluated extensively on a number of benchmarks for image retrieval, where it outperforms current state-of-the-art approaches by a large margin, *e.g.*, improving recent ABE [14] with +5.0% Recall@1 on CUB200, and HTL [4] with +7.1% Recall@1 on In-Shop Clothes Retrieval dataset.

2. Related Work

Classical pair-based loss functions. *Siamese* network [6] is a representative pair-based method that learns an embedding via contrastive loss. It encourages samples from a positive pair to be closer, and pushes samples from a negative pair apart from each other, in the embedding space. Triplet loss was introduced in [10] by using triplets as training samples. Each triplet consists of a positive pair and a negative pair by sharing the same anchor point. Triplet loss aims to learn an embedding space where the similarity of a negative pair is lower than that of a positive one, by giving a margin. Extended from triplet loss, quadruplets were also applied in recent work, such as histogram loss [32].

Recently, Song *et al*. [25] argued that both contrastive loss and triplet loss are difficult to explore full pair-wise relations between samples in a mini-batch. They proposed a lifted structure loss attempted to fully utilize such pair-wise relations. However, the lifted structure loss only samples approximately an equal number of negative pairs as the positive ones randomly, and thus discards a large number of informative negative pairs arbitrarily. In [40], Dong *et al*. proposed a binomial deviance loss by using a binomial deviance to evaluate the cost between labels and similarity, which emphasizes harder pairs. In this work, we propose a multi-similarity loss able to explore more meaningful pair-wise relations by jointly considering both self-similarity and the relative similarities.

Hard sample mining. Pair-based metric learning often generates a large amount of pair-wise samples, which are highly redundant and include many uninformative samples. Training with random sampling can be overwhelmed by these redundant samples, which significantly degrade the model capability and also slows the convergence. Therefore, sampling plays a key role in pair-based metric learning.

The importance of hard negative mining has been discussed extensively [28, 7, 38, 4]. Schroff *et al*. [28] proposed a semi-hard mining scheme by exploring semi-hard triplets, which are defined to have a negative pair farther than the positive one. However, such semi-hard mining method only generates a small number of valid semi-hard triplets, so that it often requires a large batch-size to generate sufficient semi-hard triplets, *e.g.*, 1800 as suggested in

[28]. Harwood *et al*. [7] provided a framework named smart mining to collect hard samples from the whole dataset, which suffers from off-line computation burden. Recently, Ge *et al*. [4] proposed a hierarchical triplet loss (HTL) which builds a hierarchical tree of all classes, where hard negative pairs are collected via a dynamic margin. Sampling matters in deep embedding learning was discussed in [38], and a *distance weighted sampling* was proposed to collect negative samples uniformly with respective to the pair-wise distance. Unlike these methods which mainly focus on sampling or hard sample mining, we provide a more generalized formulation that casts sampling problem into general pair weighting.

Instance weighting. Instance weighting has been widely applied to various tasks. For example, Lin *et al*. [20] proposed a focal loss that allows the model to focus on hard negative examples during training an object detector. In [2], an active bias learning was developed to emphasize high variance samples in training a neural network for classification. Self-paced learning [17], which pays more attention on samples with a higher confidence, was explored to design noise-robust algorithms [12]. These approaches [20, 13, 2, 17] were developed for weighting individual instances that are only depended on itself (referred as self-similarity), while our method aims to compute both self-similarity and the relative similarities, which is a more complicated problem that requires to measure multiple sample correlations within a local data distribution.

3. General Pair Weighting (GPW)

In this section, we formulate the sampling problem of metric learning into a unified weighting view, and provide a general pair weighting (GPW) framework for analyzing various pair-based loss functions.

3.1. GPW Framework

Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be a real-value instance vector. Then we have an instance matrix $\boldsymbol{X} \in \mathbb{R}^{m\times d}$, and a label vector $\boldsymbol{y} \in \{1, 2, \ldots, C\}^m$ for the m training samples respectively. Then an instance $\boldsymbol{x}_i$ is projected onto a unit sphere in a l-dimension space by $f(\cdot;\boldsymbol{\theta}) : \mathbb{R}^d \to S^l$, where f is a neural network parameterized by $\boldsymbol{\theta}$. Formally, we define the similarity of two samples as $S_{ij} := < f(\boldsymbol{x}_i;\boldsymbol{\theta}), f(\boldsymbol{x}_j;\boldsymbol{\theta}) >$, where $< \cdot, \cdot >$ denotes dot product, resulting in an $m \times m$ similarity matrix $\boldsymbol{S}$ whose element at (i, j) is S_{ij}.

Given a *pair-based* loss $\mathcal{L}$, it can be formulated as a function in terms of $\boldsymbol{S}$ and $\boldsymbol{y}$: $\mathcal{L}(\boldsymbol{S}, \boldsymbol{y})$. The derivative with respect to model parameters $\boldsymbol{\theta}$ at the t-th iteration can be calculated as:

$$\begin{aligned}\left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial \boldsymbol{\theta}}\right|_t &= \left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial \boldsymbol{S}}\right|_t \left.\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{\theta}}\right|_t \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} \left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t \left.\frac{\partial S_{ij}}{\partial \boldsymbol{\theta}}\right|_t .\end{aligned} \tag{1}$$

Eq. 1 is computed for optimizing model parameters $\boldsymbol{\theta}$ in deep metric learning. In fact, Eq. 1 can be reformulated into a new form for pair weighting through a new function $\mathcal{F}$, whose gradient w.r.t. $\boldsymbol{\theta}$ at the t-th iteration is computed exactly the same as Eq. 1. $\mathcal{F}$ is formulated as below:

$$\mathcal{F}(\boldsymbol{S},\boldsymbol{y}) = \sum_{i=1}^{m}\sum_{j=1}^{m} \left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t S_{ij}. \tag{2}$$

Note that $\left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t$ is regarded as a **constant scalar** that not involved in the gradient of $\mathcal{F}$ w.r.t. $\boldsymbol{\theta}$.

Since the central idea of deep metric learning is to encourage positive pairs to be closer, and push negatives apart from each other. For a pair-based loss $\mathcal{L}$, we can assume $\left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t \geqslant 0$ for a negative pair, and $\left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t \leqslant 0$ for a positive pair. Thus, $\mathcal{F}$ in Eq. 2 can be transformed into the form of pair weighting as follows:

$$\begin{aligned}\mathcal{F} &= \sum_{i=1}^{m}\left(\sum_{\boldsymbol{y}_j\neq\boldsymbol{y}_i}^{m} \left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t S_{ij} + \sum_{\boldsymbol{y}_j=\boldsymbol{y}_i}^{m} \left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t S_{ij}\right) \\ &= \sum_{i=1}^{m}\left(\sum_{\boldsymbol{y}_j\neq\boldsymbol{y}_i}^{m} w_{ij}S_{ij} - \sum_{\boldsymbol{y}_j=\boldsymbol{y}_i}^{m} w_{ij}S_{ij}\right),\end{aligned} \tag{3}$$

where $w_{ij} = \left|\left.\frac{\partial \mathcal{L}(\boldsymbol{S},\boldsymbol{y})}{\partial S_{ij}}\right|_t\right|$.

As indicted in Eq. 3, a pair-based method can be formulated as weighting of pair-wise similarities, where the weight for pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ is w_{ij}. Learning with a pair-based loss function $\mathcal{L}$ is now transformed from Eq. 1 into designing weights for pairs. It is a general pair weighting (GPW) formulation, and sampling is only a special cases.

3.2. Revisit Pair-based Loss Functions

To demonstrate the generalization ability of GPW framework, we revisit four typical pair-based loss functions for deep metric learning: contrastive loss [6], triplet loss [10], binomial deviance loss [40] and lifted structure loss [25].

Contrastive loss. Hadsell *et al*. [6] proposed a Siamese network, where a contrastive loss was designed to encourage positive pairs to be as close as possible, and negative pairs to be apart from each other over a given threshold, λ:

$$\mathcal{L}_{contrast} := (1 - \mathcal{I}_{ij})[S_{ij} - \lambda]_+ - \mathcal{I}_{ij}S_{ij}, \tag{4}$$

where $\mathcal{I}_{ij} = 1$ indicates a positive pair, and 0 for a negative one. By computing partial derivative with respect to S_{ij} in Eq. 4, we can find that all positive pairs and hard negative pairs with $S_{ij} > \lambda$ are assigned with an equal weight. This is a simple and special case of our pair weighting scheme, without considering any difference between the selected pairs.

Triplet loss. In [10], a triplet loss was proposed to learn a deep embedding, which enforces the similarity of a negative pair to be smaller than that of a randomly selected positive one over a given margin λ:

$$\mathcal{L}_{triplet} := [S_{an} - S_{ap} + \lambda]_+, \tag{5}$$

where S_{an} and S_{ap} denote the similarity of a negative pair $\{\boldsymbol{x}_a, \boldsymbol{x}_n\}$, and a positive pair $\{\boldsymbol{x}_a, \boldsymbol{x}_p\}$, with an anchor sam, e$pl\boldsymbol{x}_a$. According to the gradient computed for Eq. 5, a triplet loss weights all pairs equally on the valid triplets which are selected by $S_{an} + \lambda > S_{ap}$, while the triplets with $S_{an} + \lambda \leqslant S_{ap}$ are considered as less informative, and are discarded. Triplet loss is different from contrastive loss on pair selection scheme, but both methods consider all the selected pairs equally, which limits their ability to identify more informative pairs among the selected ones.

Lifted Structure Loss. Song *et al*. [25] designed a lifted structure loss, which was further improved to a more generalized version in [9]. It utilizes all the positive and negative pairs among the mini-batch as follows:

$$\mathcal{L}_{lifted} := \sum_{i=1}^{m} \left[\log \sum_{\boldsymbol{y}_k = \boldsymbol{y}_i} e^{\lambda - S_{ik}} + \log \sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{S_{ik}} \right]_+, \tag{6}$$

where λ is a fixed margin.

In Eq. 6, when the hinge function of an anchor $\boldsymbol{x}_i$ returns a non-zero value, we can have a weight value, w_{ij}, for the pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$, by differentiating $\mathcal{L}_{lifted}$ on S_{ij}, according to Eq. 3. Then the weight for a positive pair is computed as:

$$w_{ij}^{+} = \frac{e^{\lambda - S_{ij}}}{\sum_{\boldsymbol{y}_k = \boldsymbol{y}_i} e^{\lambda - S_{ik}}} = \frac{1}{\sum_{\boldsymbol{y}_k = \boldsymbol{y}_i} e^{S_{ij} - S_{ik}}}, \tag{7}$$

and the weight for a negative pair is:

$$w_{ij}^{-} = \frac{e^{S_{ij}}}{\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{S_{ik}}} = \frac{1}{\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{S_{ik} - S_{ij}}}. \tag{8}$$

Eq. 7 shows that the weight for a positive pair is determined by its relative similarity, measured by comparing it to the remained positive pairs with the same anchor. The weight for a negative pair is computed similarly based on Eq. 8.

Binomial Deviance Loss. Dong *et al*. introduced binomial deviance loss in [40], which utilizes softplus function instead of hinge function in contrastive loss:

$$\begin{aligned}\mathcal{L}_{binomial} = \sum_{i=1}^{m} \Bigg\{ \frac{1}{P_i} \sum_{\boldsymbol{y}_j = \boldsymbol{y}_i} \log \left[1 + e^{\alpha(\lambda - S_{ij})} \right] + \\ \frac{1}{N_i} \sum_{\boldsymbol{y}_j \neq \boldsymbol{y}_i} \log \left[1 + e^{\beta(S_{ij} - \lambda)} \right] \Bigg\},\end{aligned} \tag{9}$$

where P_i and N_i denote the numbers of positive pairs and negative pairs with anchor $\boldsymbol{x}_i$, respectively. λ, α, β are fixed hyper-parameters.

The weight for pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ is w_{ij} in Eq. 1, which can be derived from differentiating $\mathcal{L}_{binomial}$ on S_{ij} as:

$$\begin{aligned} w_{ij}^{+} &= \frac{1}{P_i} \frac{\alpha e^{\alpha(\lambda - S_{ij})}}{1 + e^{\alpha(\lambda - S_{ij})}}, \qquad \boldsymbol{y}_j = \boldsymbol{y}_i \\ w_{ij}^{-} &= \frac{1}{N_i} \frac{\beta e^{\beta(S_{ij} - \lambda)}}{1 + e^{\beta(S_{ij} - \lambda)}}, \qquad \boldsymbol{y}_j \neq \boldsymbol{y}_i \end{aligned} \tag{10}$$

As can be found, binomial deviance loss is a soft version of contrastive loss. In Eq. 3, a negative pair with a higher similarity is assigned with a larger weight, which means that it is more informative, by distinguishing two similar samples from different classes (which form a negative pair).

4. Multi-Similarity Loss

In this section, we first analyze three types of similarities that impact sample selection and weighting in deep metric learning. Then we propose a multi-similarity loss that jointly considers all three similarities with iterative sample mining and weighting.

4.1. Multiple Similarities

Though with diverse formulations, various pair-based loss functions, which commonly focus on learning from more informative pairs, can be cast into a pair weighting problem within our GPW framework. Furthermore, we observed that most pair-based approaches weight the pairs based on either self cosine similarities or relative similarities compared with other pairs. For simplicity, we take a negative pair as an example to describe three different types of similarities we defined. Analysis of a positive pair is similar. The three similarities for pair-based methods are described as follows.

S: Self-similarity. A self-similarity is computed from the pair itself, which is the most important similarity. A negative pair with a larger cosine similarity means that it is harder to distinguish two paired samples from different classes. Such pairs are referred as hard negative pairs, which are more informative and meaningful to learn discriminative features. Contrastive loss [6] and binomial deviance loss [40] are based on this criterion. As shown in case-1 of Fig. 2, the weights of three negative pairs are increased, when the negative samples come closer.

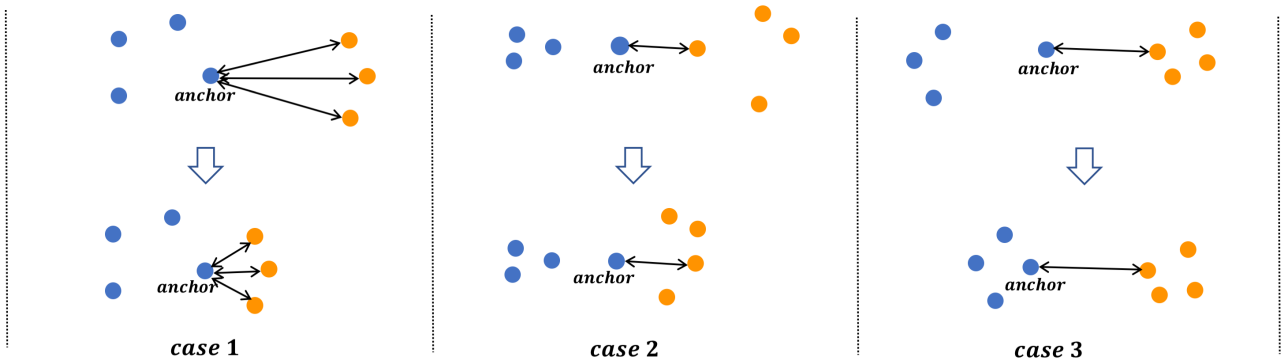

Figure 2. Violations of the three similarities for negative pairs. Case-1: The negative pairs' cosine similarities increase as they come closer to the anchor; case-2: Relative similarity compared with negatives decreases as the other negatives' cosine similarities increase; case-3: Relative similarity decrease as relevant positive pairs become closer.

	N-pairs	NCA	Histogram	contrastive	Triplet	LiftedStruct	Binomial	BinLifted	MS
S	✗	✗	✗	✓	✗	✗	✓	✓	✓
N	✓	✓	✗	✗	✗	✓	✗	✓	✓
P	✗	✓	✓	✗	✓	✗	✗	✗	✓

Table 1. This table shows similarities that these pair-based methods utilize to assign weight on one negative pair. In the first column of this table is the three perspectives 'S' : the **S**imilarity of itself; 'N': similarity compared with remaining **N**egative pairs shared the anchor point; 'P': similarity compared with **P**ositive pairs of the same anchor.

Obviously, self-similarity is difficult to fully describe sample distribution in an embedding space, and correlations of other pairs can make significant impact to similarity measurement. We introduce relative similarities by considering all pairs which have a same anchor with current pair, and define two type of relative similarities.

N: Negative relative similarity. It is computed by considering the relationship from neighboring negative pairs. As described in case-2 of Fig. 2, the relative similarity of a pair is decreased, even when its self-similarity is unchanged. This is because its neighboring negative samples move closer, which increases the self-similarities of these neighboring pairs, and thus reduce the relative similarity. Lifted structure loss [25] is based on this relative similarity, as shown in Eq. 8.

P: Positive relative similarity. Similarly, the relative similarity also considers the relationship from other positive pairs (with a same anchor). As shown in case-3 of Fig. 2, when these positive samples become closer to the anchor, the relative similarity of current pair is decreased, and thus the pair weight should be reduced accordingly. Triplet loss is computed based on this similarity, as indicated in Eq. 5.

With our GPW framework, we analyze a number of existing pair-based loss functions based on the three similarities defined, and compare them in Table 1. Detailed formulations of these functions are presented in Supplementary Material. As can be found, lifted structure loss only considers the negative relative similarity, by comparing with the negative neighboring pairs for weighting. The weight of current pair will be unchanged, when all positive samples (in case-1 of Fig. 2) or all negative samples (in case-3 of Fig. 2) move synchronously to the anchor. This conclusion can also be derived directly from Eq. 8, which only depends on $\sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{S_{ij}-S_{ik}}$. Obviously, such relevant samples (positive or negative) often include meaningful information, and are of great importance to learning discriminative features, but are discarded arbitrarily, which may reduce model capability considerably. While a weighting or sampling method based on each individual similarity has been explored previously, to the best of our knowledge, none of existing pair-based methods have assigned weights on pairs making full use of all the three similarities.

4.2. Multi-Similarity Loss

As discussed, unlike sampling or weighting schemes developed for classification and detection tasks in [2, 20], where the weight of an instance is computed individually based on a cross entropy loss, it is difficult to precisely measure the informativeness of a pair based on its individual cosine similarity. Pair-wise similarity between relevant samples or pairs should be considered, making the measurement and weighting problems more complicated and challenging.

As shown in Table 1, each of the listed approaches can consider one or two of the three similarities. To the best our knowledge, none of the existing pair-based methods can consider all of the three similarities simultaneously. To this end, we propose a Multi-Similarity (MS) loss, which consider all three perspectives by implementing a new pair weighting scheme using two iterative steps: mining and weighting. (i) informative pairs are first sampled by mea-

suring *Similarity-P*; and then (ii) the selected pairs are further weighted using *Similarity-S* and *Similarity-N* jointly. Details of the two steps are described as follows.

Pair mining. We first select informative pairs by computing *Similarity-P*, which measures the relative similarity between negative↔positive pairs having a same anchor. Specifically, a negative pair is compared to the *hardest* positive pair (with the lowest similarity), while a positive pair is sampled by comparing to a negative one having the largest similarity. Formally, assume $\boldsymbol{x}_i$ is an anchor, a negative pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ is selected if S_{ij} satisfies the condition:

$$S_{ij}^{-} > \min_{\boldsymbol{y}_k = \boldsymbol{y}_i} S_{ik} - \epsilon, \tag{11}$$

where ϵ is a given margin.

If $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$ is a positive pair, the condition is:

$$S_{ij}^{+} < \max_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} S_{ik} + \epsilon. \tag{12}$$

For an anchor $\boldsymbol{x}_i$, we denote the index set of its selected positive and negative pairs as $\mathcal{P}_i$ and $\mathcal{N}_i$ respectively. Our hard mining strategy is inspired by large margin nearest neighbor (LMNN) [35], a traditional distance learning approach which targets to seek an embedding space where neighboring positive points are encouraged to have the same class label with the anchor. The negative samples that satisfy Eq. 11 are approximately identical to *impostors* defined in LMNN [35].

Pair weighting. Pair mining with *Similarity-P* can roughly select informative pairs, and discard the less informative ones. We develop a soft weighting scheme that further weights the selected pairs more accurately, by considering both *Similarity-S* and *Similarity-N*. Our weighting mechanism is inspired by binomial deviance loss (considering *similarity-S*) and lifted structure loss (using *Similarity-N*). Specifically, given a selected negative pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in \mathcal{N}_i$, its weight w_{ij}^{-} can be computed as:

$$\begin{aligned} w_{ij}^{-} &= \frac{1}{e^{\beta(\lambda - S_{ij})} + \sum\limits_{k \in \mathcal{N}_i} e^{\beta(S_{ik} - S_{ij})}} \\ &= \frac{e^{\beta(S_{ij} - \lambda)}}{1 + \sum\limits_{k \in \mathcal{N}_i} e^{\beta(S_{ik} - \lambda)}}. \end{aligned} \tag{13}$$

and the weight w_{ij}^{+} of a positive pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in \mathcal{P}_i$ is:

$$w_{ij}^{+} = \frac{1}{e^{-\alpha(\lambda - S_{ij})} + \sum\limits_{k \in \mathcal{P}_i} e^{-\alpha(S_{ik} - S_{ij})}}, \tag{14}$$

where α, β, λ are hyper-parameters as in Binomial deviance loss (Eq. 9).

In Eq. 13, the weight of a negative pair is computed jointly from its self-similarity by $e^{\beta(\lambda - S_{ij})}$ - *Similarity-S*, and its relative similarity - *Similarity-N*, by comparing to its negative pairs. Similar rules are applied for computing the weight for a positive pair, as in Eq. 14. With these two considerations, our weighting scheme updates the weights of pairs dramatically to the violation of its self-similarity and relative similarities.

The weight computed by Eq. 13 and Eq. 14 can be considered as a combination of the weight formulations of binomial deviance loss and lifted structure loss. However, it is non-trivial to combine both functions in an elegant and suitable manner. We will compare our MS weighting with an average weighting scheme of binomial deviance (Eq. 10) and lifted structure (Eq. 8), denoted as *BinLifted*. We demonstrate by experiments that direct combination of them can not lead to performance improvement (as shown in ablation study).

Finally, we integrate pair mining and weighting scheme into a single framework, and provide a new pair-based loss function - multi-similarity (MS) loss, whose partial derivative with respect to S_{ij} is the weight defined in Eq. 13 and Eq. 14. Our MS loss is formulated as,

$$\begin{aligned} \mathcal{L}_{MS} = \frac{1}{m} \sum_{i=1}^{m} \Bigg\{ &\frac{1}{\alpha} \log \Big[1 + \sum_{k \in \mathcal{P}_i} e^{-\alpha(S_{ik} - \lambda)} \Big] \\ &+ \frac{1}{\beta} \log \Big[1 + \sum_{k \in \mathcal{N}_i} e^{\beta(S_{ik} - \lambda)} \Big] \Bigg\}. \end{aligned} \tag{15}$$

where $\mathcal{L}_{MS}$ can be minimized with gradient descent optimization, by simply implementing the proposed iterative pair mining and weighting.

5. Experiments

Our method was implemented by PyTorch. For network architecture, we used the Inception network with batch normalization [11] pre-trained on ILSVRC 2012-CLS [27], and fine-tuned it for our task. We add a FC layer on the top of the network following the global pooling layer. All the input images were cropped to 224×224. For data augmentation, we used random crop with random horizontal mirroring for training, and single center crop for testing. Adam optimizer was used for all experiments.

We conduct experiments on four standard datasets: CUB200 [33], Cars-196 [16], Stanford Online Products (SOP) [25] and In-Shop Clothes Retrieval (In-Shop) [21]. We follow the data split protocol applied in [25]. For the CUB200 dataset, we use the first 100 classes with 5,864 images for training, and the remaining 100 classes with 5,924 images for testing. The Cars-196 dataset is composed of 16,185 images of cars from 196 model categories. The first 98 model categories are used for training, with the rest for testing. For the SOP dataset, we use 11,318 classes for training, and 11,316 classes for testing. For the In-Shop

Recall@K (%)		1	2	4	8
Binomial	S	71.9	80.0	86.4	91.0
LiftedStruct*	N	69.7	79.3	86.2	91.0
MS mining	P	67.0	77.4	84.7	90.0
BinLifted	SN	70.4	79.5	86.2	91.1
MS weighting	SN	73.2	81.5	87.6	92.6
Binomial$_m$	SP	74.6	83.8	89.7	94.1
LiftedStruct$^*_m$	NP	72.2	81.7	88.0	92.4
MS	SNP	**77.3**	**85.3**	**90.5**	**94.2**

Table 2. This table shows Recall@K of methods on Cars-196. The first column contains the methods and the similarities they utilize to weight negative pairs. Embedding dimension is set to 64. The subscript m denotes employing our MS mining step.

dataset, 3,997 classes with 25,882 images are used for training. The test set is partitioned to a query set with 14,218 images of 3,985 classes, and a gallery set having 3,985 classes with 12,612 images.

For every mini-batch, we randomly choose a certain number of classes, and then randomly sample M instances from each class with $M = 5$ for all datasets in all experiments. ϵ in Eq. 11 and Eq. 12 is set to 0.1 and the hyper-parameters in Eq. 15 are: $\alpha = 2, \lambda = 1, \beta = 50$, by following [32]. Our method is evaluated on image retrieval task by using the standard performance metric: Recall@K.

5.1. Ablation Study

To demonstrate the importance of weighting the pairs from multi-similarities, we conduct an ablation study on Cars-196 and results are shown in Table 2. We describe LiftedStruct*, MS mining and MS weighting here, other methods have already been mentioned in section 3.

LiftedStruct*. Lifted structure loss is easy to get stuck in a local optima, resulting in poor performance. To evaluate the efficiency of three similarities more clearly and fairly, we make a minor modification to the lifted structure loss, allowing it to employ *Similarity-N* more effectively:

$$\mathcal{L}_{lift*} = \frac{1}{m}\sum_{i=1}^{m}\left\{\frac{1}{\alpha}\log\sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{-\alpha S_{ik}} + \frac{1}{\beta}\log\sum_{\boldsymbol{y}_k\neq\boldsymbol{y}_i} e^{\beta S_{ik}}\right\}, \tag{16}$$

where $\alpha = 2$, $\beta = 50$. This modification is motivated to make lift structure loss more focus on informative pairs, especially the hard negative pairs, and allows it to get rid of the side effect of enormous easy negative pairs. We found that this modification can boost the performance of lifted-structure loss empirically, e.g., with an over 20% improvement of Recall@1 on the CUB200 .

MS mining. To investigate the impact of each component of MS Loss, we evaluate the performance of MS mining individually, where the pairs selected into $\mathcal{N}_i$ and $\mathcal{P}_i$ are assigned with an equal weight.

MS weighting. Similarly, MS weighting scheme is also evaluated individually without the mining step in the ablation study, allowing us to analyze the effect of each similarity more perspicaciously. In MS weighting, each pair in a mini-batch is assigned with a non-zero weight, according to the weighting strategy described in Eq. 13 and Eq. 14.

With the performance reported in Table 2, we analyze the effect of each similarity as below:

Similarity-S: A cosine self-similarity is of the most importance. Binomial deviance loss, based on the *Similarity-S*, achieves the best performance by using a single similarity. Moreover, our MS weighting outperforms LiftedStruct$^*_m$ by $69.7\% \rightarrow 73.2\%$ at recall@1, and Binomial$_m$ also improves the recall@1 with $67.0\% \rightarrow 74.6\%$ over the MS mining, by adding the *Similarity-S* into their weighting schemes.

Similarity-N: Relative similarities are also helpful to measuring the importance of a pair more precisely. With *Similarity-N*, our MS weighting increases the Recall@1 by 1.3% over Binomial ($71.9\% \rightarrow 73.2\%$). Moreover, with *Similarity-N*, LiftedStruct$^*_m$ obtains a significant performance improvement over MS sampling ($67\% \rightarrow 72.2\%$), by considering both *Similarity-P* and *Similarity-N*.

Similarity-P: As shown in Table 2, by adding a mining step based on *Similarity-P*, the performances of LiftedStruct*, Binomial and MS weighting are consistently improved by a large margin. For instance, Recall@1 of Binomial is increased by nearly 3%: $71.9\% \rightarrow 74.6\%$.

Finally, the proposed MS loss achieves the best performance among these methods, by exploring multi-similarities for pair mining and weighting. However, it is critical to integrate the three similarities effectively into a single framework where the three similarities can be fully explored and optimized jointly by using a single loss function. For example, BinLifted, which uses a weighting scheme considering both *similarities-S* and *similarities-N*, has lower performance than that of single Binomial, since it implements a simple and straightforward combination of Binomial and LiftedStruct$^*_m$. More discussions on the difference between our MS weighting and the direct combination are presented in Supplementary Material.

5.2. On Embedding Size

By following [28], we study the performance of MS loss with varying embedding sizes $\{64, 128, 256, 512, 1024\}$. As shown in Fig. 3, the performance is increased consistently with the embedding dimension except at 1024. This is different from lifted structure loss, which achieves its best performance at 64 on the Cars-196 dataset [25].

5.3. Comparison with State-of-the-Art

We further compare the performance of our method with the state-of-the-art techniques on image retrieval task. As shown in Table 3, our MS loss improves Recall@1 by 7% on the CUB200, and 4% on the Cars-196 over Proxy-NCA

	CUB-200-2011						Cars-196					
Recall@K (%)	1	2	4	8	16	32	1	2	4	8	16	32
Clustering ^{64}[30]	48.2	61.4	71.8	81.9	-	-	58.1	70.6	80.3	87.8	-	-
ProxyNCA^{64} [24]	49.2	61.9	67.9	72.4	-	-	73.2	82.4	86.4	87.8	-	-
Smart Mining^{64} [7]	49.8	62.3	74.1	83.3	-	-	64.7	76.2	84.2	90.2	-	-
Margin^{128} [38]	63.6	74.4	83.1	90.0	94.2	-	79.6	86.5	91.9	95.1	97.3	-
HDC^{384} [30]	53.6	65.7	77.0	85.6	91.5	95.5	73.7	83.2	89.5	93.8	96.7	98.4
HTL^{512} [4]	57.1	68.8	78.7	86.5	92.5	95.5	81.4	88.0	92.7	95.7	97.4	**99.0**
ABIER^{512} [26]	57.5	68.7	78.3	86.2	91.9	95.5	82.0	89.0	93.2	96.1	97.8	98.7
ABE^{512} [14]	60.6	71.5	79.8	87.4	-	-	**85.2**	**90.5**	**94.0**	96.1	-	-
MS^{64}	57.4	69.8	80.0	87.8	93.2	96.4	77.3	85.3	90.5	94.2	96.9	98.2
MS^{512}	**65.7**	**77.0**	**86.3**	**91.2**	**95.0**	**97.3**	84.1	90.4	**94.0**	**96.5**	**98.0**	98.9

Table 3. Recall@K(%) performance on CUB200 and Cars-196. Superscript denotes embedding size. ABIER [26] and ABE [14] are ensemble methods.

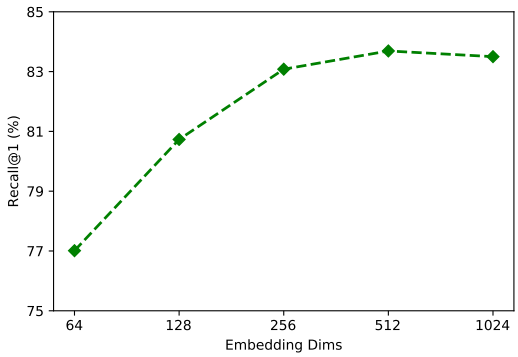

Figure 3. The effect of embedding size on MS loss

Recall@K (%)	1	10	20	30	40	50
FashionNet^{4096} [21]	53.0	73.0	76.0	77.0	79.0	80.0
HDC^{384} [30]	62.1	84.9	89.0	91.2	92.3	93.1
HTL^{128} [4]	80.9	94.3	95.8	97.2	97.4	97.8
ABIER^{512} [26]	83.1	95.1	96.9	97.5	97.8	98.0
ABE^{512} [14]	87.3	96.7	97.9	98.2	98.5	98.7
MS^{128}	88.0	97.2	98.1	98.5	98.7	98.8
MS^{512}	**89.7**	**97.9**	**98.5**	**98.8**	**99.1**	**99.2**

Table 4. Recall@K(%) performance on In-Shop.

Recall@K (%)	1	10	100	1000
Clustering ^{64}[30]	67.0	83.7	93.2	-
ProxyNCA^{64} [24]	73.7	-	-	-
Margin^{128} [38]	72.7	86.2	93.8	98.0
HDC^{384} [30]	69.5	84.4	92.8	97.7
HTL ^{512} [4]	74.8	88.3	94.8	98.4
ABIER^{512} [26]	74.2	86.9	94.0	97.8
ABE^{512} [14]	76.3	88.4	94.8	98.2
MS^{64}	74.1	87.8	94.7	98.2
MS^{128}	76.6	89.2	95.2	98.4
MS^{512}	**78.2**	**90.5**	**96.0**	**98.7**

Table 5. Recall@K(%) performance on SOP.

at dimension 64. Compared with ABE employing an embedding size of 512 and attention modules, our MS loss achieves a higher Recall@1 by +5% improvement at the same dimension on the CUB200. On the Cars-196, our MS loss obtains the second best performance in terms of Recall@1, while the best results are achieved by ABE, which applies an ensemble method with a much heavier model. Moreover, on the remaining three datasets, our MS loss is of much stronger performance than ABE.

In Table 4 and 5, our MS loss outperforms Proxy-NCA by 0.4% and Clustering by 7% at the same embedding size of 64. Furthermore, when compared with ABE, MS loss increases Recall@1 by 1.9% and 2.7% on the SOP and In-Shop respectively. Moreover, even with much compact embedding features of 128 dimension, our MS loss has already surpassed all existing methods, which utilize much higher dimensions of 384, 512 and 4096.

To summarize, on both fine-grained datasets like Cars-196 and CUB200, and large-scale datasets with enormous categories like SOP and In-Shop, our method achieves new state-of-the-art or comparable performance, even taking those methods with ensemble techniques like ABE and BIER into consideration.

6. Conclusion

We have presented a new multi-similarity loss for deep metric learning, and established a General Pair Weighting (GPW) framework which, for the first time, unify existing pair-based metric learning approaches into general pair weighting through gradient analysis. GPW provides a powerful tool for understanding and explaining various pair-based loss functions, which allows us to clearly identify the main differences and key limitations of existing methods. Furthermore, we proposed a multi-similarity loss which considers all three similarities simultaneously, and developed an iterative pair mining and weighting scheme for optimizing the multi-similarity loss efficiently. Our method obtains new state-of-the-art performance on a number of image retrieval benchmarks.

References

[1] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *ECCV*, 2016. 1

[2] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NIPS*, 2017. 3, 5

[3] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 2

[4] W. Ge, W. Huang, D. Dong, and M. R. Scott. Deep metric learning with hierarchical triplet loss. In *ECCV*, 2018. 1, 2, 3, 8

[5] A. Grabner, P. M. Roth, and V. Lepetit. 3d pose estimation and 3d model retrieval for objects in the wild. In *CVPR*, 2018. 1

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 1, 2, 3, 4

[7] B. Harwood, V. K. B G, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. In *ICCV*, 2017. 2, 3, 8

[8] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai. Triplet-center loss for multi-view 3d object retrieval. In *CVPR*, 2018. 1

[9] A. Hermans*, L. Beyer*, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv:1703.07737v4*, 2017. 1, 4

[10] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *SIMBAD*, 2015. 1, 2, 3, 4

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[12] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *NIPS*. 2014. 3

[13] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018. 3

[14] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon. Attention-based ensemble for deep metric learning. In *ECCV*, 2018. 1, 2, 8

[15] S. Kiran Yelamarthi, S. Krishna Reddy, A. Mishra, and A. Mittal. A zero-shot framework for sketch based image retrieval. In *ECCV*, 2018. 1

[16] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshops*, 2013. 6

[17] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 3

[18] M. T. Law, N. Thome, and M. Cord. Quadruplet-wise image similarity learning. In *ICCV*, 2013. 1

[19] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese cnn for robust target association. In *CVPR Workshops*, 2016. 1

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *ICCV*, 2017. 3, 5

[21] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 6, 8

[22] D. G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural computation*, 1995. 1

[23] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing*, 1999. 1

[24] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *ICCV*, 2017. 8

[25] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016. 1, 2, 3, 4, 5, 6, 7

[26] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep Metric Learning with BIER: Boosting Independent Embeddings Robustly. *PAMI*, 2018. 8

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 6

[28] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 2, 7

[29] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*. 2016. 1, 2

[30] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *CVPR*, 2017. 8

[31] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 1

[32] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*. 2016. 1, 2, 7

[33] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Master's thesis, 2011. 6

[34] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. *ICCV*, 2017. 1

[35] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*. 2006. 6

[36] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016. 1

[37] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *CVPR*, 2015. 1

[38] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. *ICCV*, 2017. 1, 2, 3, 8

[39] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, 2003. 1

[40] D. Yi, Z. Lei, and S. Z. Li. Deep metric learning for practical person re-identification. *arXiv:1407.4979*, 2014. 1, 2, 3, 4

[41] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai. Hard-aware point-to-set deep metric for person re-identification. In *ECCV*, 2018. 1

[42] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016. 1

Multi-Similarity Loss with GPW for Deep Metric Learning
Supplementary Material

1. Introduction

This supplementary material provides more details of General Pair Weighting (GPW) framework and Multi-Similarity (MS) loss. First, we further revisit three existing pair-based loss functions under our GPW framework. Then, we analyze the limitation of direct combination of binomial deviance loss [6] and lifted structure loss [1] (referred as BinLifted), compared to our weighting scheme. Finally, we analyze the impact of batch-size to the proposed MS loss with experiments conducted on CUB200 and SOP datasets.

2. Revisit Pair-based Loss Functions

In this section, we further analyze three recent pair-based loss functions in addition to those presented in the main paper: N-pairs loss [3], NCA loss [2] and histogram loss [4].

N-pairs Loss. Sohn *et al*. [3] proposed a N-pairs loss, which is a special case of the lifted structure loss by only considering single positive pair. It can be analyzed by following the same process of analyzing the lifted structure loss in the main paper, as shown in Eq. (6)-(8).

NCA Loss. Salakhutdinov *et al*. [2] introduced a NCA loss to learn a nonlinear embedding to optimize the classification performance of a soft-KNN classifier:

$$\begin{aligned}\mathcal{L}_{nca} &\coloneqq \sum_{i=1}^{m} log \frac{\sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{S_{ik}}}{\sum_{i=1}^{m} e^{S_{ik}}} \\ &= \sum_{i=1}^{m} \left[log \sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{S_{ik}} - log \sum_{i=1}^{m} e^{S_{ik}} \right].\end{aligned} \tag{1}$$

By following GPW framework, the weight of a pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$, *i.e.*, w_{ij}, can be derived from differentiating $\mathcal{L}_{nca}$ with respect to S_{ij}:

$$\begin{aligned}w_{ij}^{+} &= \frac{e^{S_{ij}}}{\sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{S_{ik}}} - \frac{e^{S_{ij}}}{\sum_{i=1}^{m} e^{S_{ik}}} \\ &= \frac{1}{\sum_{\boldsymbol{y}_k=\boldsymbol{y}_i} e^{S_{ik}-S_{ij}}} - \frac{1}{\sum_{i=1}^{m} e^{S_{ik}-S_{ij}}}, \\ w_{ij}^{-} &= \frac{e^{S_{ij}}}{\sum_{i=1}^{m} e^{S_{ik}}} = \frac{1}{\sum_{i=1}^{m} e^{S_{ik}-S_{ij}}}.\end{aligned} \tag{2}$$

As can be found in Eq. 2, the weight for a positive pair or a negative pair is calculated based on its relative similarities, which compute a similarity between a current pair and its neighboring pairs. The NCA loss focuses on hard negative pairs and the high-confident positive pairs, collected from the neighboring pairs of an anchor point, in the embedding space.

Histogram Loss. Ustinova *et al*. [4] designed a histogram loss based on quadruplets, which can be formulated as:

$$\begin{aligned}\mathcal{L}_{hist} &\coloneqq \sum_{r=1}^{R} (h_r^{-} \sum_{q=1}^{r} h_q^{+}) \\ &= \sum_{r=1}^{R} (\sum_{q=1}^{r} h_q^{+}) h_r^{-} \\ &= \sum_{q=1}^{R} (\sum_{r=q}^{R} h_r^{-}) h_q^{+}\end{aligned} \tag{3}$$

where R is the dimension of a histogram for positive or negative cosine similarities. h_q^{+} is the histogram estimation at node q of the cosine similarity for a positive pair, and h_r^{-} is that of a negative pair at node r.

$$h_r^{+} = \frac{1}{|\mathcal{S}^{+}|} \sum_{\boldsymbol{y}_i=\boldsymbol{y}_j} \delta_{ijr}, \tag{4}$$

where δ_{ijr} is defined as:

$$\delta_{ijr} = \begin{cases} (S_{ij} - t_{r-1})/\Delta, & S_{ij} \in [t_{r-1}, t_r], \\ (t_{r+1} - S_{ij})/\Delta, & S_{ij} \in [t_r, t_{r+1}], \\ 0, & otherwise, \end{cases} \tag{5}$$

where $\Delta = 2/(R-1)$, $t_r = r\Delta - 1$. Estimation of h_q^{-} is computed analogously.

To better understanding histogram loss within our GPW framework, we first provide the following formulations (more details can be found in [4]),

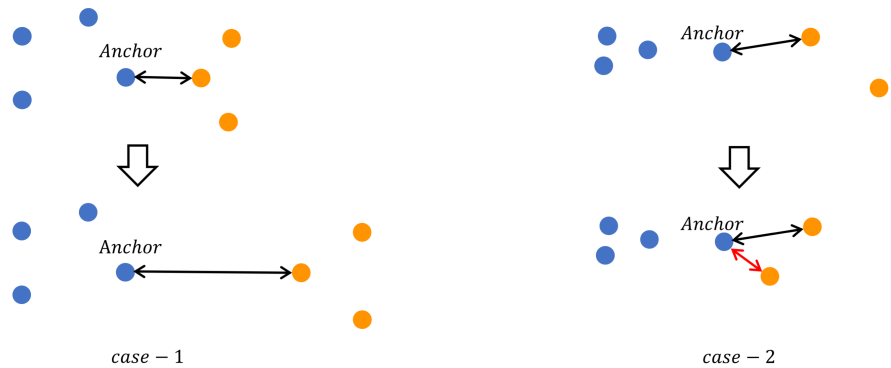

Figure 1. Limitations of the BinLifted weighting scheme: (i) in case-1, the negative pair on the bottom, which is of much lower cosine similarity than the top one, will be assigned with a larger weight; (ii) in case-2, a negative pair is fixed when the neighboring negative samples move closer to the anchor, which reduce the relative similarly (*Similarity-N*) of the negative pair.

$$\begin{aligned}\frac{\partial \mathcal{L}_{hist}}{\partial h_q^+} &= \sum_{r=q}^{R} h^- \\ \frac{\partial \mathcal{L}_{hist}}{\partial h_r^-} &= \sum_{q=1}^{r} h^+\end{aligned} \tag{6}$$

For a positive pair S_{ij}

$$\frac{\partial h_r^+}{\partial S_{ij}} = \begin{cases} \frac{+1}{\Delta|\mathcal{S}^+|}, & S_{ij} \in [t_{r-1}, t_r], \\ \frac{-1}{\Delta|\mathcal{S}^+|}, & S_{ij} \in [t_r, t_{r+1}], \\ 0, & otherwise, \end{cases} \tag{7}$$

where $|\mathcal{S}^+|$ is the number of positive pairs. Then we have

$$\frac{\partial h_r^-}{\partial S_{ij}} = 0, \tag{8}$$

since h_r^- is calculated from negative pairs, and thus is unrelated to the similarities of positive pairs, S_{ij}.

Finally, the partial derivative of $\mathcal{L}_{hist}$ w.r.t. the similarity of a positive pair $S_{ij} \in [t_p, t_{p+1}]$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}_{hist}}{\partial S_{ij}} &= \sum_{q=1}^{R} \frac{\partial \mathcal{L}_{hist}}{\partial h_q^+} \frac{\partial h_q^+}{\partial S_{ij}} + \sum_{r=1}^{R} \frac{\partial \mathcal{L}_{hist}}{\partial h_r^-} \frac{\partial h_r^-}{\partial S_{ij}} \\ &= \sum_{q=1}^{R} \left(\sum_{r=q}^{R} h_r^- \right) \frac{\partial h_q^+}{\partial S_{ij}} \\ &= \left(\sum_{r=p}^{R} h_r^- \right) \frac{\partial h_p^+}{\partial S_{ij}} + \left(\sum_{r=p+1}^{R} h_r^- \right) \frac{\partial h_{p+1}^+}{\partial S_{ij}} \\ &= \frac{1}{\Delta|\mathcal{S}^+|} \left(\sum_{r=p+1}^{R} h_r^- - \sum_{r=p}^{R} h_r^- \right) \\ &= \frac{-1}{\Delta|\mathcal{S}^+|} h_p^- .\end{aligned} \tag{9}$$

Thus the weight value assigned to this positive pair is $\frac{1}{\Delta|\mathcal{S}^+|} h_p^-$. Similarly, for a negative pair with a cosine similarity of $S_{ij} \in [t_p, t_{p+1}]$, the weight under a histogram loss is $\frac{1}{\Delta|\mathcal{S}^-|} h_{p+1}^+$.

Though with complicated formulation and rough derivation, pair weight scheme of histogram loss can be presented in an extremely concise and clear form, as shown in Eq. 9. h_p^- is approximately the ratio of negative pairs which have a lower cosine similarity compared with the current positive pair, and $\frac{1}{\Delta|\mathcal{S}^-|}$ can be regarded as a fixed normalizer. Similarly, the weight of a negative pair is the ratio of positive pairs having a lower cosine similarity than the current negative one. Therefore, the weighting scheme clearly indicates that a histogram loss estimates pair weights only based on *Similarity-P*, e.g., by comparing the negative pairs with the positive ones, and vise verse. This may reduce the performance of histogram loss, leading to lower performance than that of binomial deviance loss on the CUB200 and SOP, as reported in [4].

3. BinLifted v.s. MS Weighting

As discussed in the main paper, our MS loss is related to a direct combination of binomial deviance loss and lifted structure loss (referred as BinLifted), by jointly considering both *Similarity-S* and *Similarity-N*. In our ablation study presented in Section 5.1 (in main paper), we show by experiments that our weighting scheme is superior to that of BinLifted. Here we explain the benefits of our iterative mining and weighting scheme. Given an example of a negative pair $\{\boldsymbol{x}_i, \boldsymbol{x}_j\}$, the weight computed by Binlifted weighting scheme $\widehat{w}_{ij}$ is:

$$\widehat{w}_{ij} = \frac{1}{2} \left(\frac{e^{\beta(S_{ij}-\lambda)}}{1+e^{\beta(S_{ij}-\lambda)}} + \frac{e^{\beta S_{ij}}}{\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{\beta S_{ik}}} \right). \tag{10}$$

From Eq. 10, the weight of BinLifted satisfies:

$$\widehat{w}_{ij} > \frac{1}{2} \max \left(\frac{e^{\beta(S_{ij}-\lambda)}}{1+e^{\beta(S_{ij}-\lambda)}}, \frac{e^{\beta S_{ij}}}{\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{\beta S_{ik}}} \right). \tag{11}$$

which provides a lower bound for the weighting by BinLifted method. We found that a negative pair with a large relative similarity, will have a weight of $\frac{e^{\beta S_{ij}}}{\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{\beta S_{ik}}}$ which is close to 1, so that BinLifted method will assign a large weight to this pair, and ignore or reduce the impact from *Similarity-S*. As shown in case-1 of Fig. 1, a negative pair on the bottom has a much lower *Similarity-S* than the negative pair on the top, but BinLifted will compute a similar weight for the two pairs. Furthermore, when the *Similarity-S* of a pair is higher than a threshold λ, the weight will be computed as $\frac{e^{\beta(S_{ij}-\lambda)}}{1+e^{\beta(S_{ij}-\lambda)}}$, which can be a large value, regardless of *Similarity-N*. As illustrated in case-2 of Fig. 1, two negative pairs on the bottom will be assigned with a similar weight, due to large values of *Similarity-S*, while omitting the large difference on the relative similarities (*Similarity-N*) between the two pairs. However, a negative pair on the top of case-2 will also be assigned with a similar weight to the bottom one, even its *Similarity-S* is much larger.

In summary, BinLifted estimates the weight of a pair by mainly considering the larger value computed from *Similarity-S* or *Similarity-N*, while neglecting the smaller one. This drawback may reduce the performance considerably, which can be even lower than that of using a single binomial deviance loss, as shown in the ablation study.

In contrast, our MS weighting is able to compute the weight for a negative pair dynamically:

$$w_{ij} = \frac{e^{\beta(S_{ij}-\lambda)}}{1+\sum_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} e^{\beta(S_{ik}-\lambda)}}. \tag{12}$$

which addresses the issue raised in case-1 and case-2 of Fig. 1. This allows it to explore more meaningful information from both *Similarity-S* and *Similarity-S* (Eq. 12), rather than only focusing on the single one with the larger value. Positive pairs can be analyzed analogously.

4. Effect of Batch Size

To analyze the performance of our MS loss with different batch-size, we conduct experiments on the SOP [1] and CUB200 [5] datasets. We set the embedding size to 512 and $K = 5$, and use Adam optimizer with a learning rate of 10^{-5} for all experiments. The recall@1 performance of MS loss on the CUB200 and the SOP are reported in Tables 1 and 2, with batch-size set at $\{20, 40, 80, 160, 240\}$ and $\{20, 40, 80, 160, 320, 640, 1000\}$ respectively.

We observed that batch-size effects the performance of MS loss differently on the CUB200 and SOP: (i) the CUB200 is less sensitive to the change of batch-size than the SOP; (ii) the performance on the CUB200 decreases when the batch-size is increased, while the performance on the SOP benefits from large batch-sizes significantly.

We found that the batch-size impacts a dataset with large inter-class variations more significantly than the small ones. The CUB200 is a fine-grained dataset with smaller inter-class variations than the SOP, making it easier to collect hard negative pairs with subtle difference. However, by using a small batch-size of 20 on the SOP dataset, our MS mining is difficult to collect any informative pair at some iterations, which happened at over 20% iterations in our experiments. Therefore, a large batch-size, *e.g.*, > 320, is required for a dataset with large variations like the SOP. This ensures it to collect enough hard negative pairs, which are meaningful to train a discriminative model, and thus improve the performance.

Batch-size	Recall@1 (%)
20	64.75
40	65.07
80	**65.65**
160	65.43
240	64.60

Table 1. Recall@1 performance of MS loss at the batch-size of $\{20, 40, 80, 120, 160, 240\}$ on the CUB200 database.

Batch-size	Recall@1 (%)
20	71.40
40	73.82
80	75.61
160	76.63
320	77.59
640	78.19
1000	**78.35**

Table 2. Recall@1 performance of MS loss at the batch-size of $\{20, 40, 80, 160, 320, 640\}$ on the SOP database.

References

[1] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016. 1, 3

[2] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, 2007. 1

[3] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*. 2016. 1

[4] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*. 2016. 1, 2

[5] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Master's thesis, 2011. 3

[6] D. Yi, Z. Lei, and S. Z. Li. Deep metric learning for practical person re-identification. *arXiv:1407.4979*, 2014. 1