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

Xun Wang, Xintong Han, Weiling 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% → 65.7% on CUB200, and 80.9% → 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].

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. 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 schemes for pair-based metric learning techniques. For example,
ample, 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
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 $x_i \in \mathbb{R}^d$ be a real-value instance vector. Then we have an instance matrix $X \in \mathbb{R}^{m \times d}$, and a label vector $y \in \{1, 2, \ldots, C\}^m$ for the $m$ training samples respectively. Then an instance $x_i$ is projected to a unit sphere in a $l$-dimension space by $f(\cdot; \theta) : \mathbb{R}^d \rightarrow S^l$, where $f$ is a neural network parameterized by $\theta$. Formally, we define the similarity of two samples as $S_{ij} := f(x_i; \theta), f(x_j; \theta)$, where $< \cdot, \cdot >$ denotes dot product, resulting in an $m \times m$ similarity matrix $S$ 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 $S$ and $y$: $\mathcal{L}(S, y)$. The derivative with respect to model parameters $\theta$ at the $t$-th iteration can be calculated as:

$$\frac{\partial \mathcal{L}(S, y)}{\partial \theta} l_t = \frac{\partial \mathcal{L}(S, y)}{\partial S} l_t \frac{\partial S}{\partial \theta} l_t = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t \frac{\partial S_{ij}}{\partial \theta} l_t. \quad (1)$$

Eq. 1 is computed for optimizing model parameters $\theta$ 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. $\theta$ at the $t$-th iteration is computed exactly the same as Eq. 1. $\mathcal{F}$ is formulated as below:

$$\mathcal{F}(S, y) = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t S_{ij}. \quad (2)$$

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

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 $\frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t \geq 0$ for a negative pair, and $\frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t \leq 0$ for a positive pair. Thus, $\mathcal{F}$ in Eq. 2 can be transformed into the form of pair weighting as follows:

$$\mathcal{F} = \sum_{i=1}^{m} \left( \sum_{y_i \neq y_j} \frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t S_{ij} + \sum_{y_j = y_i} \frac{\partial \mathcal{L}(S, y)}{\partial S_{ij}} l_t S_{ij} \right) \quad (3)$$

$$= \sum_{i=1}^{m} \left( \sum_{y_i \neq y_j} w_{ij} S_{ij} - \sum_{y_j = y_i} w_{ij} S_{ij} \right),$$

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

As indicated in Eq. 3, a pair-based method can be formulated as weighting of pair-wise similarities, where the weight for pair $(x_i, x_j)$ is $w_{ij}$. Learning with a pair-based loss function $\mathcal{L}$ is now transformed from Eq. 1 into computing the weight of pairs in Eq. 3. It is a general pair weighting (GPW) formulation, and sampling is only one of its 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, $\lambda$:

$$\mathcal{L}_{\text{contrast}} = (1 - I_{ij})(S_{ij} - \lambda)_+ - I_{ij}S_{ij}, \quad (4)$$

where $I_{ij} = 1$ if $y_i = y_j$, and 0 otherwise.
where \( I_{ij} = 1 \) 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 \( \lambda \):

\[
L_{triplet} := [S_{an} - S_{ap} + \lambda]_+,
\]

where \( S_{an} \) and \( S_{ap} \) denote the similarity of a negative pair \( \{x_a, x_n\} \) and a positive pair \( \{x_a, x_p\} \), with an anchor sample, \( x_a \). 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 \( S_{an} + \lambda \leq S_{ap} \) 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:

\[
L_{lifted} := \sum_{i=1}^{m} \left[ \log \sum_{y_k = y_i} e^{\lambda - S_{ik}} + \log \sum_{y_k \neq y_i} e^{S_{ik}} \right]_+.
\]

where \( \lambda \) is a fixed margin.

In Eq. 6, when the hinge function of an anchor \( x_i \) returns a non-zero value, we can have a weight value, \( w_{ij}^+ \), for the pair \( \{x_i, x_j\} \), by differentiating \( L_{lifted} \) on \( S_{ij} \), according to Eq. 3. Then the weight for a positive pair is computed as:

\[
w_{ij}^+ = \frac{e^{\lambda - S_{ij}}}{\sum_{y_k = y_i} e^{\lambda - S_{ik}}} = \frac{1}{\sum_{y_k = y_i} e^{S_{ij} - S_{ik}}},
\]

and the weight for a negative pair is:

\[
w_{ij}^- = \frac{e^{S_{ij}}}{\sum_{y_k \neq y_i} e^{S_{ik}}} = \frac{1}{\sum_{y_k \neq y_i} e^{S_{ik} - S_{ij}}}.
\]

Eq. 7 shows that the weight for a positive pair is determined by its relative similarity, measured by comparing it to the remaining 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:

\[
L_{binomial} = \frac{1}{N_i} \sum_{y_j \neq y_i} \log \left[ 1 + e^{\beta(S_{ij} - \lambda)} \right] + \frac{1}{P_i} \sum_{y_j = y_i} \log \left[ 1 + e^{\alpha(S_{ij} - \lambda)} \right],
\]

where \( P_i \) and \( N_i \) denote the numbers of positive pairs and negative pairs with anchor \( x_i \), respectively. \( \lambda, \alpha, \beta \) are fixed hyper-parameters.

The weight for pair \( \{x_i, x_j\} \) is \( w_{ij} \) in Eq. 1, which can be derived from differentiating \( L_{binomial} \) on \( S_{ij} \) as:

\[
w_{ij}^+ = \frac{1}{P_i} \frac{\alpha e^{\alpha(\lambda - S_{ij})}}{1 + e^{\alpha(\lambda - S_{ij})}}, \quad y_j = y_i
\]

\[
w_{ij}^- = \frac{1}{N_i} \frac{\beta e^{\beta(S_{ij} - \lambda)}}{1 + e^{\beta(S_{ij} - \lambda)}}, \quad y_j \neq y_i
\]

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.

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 similarity of itself; ‘N’: similarity compared with remaining negative pairs shared the anchor point; ‘P’: similarity compared with positive pairs of the same anchor.

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

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 \( \sum_{y_k=y_i} e^{S_{ij} - S_{ik}} \). 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 of 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 \( x_i \) is an anchor, a negative pair \( \{ x_i, x_j \} \) is selected if \( S_{ij} \) satisfies the condition:

\[
S_{ij}^- > \min_{y_k=y_i} S_{ik} - \epsilon, \tag{11}
\]

where \( \epsilon \) is a given margin.

If \( \{ x_i, x_j \} \) is a positive pair, the condition is:

\[
S_{ij}^+ < \max_{y_k\neq y_i} S_{ik} + \epsilon. \tag{12}
\]

For an anchor \( x_i \), we denote the index set of its selected positive and negative pairs as \( P_i \) and \( N_i \) 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 \( \{ x_i, x_j \} \in N_i \), its weight \( w_{ij}^- \) can be computed as:

\[
w_{ij}^- = \frac{1}{e^{\beta(S_i-S_j)} + \sum_{k\in N_i} e^{\beta(S_{ik}-S_{ij})}}\tag{13}
\]

and the weight \( w_{ij}^+ \) of a positive pair \( \{ x_i, x_j \} \in P_i \) is:

\[
w_{ij}^+ = \frac{1}{e^{-\alpha(S_i-S_j)} + \sum_{k\in P_i} e^{-\alpha(S_{ik}-S_{ij})}}\tag{14}
\]

where \( \alpha, \beta, \lambda \) 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 \( e^{\beta(S_i-S_j)} \) - 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,

\[
\mathcal{L}_{MS} = \frac{1}{m} \sum_{i=1}^{m} \left\{ \frac{1}{\alpha} \log \left[ 1 + \sum_{k\in P_i} e^{-\alpha(S_{ik}-\lambda)} \right] + \frac{1}{\beta} \log \left[ 1 + \sum_{k\in N_i} e^{\beta(S_{ik}-\lambda)} \right] \right\}, \tag{15}
\]

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 \( 224 \times 224 \). 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
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. $\epsilon$ 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:

$$
L_{lfs} = \frac{1}{m} \sum_{i=1}^{m} \left\{ \frac{1}{\alpha} \log \sum_{y_k=y_i} e^{-\alpha S_{ik}} + \frac{1}{\beta} \log \sum_{y_k \neq y_i} e^{\beta S_{ik}} \right\},
$$

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 perspicuously. 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* by 69.7% $\rightarrow$ 73.2% at recall@1, and Binomialm 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, LiftedStructm 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 LiftedStructm. 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.
Table 3. Recall@K(%) performance on CUB200 and Cars-196. Superscript denotes embedding size. ABIER [26] and ABE [14] are ensemble methods.

| Embedding Dims | CUB-200-2011 | Cars-196 |
|----------------|-------------|---------|
|                | 1 2 4 8 16 32 | 1 2 4 8 16 32 |
| Clustering     | 58.1 70.6 80.3 87.8 - - | 5.1 7.2 8.4 8.7 9.0 - - |
| ProxyNCA       | 73.2 82.4 86.4 87.8 - - | 73.2 82.4 86.4 87.8 - - |
| Smart Mining   | 64.7 76.2 84.2 90.2 - - | 64.7 76.2 84.2 90.2 - - |
| Margin         | 79.6 86.5 91.9 95.1 97.3 - | 98.6 97.6 93.8 96.7 98.4 - |
| HDC            | 91.5 95.5 97.4 99.7 - - | 91.5 95.5 97.4 99.7 - - |
| HTL            | 92.5 97.5 97.8 97.8 - - | 92.5 97.5 97.8 97.8 - - |
| ABIER [26]     | 82.0 89.0 93.2 96.1 98.7 - | 82.0 89.0 93.2 96.1 98.7 - |
| ABE [14]       | 85.2 90.5 94.0 96.1 - - | 85.2 90.5 94.0 96.1 - - |
| MS [64]        | 77.3 85.3 90.5 94.2 - - | 77.3 85.3 90.5 94.2 - - |
| MS [512]       | 80.5 90.8 94.0 96.5 98.0 98.9 - | 80.5 90.8 94.0 96.5 98.0 98.9 - |

Figure 3. The effect of embedding size on MS loss

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

| Embedding Dims | 1 10 20 30 40 50 |
|----------------|-----------------|
| FashionNet    | 53.0 73.0 76.0 77.0 79.0 80.0 |
| HDC [30]      | 62.1 84.9 89.0 91.2 92.3 93.1 |
| HTL [4]       | 80.9 94.3 95.8 97.2 97.4 97.8 |
| ABIER [26]    | 83.1 95.1 96.9 97.5 97.8 98.0 |
| ABE [14]      | 87.3 96.7 97.9 98.2 98.5 98.7 |
| MS [256]      | 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 5. Recall@K(%) performance on SOP.

| Embedding Dims | 1 10 100 1000 |
|----------------|--------------|
| Clustering    | 67.0 83.7 93.2 - |
| ProxyNCA [24] | 73.7 - - - |
| Margin [38]   | 72.7 86.2 93.8 98.0 |
| HDC [30]      | 69.5 84.4 92.8 97.7 |
| HTL [4]       | 74.8 88.3 94.8 98.4 |
| ABE [26]      | 74.2 86.9 94.0 97.8 |
| ABE [14]      | 76.3 88.4 94.8 98.2 |
| MS [512]      | 78.2 90.5 96.0 98.7 |

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.
