Ranked List Loss for Deep Metric Learning

Xinshao Wang[1,2], Yang Hua[1], Elyor Kodirov[2], Guosheng Hu[2,1], Romain Garnier[2], Neil M. Robertson[1,2]
[1] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK
[2] Anyvision Research Team, UK
{xwang39, y.hua, n.robertson}@qub.ac.uk, {elyor, guosheng.hu, romaing}@anyvision.co

Abstract

The objective of deep metric learning (DML) is to learn embeddings that can capture semantic similarity information among data points. Existing pairwise or tripletwise loss functions used in DML are known to suffer from slow convergence due to a large proportion of trivial pairs or triplets as the model improves. To improve this, ranking-motivated structured losses are proposed recently to incorporate multiple examples and exploit the structured information among them. They converge faster and achieve state-of-the-art performance. In this work, we present two limitations of existing ranking-motivated structured losses and propose a novel ranked list loss to solve both of them. First, given a query, only a fraction of data points is incorporated to build the similarity structure. Consequently, some useful examples are ignored and the structure is less informative. To address this, we propose to build a set-based similarity structure by exploiting all instances in the gallery. The samples are split into a positive set and a negative set. Our objective is to make the query closer to the positive set than to the negative set by a margin. Second, previous methods aim to pull positive pairs as close as possible in the embedding space. As a result, the intraclass data distribution might be dropped. In contrast, we propose to learn a hypersphere for each class in order to preserve the similarity structure inside it. Our extensive experiments show that the proposed method achieves state-of-the-art performance on three widely used benchmarks.

1. Introduction

Deep metric learning (DML) plays a crucial role in a variety of applications in computer vision, such as image retrieval [28, 19], clustering [10], and transfer learning [20]. In addition, DML is a good solution for challenging extreme classification settings [22, 40], in which there exist an enormous number of classes and only a few images per class. For example, by using DML, FaceNet [24] achieves superhuman performance on face verification with 260M face images of 8M identities.

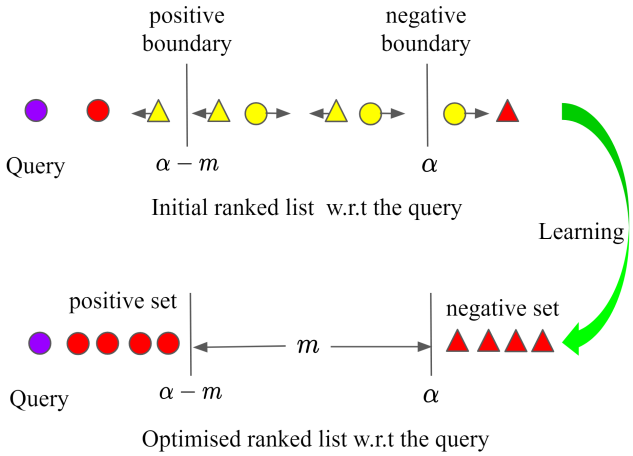


Figure 1: Illustration of our proposed RLL. Given a query and its ranked list, RLL aims to make the query closer to the positive set than to the negative set by a margin m. Circle and triangle represent two different classes. The blue circle is a query. The yellow shapes represent nontrivial examples while the red shapes represent trivial examples. The arrow indicates the query's gradient direction determined by the corresponding non-trivial examples. The final gradient direction of the query is *a weighted combination of them*. The optimised ranked list is shown in the bottom.

Loss function plays a key role in successful DML frameworks and a large variety of loss functions have been proposed in the literature. Contrastive loss [2, 6] captures the relationship between pairwise data points, i.e., similarity or dissimilarity. Triplet-based losses are also widely studied [24, 33, 3]. A triplet is composed of an anchor point, a similar (positive) data point and dissimilar (negative) data point. The purpose of triplet loss is to learn a distance metric by which the anchor point is closer to the similar point than the dissimilar one by a margin. In general, the triplet loss outperforms the contrastive loss [20, 24] because the relationship between positive and negative pairs is considered. Inspired by this, recent work [24, 28, 20, 29, 16, 19] proposes to take into consideration the richer structured information among multiple data points and achieve impressive

performance on many applications, e.g., image retrieval and clustering.

However, there are still certain limitations in current state-of-the-art DML approaches. Firstly, we notice that *only a proportion of informative examples* is incorporated to capture the structure in previous ranking-motivated loss functions. In this case, some non-trivial examples are wasted and the structured information is extracted from fewer data points. To address it, we propose to *utilise all non-trivial data points* to build a more informative structure and exploit it to learn more discriminative embeddings. Specifically, given a query, we obtain a ranked list by sorting all other data points (gallery) according to the similarities. Ideally, all the positive examples are supposed to be ranked before the negative samples in the feature space. To achieve this, we propose ranked list loss (RLL) to organise the samples of each query. Given a query, the optimisation of RLL is to rank all positive points before the negative points and forcing a margin between them. In other words, RLL aims to explore the set-based similarity structure, which contains richer information than the point-based approach, e.g., triplet loss.

Secondly, we observe that the intraclass data distribution is not considered in the previous structured losses. All algorithms [24, 20, 28, 29, 19] target to pull data points in the same class as close as possible. Consequently, these approaches try to *shrink samples of the same class into one point* in the feature space and may easily drop their similarity structure. To address this, we propose to learn a hypersphere for each class in RLL. Specifically, instead of pulling intraclass examples as compact as possible, we only force the distance of a positive pair smaller than a threshold, which is the diameter of each class's hypersphere. In this case, RLL can help preserve the similarity structure inside each class as much as possible.

Empirically, the convergence rate of DML methods highly depends on the possibility of seeing non-trivial samples [24]. Given an anchor (query), it is non-trivial to *separate the positive and negative sets by a margin* when all data points are considered. As a result, only a few ranked lists are perfectly optimized as the model improves during training. Therefore, our method can take advantage of a maximum of elements with non-zero losses and release the potentials for the learning procedure. The proposed RLL is illustrated in Figure 1.

Our contributions in this paper are listed as follows:

- We propose a novel ranking-motivated structured loss (RLL) to learn discriminative embeddings. In contrast with previous ranking-motivated losses, we are the first to incorporate all non-trivial data points and exploit the structure among them. Besides, we learn a hypersphere for each class to preserve intraclass data distribution instead of shrinking each class into one point in the embedding space.
- We achieve new state-of-the-art performance on three popular benchmarks, i.e., CARS196 [15], CUB-200-2011 [32] and SOP [20].

2. Preliminaries

Notations. Let $\mathbf{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be the input data, where $(\mathbf{x}_i, y_i)$ indicates i-th image and its corresponding class label. The total number of classes is C, i.e., $y_i \in [1, 2, ..., C]$. The images from c-th class are represented as $\{\mathbf{x}_i^c\}_{i=1}^{N_c}$, where N_c is the number of images in c-th class.

2.1. Structured Losses

2.1.1 Ranking-Motivated Structured Losses

Triplet Loss [37, 24] aims to pull the anchor point closer to the positive point than to the negative point by a fixed margin m:

$$L(\mathbf{X}; f) = \frac{1}{|\mathbf{\Gamma}|} \sum_{(i,j,k)\in\mathbf{\Gamma}} [d_{ij}^2 + m - d_{ik}^2]_+, \qquad (1)$$

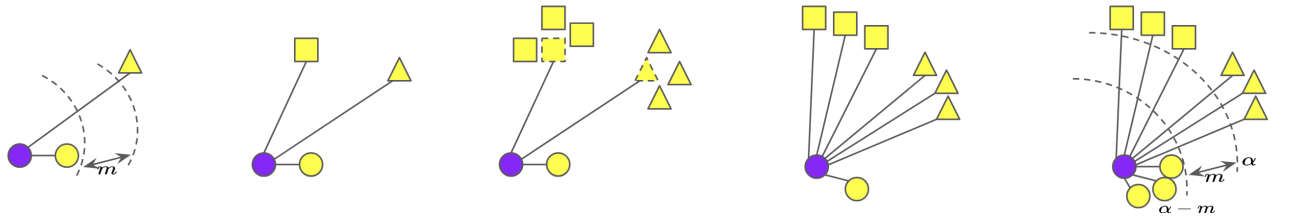
where $\mathbf{\Gamma}$ is the set of triplets, i, j and k are the indexes of anchor, positive and negative points, respectively. f is the embedding function, $d_{ij} = ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2$ is the Euclidean distance. $[\cdot]_+$ is the hinge function.

N-pair-mc [28] exploits the structured relationship among multiple data points to learn the embedding function. Triplet loss pulls one positive point while pushing a negative one simultaneously. To improve the triplet loss by interacting with more negative classes and examples, N-pair-mc aims to *identify one positive example from* $N-1$ *negative examples of* $N-1$ *classes* (one negative example per class):

$$L(\{(\mathbf{x}_i, \mathbf{x}_i^+)\}_{i=1}^N; f) = \frac{1}{N} \sum_{i=1}^N log\{1+ \sum_{j\neq i} \exp(\mathbf{f}_i^\top \mathbf{f}_j^+ - \mathbf{f}_i^\top \mathbf{f}_i^+)\}, \qquad (2)$$

where $\mathbf{f}_i = f(\mathbf{x}_i)$ and $\{(\mathbf{x}_i, \mathbf{x}_i^+)\}_{i=1}^N$ are N pairs of examples from N different classes, i.e., $y_i \neq y_j, \forall i \neq j$. Here, $\mathbf{x}_i$ and $\mathbf{x}_i^+$ are the query and the positive example respectively. $\{\mathbf{x}_j^+, j \neq i\}$ are the negative examples.

Lifted Struct [20] is proposed by Song *et al.* to learn the embedding function by *incorporating all negative examples*. The objective of Lifted Struct is to pull one positive pair $(\mathbf{x}_i^+, \mathbf{x}_j^+)$ as close as possible and pushes all negative data points corresponding to $\mathbf{x}_i^+$ or $\mathbf{x}_j^+$ farther than a margin



Triplet *N*-pair-mc Proxy NCA Lifted Struct Ranked List Loss

Figure 2: Illustration of different ranking-motivated structured losses. Different shapes (circle, triangle and square) represent different classes. For simplicity, only 3 classes are shown. The blue circle is an anchor (query). In triplet [24], the anchor is compared with only one negative example and one positive example. In N-pair-mc [28], Proxy-NCA [19] and Lifted Struct [20], one positive example and multiple negative classes are incorporated. N-pair-mc randomly selects one example per negative class. Proxy NCA pushes the anchor away from negative proxies instead of negative examples. The proxy is class-level and can represent any instance in the corresponding class. Lifted Struct uses all examples from all negative classes. On the contrary, our proposed ranked list loss not only exploits all negative examples, but also makes use of all positive ones.

α. Mathematically:

$$L(\mathbf{X};f) = \frac{1}{2|\mathbf{P}|}\sum_{(i,j)\in\mathbf{P}}[\{d_{ij} + \log(\sum_{(i,k)\in\mathbf{N}}\exp(\alpha - d_{ik}) + \sum_{(j,l)\in\mathbf{N}}\exp(\alpha - d_{jl}))\}]_+, \tag{3}$$

where $\mathbf{P}$ and $\mathbf{N}$ respectively represent the sets of positive pairs and negative pairs. Given the query $\mathbf{x}_i$, Lifted Struct intends to *identify one positive example from all corresponding negative data points*.

Proxy-NCA [19] is proposed to address the sampling problem using proxies. The proxy $\mathbf{W}$ is a small set of data points that represent training classes in the original data. The proxy for $\mathbf{u}$ is chosen by:

$$p(\mathbf{u}) = \text{argmin}_{\mathbf{w}\in\mathbf{W}} d(\mathbf{u},\mathbf{w}), \tag{4}$$

$p(\mathbf{u})$ denotes the closest point to $\mathbf{u}$ from $\mathbf{W}$. The Proxy-NCA loss is the traditional NCA loss defined over proxies instead of the original data points:

$$L(\mathbf{a},\mathbf{u},\mathbf{Z}) = -log(\frac{\exp(-d(\mathbf{a},p(\mathbf{u})))}{\sum_{\mathbf{z}\in\mathbf{Z}}\exp(-d(\mathbf{a},p(\mathbf{z})))}), \tag{5}$$

where $\mathbf{Z}$ is the negative set, $p(\mathbf{u})$ and $p(\mathbf{z})$ are the proxies of positive and negative points, respectively. $\mathbf{a}$ is the anchor and $d(\cdot,\cdot)$ is the Euclidean distance between two points. With static proxy assignment, i.e., *one proxy per class*, the performance is much better than dynamic proxy assignment. However, the proxies in the static proxy assignment are learned during training and similar to the class vectors of the fully connected layer in classification. Therefore, *Proxy-NCA does not preserve the scalability of DML* as the number of classes needs to be considered.

The proposed RLL is ranking-motivated structured loss, which avoids two limitations of traditional methods by incorporating all non-trivial data points and exploring intrinsic structured information among them. The illustration and comparison of different ranking-motivated losses and our method is presented in Figure 2.

2.1.2 Clustering-Motivated Structured Losses

Struct Clust [29] is recently proposed to learn the embedding function f by optimising the clustering quality metric. The proposed structured loss function is defined as:

$$L(\mathbf{X};f) = [F(\mathbf{X},\hat{\mathbf{y}};f) + \gamma\triangle(\mathbf{y},\hat{\mathbf{y}})) - F(\mathbf{X},\mathbf{y};f)]_+, \tag{6}$$

$$\triangle(\mathbf{y},\hat{\mathbf{y}}) = 1 - \text{NMI}(\mathbf{y},\hat{\mathbf{y}}), \tag{7}$$

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are the predicted and ground-truth clustering assignments respectively. F measures the quality of the clustering on $\mathbf{X}$ with the label assignment and distance metric. $\text{NMI}(\mathbf{y},\hat{\mathbf{y}})$ is the normalised mutual information [25]. NMI is 1 if the predicted clustering assignment is as good as the ground-truth and 0 if it is the worst. $\hat{\mathbf{y}}$ is predicted based on the learned distance metric f and Struct Clust [29] aims to learn f such that the F of the ground-truth assignment is larger than any other predicted clustering assignment.

However, *this algorithm is NP-hard* as we need to optimise both the clustering medoids and the distance metric simultaneously. As a result, the loss augmented inference and refinement are applied to select facilities (clustering medoids) based on the greedy algorithm [18]. *Large enough greedy search iterations are needed to find a local optimum*, which might be costly.

Spectral Clust [16] also aims to optimise the quality of the clustering. Spectral Clust relaxes the problem of clustering with Bregman divergences [1] and computes the gradient in a closed-form, which reduces the algorithmic complexity of existing iterative methods, e.g., Struct Clust [29]. However, it is still non-trivial to learn deep models based

on mini-batch implementation. Large batch size (1260 = 18 classes x 70 per class) is required for the clustering in the mini-batch. As a result, Spectral Clust iteratively computes submatrices and concatenates them into a single matrix for computing the loss and gradient, which might be expensive.

Both ranking-motivated and clustering-motivated structured loss functions exploit the structured similarity information among multiple data points. However, in general, clustering-motivated losses are more difficult to optimise than ranking-motivated losses.

2.2. Mining Non-trivial Examples

Example mining strategies are widely applied in existing methods [24, 35, 27, 11, 41, 26, 3, 20, 28, 36] to provide non-trivial examples for faster convergence and better performance. In FaceNet [24], they propose to mine semi-hard negative samples. In N-pair-mc [28], hard negative class mining is proposed to provide informative negative examples. In Lifted Struct [20], harder negative examples are emphasized. In our work, we simply mine examples which have non-zero losses.

3. Methodology

Our objective is to learn a discriminative function f (a.k.a. deep metric) such that the similarity between positive pairs is higher than the similarity between negative pairs in the feature space. There exist at least two images in each class so that all classes can be evaluated. In this case, given a query from any class, we aim to identify its matching samples from all other examples.

3.1. Pairwise Constraint

Inspired by the former work on pairwise similarity constraint [6, 38], we aim to pull positive examples closer than a predefined threshold (boundary). In addition, we intend to separate the positive and negative sets by a margin m. To achieve this, we choose the pairwise margin loss [38] as our basic pairwise constraint to construct the set-based similarity structure.

Given an image $\mathbf{x}_i$, we aim to push its negative point farther than a boundary α and pull its positive one closer than another boundary $\alpha - m$. Thus m is the margin between two boundaries. Mathematically,

$$L_{\mathrm{m}}(\mathbf{x}_i, \mathbf{x}_j; f) = (1-y_{ij})[\alpha - d_{ij}]_+ + y_{ij}[d_{ij} - (\alpha - m)]_+, \tag{8}$$

where $y_{ij} = 1$ if $y_i = y_j$, and $y_{ij} = 0$ otherwise. $d_{ij} = ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2$ is the Euclidean distance between two points.

3.2. Ranked List Loss

Given a query $\mathbf{x}_i^c$, we rank all other data points (gallery) according to their similarities to the query, which is illustrated in Figure 1. In each ranked list, there are $N_c - 1$ positive points in the positive set and $\sum_{k \neq c} N_k$ points in the negative set. The positive set with respect to the query $\mathbf{x}_i^c$ is denoted as $\mathbf{P}_{c,i} = \{\mathbf{x}_j^c | j \neq i\}, |\mathbf{P}_{c,i}| = N_c - 1$. Similarly, we represent the negative set with respect to $\mathbf{x}_i^c$ as $\mathbf{N}_{c,i} = \{\mathbf{x}_j^k | k \neq c\}, |\mathbf{N}_{c,i}| = \sum_{k \neq c} N_k$.

Non-trivial Sample Mining. Mining informative examples is widely adopted [24, 41, 3, 20, 28, 36, 9] because it allows fast convergence and good performance. By informative examples, we mean non-trivial data points which have non-zero losses, i.e., violating the pairwise constraint with respect to the query. Since they have zero gradients, including them for training will 'weaken' the contribution of non-trivial examples during gradient fusion as the model improves [9].

We mine both non-trivial positive and negative examples. For the query $\mathbf{x}_i^c$, the non-trivial positive set after mining is represented as $\mathbf{P}_{c,i}^* = \{\mathbf{x}_j^c | j \neq i, d_{ij} > (\alpha - m)\}$. Similarly, we denote the negative set after mining as $\mathbf{N}_{c,i}^* = \{\mathbf{x}_j^k | k \neq c, d_{ij} < \alpha\}$.

Loss-based Negative Examples Weighting. With respect to each query $\mathbf{x}_i^c$, there are a large number of non-trivial negative examples ($\mathbf{N}_{c,i}^*$) with different magnitude of losses. To make better use of them, we propose to weight the negative examples based on their loss values, i.e., how much each negative pair violates the constraint. The weighting strategy is formally represented as:

$$w_{ij} = \exp(T \cdot (\alpha - d_{ij})), \mathbf{x}_j^k \in \mathbf{N}_{c,i}^*. \tag{9}$$

We notice that the gradient magnitude with respect to any embedding is always one in Eq. (8). Mathematically,

$$||\frac{\partial L_{\mathrm{m}}(\mathbf{x}_i, \mathbf{x}_j; f)}{\partial f(\mathbf{x}_j)}||_2 = ||\frac{f(\mathbf{x}_i) - f(\mathbf{x}_j)}{||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2}||_2 = 1. \tag{10}$$

Consequently, the gradient magnitude of any embedding is only determined by our weighting strategy w_{ij}. In this case, it is also convenient to evaluate its influence, which is studied in section 4.3. In Eq. (9), $T \geq 0$ is the temperature parameter which controls the degree (slope) of weighting negative examples. If $T = 0$, it treats all non-trivial negative examples equally. If $T = +\infty$, it becomes the hardest negative example mining.

Optimisation Objective. For each query $\mathbf{x}_i^c$, we propose to make it closer to its positive set $\mathbf{P}_{c,i}$ than to its negative set $\mathbf{N}_{c,i}$ by a margin m. At the same time, we force all negative examples to be farther than a boundary α. Consequently, we pull all samples from the same class into a hypersphere. The diameter of each class hypershpere is $\alpha - m$.

In order to pull all non-trivial positive points in $\mathbf{P}_{c,i}^*$ together and learn a class hypersphere, we minimise:

$$L_{\mathrm{P}}(\mathbf{x}_i^c; f) = \frac{1}{|\mathbf{P}_{c,i}^*|} \sum_{\mathbf{x}_j^c \in \mathbf{P}_{c,i}^*} L_{\mathrm{m}}(\mathbf{x}_i^c, \mathbf{x}_j^c; f). \tag{11}$$

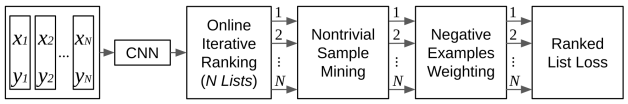


Figure 3: The overall framework of our proposed ranked list loss. For each input mini-batch, every image acts as a query iteratively and obtains a list of other images ranked by the similarity scores. For each ranked list, we mine non-trivial data points and weight negative examples based on their pairwise margin losses with respect to the query. At last, the ranked list loss is computed for every query.

We do not weight positive points because there exist only a few positive examples. Similarly, to push the non-trivial negative points in $\mathbf{N}^*_{c,i}$ beyond the boundary α, we minimise:

$$L_{\mathrm{N}}(\mathbf{x}_i^c; f) = \sum_{\mathbf{x}_j^k \in |\mathbf{N}^*_{c,i}|} \frac{w_{ij}}{\sum_{\mathbf{x}_j^k \in |\mathbf{N}^*_{c,i}|} w_{ij}} L_{\mathrm{m}}(\mathbf{x}_i^c, \mathbf{x}_j^k; f). \tag{12}$$

In RLL, we treat the two minimisation objectives equally and optimise them jointly:

$$L_{\mathrm{RLL}}(\mathbf{x}_i^c; f) = L_{\mathrm{P}}(\mathbf{x}_i^c; f) + \lambda L_{\mathrm{N}}(\mathbf{x}_i^c; f), \tag{13}$$

where λ controls the balance between positive and negative sets. We fix $\lambda = 1$ without tuning, which works well in our practice. In the ranked list of $\mathbf{x}_i^c$, we regard the features of other examples as constants. Therefore, only $f(\mathbf{x}_i^c)$ is updated based on the influence of weighted combination of other elements.

3.3. Learning Deep Models Based on RLL

To learn deep models, we implement our RLL based on mini-batch and stochastic gradient descent. Each mini-batch is a randomly sampled subset of the whole training classes, which can be regarded as *a simplified ranking problem with a smaller gallery* (identifying the matching examples from a smaller number of classes).

Every image $\mathbf{x}_i^c$ in the mini-batch acts as the query (anchor) iteratively and the other images serve as gallery. The RLL of each mini-batch is represented as:

$$L_{\mathrm{RLL}}(\mathbf{X}; f) = \frac{1}{N} \sum_{\forall c, \forall i} L_{\mathrm{RLL}}(\mathbf{x}_i^c; f), \tag{14}$$

N is the batch size. The learning of the deep embedding function f based on RLL is illustrated in Algorithm 1. The overall pipeline is shown in Figure 3.

Computational Complexity. As illustrated in Algorithm 1, our proposed method does not require the input data to be prepared in any rigid format, e.g., triplets, n-pair tuplets. Instead, it takes random input images with multiclass labels. We conduct online iterative ranking and loss computation (step 2 in Algorithm 1) after obtaining images' embeddings (step 1 in Algorithm 1). Therefore, the computational complexity of RLL is $O(N^2)$, which is the same as existing ranking-motivated structured loss functions [20, 28, 19].

Algorithm 1 Ranked List Loss on one mini-batch

Mini-Batch Setting: The batch size N, the number of classes C, the number of images per class N_c.
Parameters Setting: The distance constraint α on negative examples, the margin between positive and negative examples m, the weighting temperature T.
Input: $\mathbf{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\{\mathbf{x}_i^c\}_{i=1}^{N_c}\}_{c=1}^C$, the embedding function f, the learning rate β.
Output: Updated f.
Step 1: Feedforward all images $\{\mathbf{x}_i\}_{i=1}^N$ into f to obtain the images' embeddings $\{f(\mathbf{x}_i)\}_{i=1}^N$.
Step 2: Online iterative ranking and loss computation.
    foreach $f(\mathbf{x}_i^c) \in \{\{f(\mathbf{x}_i^c)\}_{i=1}^{N_c}\}_{c=1}^C$ **do**
        Mine non-trivial positive set $\mathbf{P}^*_{c,i}$.
        Mine non-trivial negative subsets $\mathbf{N}^*_{c,i}$.
        Weight negative examples as Eq. (9).
        Compute $L_{\mathrm{P}}(\mathbf{x}_i^c; f)$ as Eq. (11):
        Compute $L_{\mathrm{N}}(\mathbf{x}_i^c; f)$ as Eq. (12).
        Compute $L_{\mathrm{RLL}}(\mathbf{x}_i^c; f)$ as Eq. (13).
    end for
    Compute $L_{\mathrm{RLL}}(\mathbf{X}; f)$ as Eq. (14).
Step 3: Gradient computation and backpropagation to update the parameters of f.
    $\nabla_f = \partial L_{\mathrm{RLL}}(\mathbf{X}; f)/\partial f$
    $f = f - \beta \cdot \nabla_f$

3.4. Implementation Details

In each mini-batch, we randomly sample C classes and K images per class. We set $C = 60, K = 3$. Thus $N = 180$, $N_c = K = 3, \forall c$. In this case, there are 2 positive images and 177 negative images in the ranked list corresponding to each query, which simulates the global set-based similarity structure. More precisely, only a few matching examples exist in a large gallery. We use the same data preprocessing and augmentation as in [20]. Specifically, the input images are first resized to 256×256 and then cropped at 227×227. During training, we use random crop and random horizontal mirroring for data augmentation. For testing, we only use a single center crop without mirroring. We set the embedding size to 512 on all datasets following the setting in [16, 28]. As done in [29, 19], the features are L_2 normalised before computing their distance during training and testing.

We use GoogLeNet V2 [12] as our backbone network for fair comparison with [29, 19, 16]. In this net, there are three fully connected layers used in different layers. We re-

fer them based on their relative locations as follows: L for low-level layer (inception-3c/output), M for mid-level layer (inception-4e/output) and H for high-level layer (inception-5b/output). Following [29, 19, 16], the pretrained model on ImageNet [23] is used for initialisation in our experiments. Three original 1000-neuron fully connected layers followed by the softmax layer and cross-entropy loss are changed to three 512-neuron fully connected layers followed by our proposed ranked list loss. According to [20], the new layers are randomly initialised and optimised with 10 times larger learning rate than the others for faster convergence.

We use GoogLeNet V2 [12] (with batch normalization) as our backbone network for fair comparison with [29, 19, 16]. Following them, the pretrained model on ImageNet [23] is used for initialisation in our experiments. Three original 1000-neuron fully connected layers followed by the softmax layer and cross-entropy loss are changed to three 512-neuron fully connected layers followed by our proposed ranked list loss. According to [20], the new layers are randomly initialised and optimised with 10 times larger learning rate than the others for faster convergence. Our method is implemented in the Caffe deep learning framework [13].

4. Experiments

4.1. Datasets and Settings

Datasets. We conduct experiments on three popular benchmarks: (1) *CUB-200-2011* [15] has 11,788 images of 200 bird species. 5,864 images of the first 100 classes are used for training and 5,924 images of the other 100 classes for testing. (2) *CARS196* [32] contains 16,185 images of 196 car models. We use the first 98 classes (8,054 images) for training and the remaining 98 classes (8,131 images) for testing. (3) *SOP* [20] contains 120,053 images of 22,634 online products sold on eBay.com. 59,551 images of 11,318 categories and 60,502 images of 11,316 categories are used for training and testing respectively. The train/test split and evaluation protocol are the same as [20]. For CUB-200-2011 and CARS196, our method is evaluated on the original images (without using the bounding box information).

Metrics. Following the standard [20], we report the image retrieval performance and the image clustering quality in terms of Recall@K and NMI [25] respectively.

Training Settings. We run our experiments on a single Tesla V100 GPU with 32 GB RAM. The standard stochastic gradient descent (SGD) optimiser is used with a momentum of 0.9, a weight decay rate of $1e^{-5}$. We set the base learning rate to $1e^{-2}$ for CARS196 and SOP. Smaller base learning rate $1e^{-3}$ is better for CUB as it overlaps a little with ImageNet and contains fewer images [15, 34]. On CARS and CUB, the training procedure converges at $10k$ iterations, while $16k$ iterations for SOP. We set hyper-parameters empirically as follows $m = 0.4, T = 10, \alpha = 1.2$.

4.2. Comparison with the State-of-the-art Methods

Competitors. We compared our method with the following methods which are implemented and tested under the same setting: Triplet Semihard, Lifted Struct, N-pair-mc, Struct Clust, Spectral Clust, and Proxy NCA [1]. The methods have been described in Section 2 except for Triplet Semihard [24] that mines semihard negative examples to improve the conventional triplet loss.

Results. The comparison between our method and other competitors on both small datasets (CUB-200-2011 and CARS196) and large dataset (SOP) is presented in Table 1 and Table 2 respectively. As stated in Section 3.4, we have three fully connected layers in GoogLeNet V2. We report two sets of results. For fair comparison, we report the results of the high-level embedding, which is denoted as RLL-H. In addition, we find emperically that the multilevel embedding (RLL-(L,M,H) in short) by concatenating the low-level, mid-level and high-level embeddings can achieve better performance. We have the following observations from Table 1 and 2:

- Overall, our method outperforms all the compared methods. This validates the effectiveness of our proposed loss function.
- On the small datasets CARS196 and CUB-200-2011, RLL-H achieves the state-of-the-art performance on two tasks by single-level embedding. For example, on CUB-2011-2011, the Recall@1 and NMI of RLL-H are higher than previous state-of-the-art by 4.2% and 4.1%, respectively.
- On the large dataset SOP, RLL-H also outperforms all the previous methods in the image retrieval task. Specifically, the Recall@1 is higher than Proxy NCA by 2.4%. However, our method slightly underperforms with comparison to Proxy NCA. Interestingly, all methods perform similarly when this metric is used. This may indicate that NMI is not a good metric for a large-scale dataset.
- RLL-(L,M,H) works much better than RLL-H on every dataset. This indicates that the multilevel embedding is more discriminative than the single-level feature representation in our method.

Discussion. It is worth to mention that although the performance of Proxy NCA is also good on CARS196 and SOP,

[1] The methods in [7, 34, 31, 4, 17] using GoogLeNet V1 [30] and the margin loss [38] using ResNet50 [8] are not reported. Also, we *do not compare with ensemble models* [41, 21, 14, 39] since ours is single model. Although HTL [5] also uses GoogLeNet V2, we do not benchmark it because it builds the global class-level hierarchical tree by using all original classes as leaves and updates the tree after every epoch, thus being very computationally expensive and unscalable.

Table 1: Comparison with the state-of-the-art methods on CARS196, CUB-200-2011 in terms of Recall@K (%) and NMI (%). All the compared methods use GoogLeNet V2 as the backbone architecture. For fair comparison, RLL-H denotes single-level embedding, i.e., the high-level embedding. RLL-(L,M,H) denotes multilevel embedding by concatenating the low-level, mid-level and high-level embeddings.

	CARS196					CUB-200-2011				
	R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI
Triplet Semihard [24]	51.5	63.8	73.5	82.4	53.4	42.6	55.0	66.4	77.2	55.4
Lifted Struct [20]	53.0	65.7	76.0	84.3	56.9	43.6	56.6	68.6	79.6	56.5
N-pair-mc [28]	53.9	66.8	77.8	86.4	57.8	45.4	58.4	69.5	79.5	57.2
Struct Clust [29]	58.1	70.6	80.3	87.8	59.0	48.2	61.4	71.8	81.9	59.2
Spectral Clust [16]	73.1	82.2	89.0	93.0	64.3	53.2	66.1	76.7	85.3	59.2
Proxy NCA [19]	73.2	82.4	86.4	88.7	64.9	49.2	61.9	67.9	72.4	59.5
RLL-H	**74.0**	**83.6**	**90.1**	**94.1**	**65.4**	**57.4**	**69.7**	**79.2**	**86.9**	**63.6**
RLL-(L,M,H)	**82.1**	**89.3**	**93.7**	**96.7**	**71.8**	**61.3**	**72.7**	**82.7**	**89.4**	**66.1**

Table 2: Comparison with the state-the-of-art methods on SOP. The evaluation settings follow Table 1. The '–' denotes the corresponding results are not reported in the original paper.

	SOP			
	R@1	R@10	R@100	NMI
Triplet Semihard [24]	66.7	82.4	91.9	89.5
Lifted Struct [20]	62.5	80.8	91.9	88.7
N-pair-mc [28]	66.4	83.2	93.0	89.4
Struct Clust [29]	67.0	83.7	93.2	89.5
Spectral Clust [16]	67.6	83.7	93.3	89.4
Proxy NCA [19]	73.7	–	–	**90.6**
RLL-H	**76.1**	**89.1**	**95.4**	89.7
RLL-(L,M,H)	**79.8**	**91.3**	**96.3**	90.4

Proxy NCA does not preserve the scalability of deep metric learning as the number of classes needs to be considered. Proxy NCA learns one proxy per class, which also requires more learning parameters. Our method not only achieves better performance, but also preserves the scalability of deep metric learning, as shown in Table 2.

4.3. Ablation study

4.3.1 Mining Non-trivial Examples

As presented in Section 3.2, for each query, RLL mines examples which violate the pairwise constraint with respect to the query. Specifically, we mine negative examples whose distance is smaller than α in Eq. (12). Simultaneously, we mine positive examples whose distance is larger than $\alpha - m$ in Eq. (11). As a result, a margin m is established between negative and positive examples in each ranked list. Since the sample mining range is determined by the constraint parameters α, m, we conduct experiments on the large dataset SOP to analyse their influence.

Impact of α. To study the impact of α, we set the temperature $T = 10$ and the margin $m = 0.4$ in all experiments. The results are presented in Table 3. We observe that a proper negative constraint α is important for RLL to learn discriminative embeddings. This is consistent with our intuition as α controls how much the negative examples are pushed away.

Impact of m. To see the impact of m, we fix $\alpha = 1.2$ and $T = 10$. The results of different margin values are presented in Table 4. We have three important observations:

- When $m > 0$, RLL performs much better by around 3% than $m = 0$. It shows that the margin is important for improving the generalisation capability of RLL.
- The margin-based RLL is insensitive to the margin value. The performance difference is smaller than 1% when m ranges from 0.2 to 1.2.
- When $m = \alpha = 1.2$, there is no mining over positive

Table 3: The impact of α on the distance distribution of negative examples. SOP is used. Recall@K (%) results are reported. In all experiments, $m = 0.4, T = 10$.

$m = 0.4, T = 10$	R@1	R@10	R@100
$\alpha = 1.4$	76.2	89.4	95.6
$\alpha = 1.2$	**79.8**	**91.3**	**96.3**
$\alpha = 1.0$	78.7	90.5	95.9

Table 4: The impact of the distance margin m between negative and positive examples. The Recall@K (%) results on SOP are shown with $\alpha = 1.2, T = 10$ in all experiments.

$\alpha = 1.2, T = 10$	R@1	R@10	R@100
$m = 0$	76.1	89.8	95.7
$m = 0.2$	79.0	91.2	**96.3**
$m = 0.4$	**79.8**	**91.3**	**96.3**
$m = 0.6$	79.2	90.6	96.0
$m = 1.2$	79.1	90.5	95.8

points ($\alpha - m = 0$). In this case, RLL pulls positive examples as close as possible, which has the same effect as conventional contrastive loss.

4.3.2 Weighting Negative Examples

In this section, we conduct experiments to evaluate the influence of different temperatures T for weighting negative examples in Eq. (9). We fix $m = 0.4$ and $\alpha = 1.2$ in all experiments. The temperature parameter $T(T > 0)$ controls the slope of weighting. The results are presented in Table 5. We find:

- When $T = 0$, RLL treats all non-trivial negative examples equally, i.e., no weighting is applied. The Recall@1 result is 78.8%, which is only 1% lower than the best performance using proper weighting. This demonstrates the superiority of RLL even without weighting.
- RLL is insensitive to the setting of T. The performance gap is around 1% when T ranges from 0 to 20. In addition, the performance drops when T is large. This is because 'very' hard examples exist in the training data (e.g., outliers) [24, 3].

Table 5: The results of different T on SOP in terms of Recall@K (%). We fix $m = 0.4, \alpha = 1.2$ in all experiments.

$m = 0.4, \alpha = 1.2$	R@1	R@10	R@100
$T = 0$	78.8	90.7	96.1
$T = 5$	79.1	91.0	96.2
$T = 10$	**79.8**	**91.3**	**96.3**
$T = 15$	79.3	90.9	96.0
$T = 20$	78.6	90.5	95.7

4.3.3 Single-level versus Multilevel Embeddings

As demonstrated in Section 4.2, we find that RLL performs better using the multilevel embedding. To compare different single-level embeddings with the multilevel embedding, we conduct experiments on SOP. The results are reported in Table 6. We observe that:

- All single-level embeddings perform worse than the multilevel embedding by around 3%.
- The low-level embedding also performs very well in contrast with mid-level and high-level embeddings. The performance gap of different single-level embeddings is smaller than 1%. Note that the low-level embedding can be used for fast inference, which is essential for resource-constrained computational devices, e.g., mobile phones.

Table 6: Single-level embeddings versus multilevel embedding on SOP in terms of Recall@K (%). L, M and H represent the low-level, mid-level and high-level embedding respectively. (L,M,H) means the concatenation of low-level, mid-level and high-level embeddings.

Embedding	R@1	R@10	R@100
L	76.1	88.8	94.9
M	76.9	89.6	95.5
H	76.1	89.1	95.4
(L,M,H)	**79.8**	**91.3**	**96.3**

4.3.4 The Impact of Batch Size

The batch size is usually important in deep metric learning. It determines the size of problem we are going to solve every iteration during training. As presented in Section 3.3 and 3.4, the batch size decides the number of negative classes in the gallery. We conduct experiments on SOP to evaluate the influence of batch size in our approach. Specifically, we fix the number of images per class ($\forall c, Nc = K = 3$) and only change the number of classes ($C \in \{40, 50, 55, 60, 65\}$) in each mini-batch. The results are reported in Table 7. We can see that the batch size does not play a crucial role in RLL. The performance gap is only 0.6% when C changes from 120 to 195.

Table 7: The results of different batch size on SOP.

Batch size	R@1	R@10	R@100
$120 = 40 \times 3$	79.2	90.9	96.2
$150 = 50 \times 3$	79.5	91.1	96.2
$165 = 55 \times 3$	79.7	91.2	**96.3**
$180 = 60 \times 3$	**79.8**	**91.3**	**96.3**
$195 = 65 \times 3$	**79.8**	**91.3**	**96.3**

5. Conclusion

In this paper, the ranked list loss is proposed to exploit all informative data points in order to build a more informative structure for learning discriminative embeddings, which is not considered in the previous ranking-motivated losses. Given a query, RLL splits its positive and negative sets and forces a margin between them. In addition, non-trivial samples mining and negative examples weighting are exploited to make better use of all informative data points. The proposed RLL achieves state-of-the-art performance on three popular benchmarks using the single-level embedding. In addition, we find that RLL works better when using the multilevel embedding empirically.

References

[1] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, pages 1705–1749, 2005. 3

[2] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 1

[3] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *CVPR*, 2016. 1, 4, 8

[4] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep adversarial metric learning. In *CVPR*, 2018. 6

[5] W. Ge, W. Huang, D. Dong, and M. R. Scott. Deep metric learning with hierarchical triplet loss. In *ECCV*, 2018. 6

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

[7] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, et al. Smart mining for deep metric learning. In *ICCV*, 2017. 6

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

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

[10] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *ICASSP*, 2016. 1

[11] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016. 4

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5, 6

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMMM*, 2014. 6

[14] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon. Attention-based ensemble for deep metric learning. In *ECCV*, 2018. 6

[15] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, 2013. 2, 6

[16] M. T. Law, R. Urtasun, and R. S. Zemel. Deep spectral clustering learning. In *ICML*, 2017. 1, 3, 5, 6, 7

[17] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou. Deep variational metric learning. In *ECCV*, 2018. 6

[18] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, 2015. 3

[19] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *ICCV*, 2017. 1, 2, 3, 5, 6, 7

[20] 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

[21] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Bier-boosting independent embeddings robustly. In *ICCV*, 2017. 6

[22] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, 2014. 1

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 211–252, 2015. 6

[24] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1, 2, 3, 4, 6, 7, 8

[25] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*. Cambridge University Press, 2008. 3, 6

[26] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li. Embedding deep metric for person re-identification: A study against large variations. In *ECCV*, 2016. 4

[27] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 4

[28] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016. 1, 2, 3, 4, 5, 7

[29] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *CVPR*, 2017. 1, 2, 3, 5, 6, 7

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 6

[31] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, 2016. 6

[32] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 6

[33] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 1

[34] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *ICCV*, 2017. 6

[35] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 4

[36] X. Wang, Y. Hua, E. Kodirov, G. Hu, and N. M. Robertson. Deep metric learning by online soft mining and class-aware attention. In *AAAI*, 2019. 4

[37] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006. 2

[38] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. In *ICCV*, 2017. 4, 6

[39] H. Xuan, R. Souvenir, and R. Pless. Deep randomized ensembles for metric learning. In *ECCV*, 2018. 6

[40] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, 2016. 1

[41] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *ICCV*, 2017. 4, 6