Learning Better Visual Data Similarities via New Grouplet Non-Euclidean Embedding

Yanfu Zhang[1], Lei Luo[1], Wenhan Xian[1], Heng Huang[1,2]*
[1]Electrical and Computer Engineering, University of Pittsburgh, [2]JD Explore Academy
yaz91@pitt.edu, zzdxpyyll@gmail.com, wex37@pitt.edu, henghuanghh@gmail.com

Abstract

In many computer vision problems, it is desired to learn the effective visual data similarity such that the prediction accuracy can be enhanced. Deep Metric Learning (DML) methods have been actively studied to measure the data similarity. Pair-based and proxy-based losses are the two major paradigms in DML. However, pair-wise methods involve expensive training costs, while proxy-based methods are less accurate in characterizing the relationships between data points. In this paper, we provide a hybrid grouplet paradigm, which inherits the accurate pair-wise relationship in pair-based methods and the efficient training in proxy-based methods. Our method also equips a non-Euclidean space to DML, which employs a hierarchical representation manifold. More specifically, we propose a unified graph perspective — different DML methods learn different local connecting patterns between data points. Based on the graph interpretation, we construct a flexible subset of data points, dubbed grouplet. Our grouplet doesn't require explicit pair-wise relationships, instead, we encode the data relationships in an optimal transport problem regarding the proxies, and solve this problem via a differentiable implicit layer to automatically determine the relationships. Extensive experimental results show that our method significantly outperforms state-of-the-art baselines on several benchmarks. The ablation studies also verify the effectiveness of our method.

1. Introduction

Learning the semantic similarity between visual data is important for many machine vision tasks, including clustering [49], image retrieval [25, 34], person re-identification [48, 8], and few-shot learning [36, 33]. Conventionally, the Mahalanobis distance [24, 46] defined on hand-crafted features are exploited to characterize the data

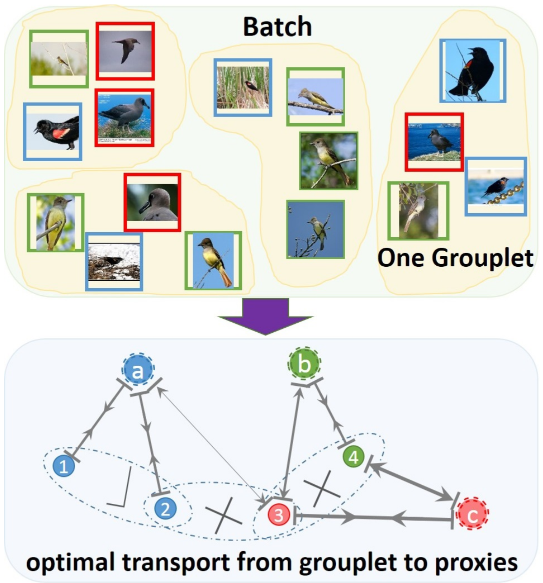

Figure 1: Top: Bounds of different colors indicate images from different clusters. Our method employs *grouplet* (yellow shaded), which can be viewed as randomly split of images in a batch. Bottom: Inside a grouplet, we determine the data-wise relationships dynamically via optimal transport. Different colors indicate different clusters. All pairs of data (circles with numbers) and proxy (circles with letters) are evaluated via *jointly* considering the data-wise relationships (*e.g.* automatically determine that $(1, 2)$ is a positive pair, and $(2, 3)$ is negative). Data embeddings are pulled (inner arrow) or pushed (outer arrow) harder to/away from the proxies if mismatched, *e.g.* $(3, a)$ is well distributed, and the push is weak (thin line); $(4, c)$ is badly distributed, which leads to a harsh push (thick line).

similarity. Boosted by the emerging advance of deep representation learning, refined distance metrics [17, 39] are proposed in recent works to accurately capture the geometric structure of data embeddings.

Existing supervised DML methods can be categorized into two paradigms, pair-based methods and proxy-based

*This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, IIA 2040588.

methods. Pair-based methods, *e.g.* contrastive loss [9, 4] and triplet loss [43, 32], consider pairwise relationships between data points. In detail, a pair of data with the same label is positive, otherwise negative. Pair-based losses then define some rules to select the tuples of pairs and learn the data relationships inside these pairs. To accelerate the training, sampling techniques detecting informative tuples are frequently utilized in pair-based methods. Alternatively, proxy-based methods, *e.g.* Proxy NCA [25] and soft triplet [31] introduce proxies to summarize subsets of training data and learn the data-wise relationship indirectly, through the data-proxy relationship. Proxy-based methods resolve the complexity issue in pair-based methods.

Though pair-based and proxy-based paradigms each have their own merits, both paradigms, however, have some intrinsic drawbacks. In pair-based methods, the training requires explicit pair relationships. An obvious advantage is that the local structures are evaluated more accurately. Nevertheless, in pair-based losses, the candidate training samples are composed of all valid combinations of data points (*e.g.* the number of training samples in contrastive loss [9] is the square of total data numbers), which substantially increases the training time. On the other hand, the training of proxy-based methods is based on individual data samples, which enjoys significantly better computational efficiency compared to pair-based methods. However, the data relationships learned in proxy-based methods are estimated and bounded through proxies. This indicates encoding may result in sub-optimal embedding manifolds.

To address the above problems, in this paper we propose a hybrid grouplet paradigm for non-Euclidean deep metric learning, which can be viewed as evaluating enlarged bipartite subgraphs consisting of multiple data points (dubbed *grouplet*) and proxies (Fig. 1). Our grouplet is free of explicit pair construction and doesn't require exhausting all combinations of data points. The data-wise relationships are dynamically determined via a constrained optimal transport layer. To further exploit the hierarchical structure of proxies and data points, we resort to a non-Euclidean embedding space and the associated similarity. Our contributions can be summarized as follows,

- We provide a graph perspective for deep metric learning, which generalizes two major DML paradigms, pair-based and proxy-based methods (Fig. 2).
- We formulate a grouplet deep metric learning method, which inherits the accurate estimation for pair-wise relationship from pair-based methods, and the efficient computation from proxy-based methods.
- We propose to learn non-Euclidean embeddings and the associated embedding-proxy similarity.
- The experimental results show significant improvement over state-of-the-art methods on several benchmarks. The ablation study demonstrates the effectiveness of our method.

Notations: Throughout the paper, the bold capital and bold lowercase symbols are used to represent matrices and vectors, respectively. $\boldsymbol{A} \geq 0$ denotes that elements of a matrix $\boldsymbol{A}$ are greater than or equal to 0. $G = \{V, E\}$ represents a graph with node set V and edge set E. A $n \times n$-identity matrix is denoted by $\mathbf{I}_n$, $\mathbf{1}_n$ is a n-dimension one vector, and $\mathbf{0}$ denotes a zero matrix.

2. Related Works

2.1. Non-Euclidean Representation Learning

Representation learning learns ubiquitous high-dimensional embeddings which can be utilized by various tasks. Based on the representations, classification or clustering models usually learn the hyper-planes to indicate the class boundaries. Conventionally, the embeddings learned by the models live the Euclidean space. Recently, non-Euclidean space has been an emerging research area [23], where space is parameterized by the curvature and the hyper-planes are defined as geodesics. For example, spherical embeddings are used in few-shot learning [40] and person re-identification [41]. Hyperbolic space also finds important applications, particularly in graph-related tasks [27, 6]. For example, trees can be embedded with arbitrary accuracy into the Poincarè ball. Hyperbolic neural networks [11] are used in natural language processing and image retrieval [16].

2.2. Deep Metric Learning

Pair-Based Losses: Pair-based methods define their losses on explicit positive or negative pairs. Among these methods, contrastive loss [9, 4] and triplet loss [43, 32] are two seminal examples. Generalized triplet losses, for example, N-pair loss [34] and lifted structure loss [29], associate multiple positive or negative data points to an anchor. Based on these works, General Pair Weighting (GPW) [44] proposed a framework unifying the pair weighting.
Proxy-Based Losses: The initial proxy-based work is Proxy-NCA [25], an approximated version of Neighborhood Component Analysis, via computing the distances between the embeddings and the cluster-wise proxies. Several extensions are developed recently, including SoftTriplet loss [31], Proxy Anchor [17], and ProxyNCA++ [39]. Proxy-based methods are efficient at the cost of coarsened capturing of data-to-data relationship.

Recently, there is also a unified benchmark for a comprehensive view of deep metric learning approaches [26].

2.3. Deep Implicit Layers

Deep neural networks are heavily dependent on the gradient-based optimization, *e.g.* Momentum Stochastic

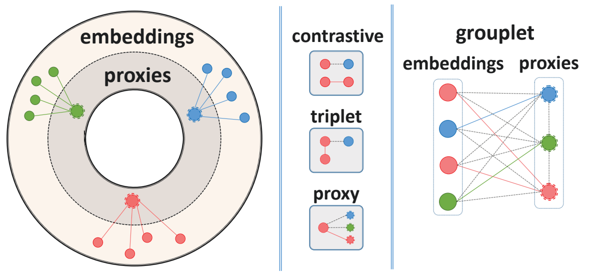

Figure 2: Illustration of DML. Left: embeddings (solid-line circles) for data points and proxies (dashed-line circles). Different colors indicate different clusters. Middle: three representative DML losses and the associated subgraphs. Colored solid lines between nodes indicate positive pairs, and black dashed lines indicate negative pairs. Right: a grouplet and the associated subgraph. The edges (weights computed by the optimal transport) represent the relationships between the grouplet to the proxies.

Gradient Descent [37] and Adam [19]. However, constrained optimization is seldom integrated into deep neural networks due to the difficulty of the automatic differentiation concerning the boundary. Recently, neural ordinary differential equations [7] provide a new interpretation for the residual neural layers, which states that each residual layer can be viewed as one differential operation. Inspired by this perspective, it is shown that deep neural networks can be utilized to solve convex constrained problems, referred to as differentiable optimization [1]. For example, OptNet [2] designs a special deep layer to solve quadratic programming problems.

3. Method

In this section, we first propose a unified graph perspective towards deep metric learning in §3.1. From this perspective, we describe our grouplet DML paradigm in §3.2, and introduce our non-Euclidean embeddings in §3.3. A differentiable optimal transport layer is discussed in §3.4 which makes our method end-to-end trainable.

3.1. Unifying DML from A Graph Perspective

DML methods learn the embeddings of data points via constraining the distances between them. Before introducing our grouplet non-euclidean approach, we first present a graph perspective towards understanding deep metric learning, illustrated in Fig. 2. From a graph perspective, we consider a weighted graph where the embeddings are viewed as node representations encoding the data points. We define $f : \mathbb{R} \to \mathbb{R}$ mapping the node-wise distances to the edge weights. Under this formulation, popular DML paradigms can be interpreted as learning the constrained local structures for the data distribution. Pair-based losses explicitly consider the relationship between data points sampled from different clusters. For example, triplet loss considers all valid subgraphs induced by an anchor, a positive sample, and a negative sample. Specifically, the triplet loss attempts to embed the data to the manifold such that the anchor is strongly connected to the positive sample, and weakly connected to the negative sample. Alternatively, proxy-based methods introduce auxiliary proxies to summarize different clusters, which can be interpreted as a hierarchical embedding structure. Proxies-based losses characterize the relationship between proxies and data points, which can be viewed as weighted bipartite graphs. Parallel to pair-based losses, the local connectivity patterns of the bipartite subgraphs are optimized to fit into the data relationships.

Compared to proxy-based losses, pair-based losses can learn a more accurate local structure as a consequence of directly evaluating the subgraphs. However, they involve substantially expensive training costs concerning the combination number of subgraphs. Proxy-based losses are efficient as that their training is point estimation defined on the bipartite data-proxy subgraphs.

3.2. Grouplet Deep Metric Learning

In this paper, we propose a hybrid paradigm — the hierarchical subgraphs composed of multiple data points (referred to as grouplet) and proxies. Compared to pair-based methods, our grouplet is more flexible — neither the precise number nor the clusters for data points are specified. Instead, we explicitly learn the structure of the bipartite subgraphs by solving the optimal transport from the grouplets to proxies. Of note, our hybrid method inherits the computational efficiency of proxy-based methods, meanwhile preserves the pair-wise relationship via the flexible grouplets because the data relationships are encoded in the constraints of the optimal transport problem. In the next part, we will detail the choice of the non-Euclidean embeddings and the associated similarity.

We consider cluster-wise proxies for c clusters, which is a standard step in proxies-based methods. A k-grouplet is defined as k randomly sampled data points. Our grouplet loss is defined on the bipartite subgraphs, consisting of a k-grouplet and the proxies. Of note, our grouplet is free of cluster considerations, *i.e.* the numbers of positive pairs and negative pairs are not specified. Instead, we characterize the bipartite subgraphs via the optimal transport cost, in which the positive pairs and negative pairs are identified automatically. Formally, the embeddings for a k-grouplet are denoted by $\boldsymbol{Y} \in \mathbb{R}^{n\times k}$ and the proxies $\boldsymbol{P} \in \mathbb{R}^{n\times c}$. $\boldsymbol{y}_i \in \mathbb{R}^n$ and $\boldsymbol{p}_j \in \mathbb{R}^n$ are the i-th data embedding and j-th proxy, respectively. Assume we have $d_p : \mathcal{Y} \times \mathcal{P} \to \mathbb{R}$ defining the similarity between the embedding space $\mathcal{Y} \subseteq \mathbb{R}^n$ and the proxies space $\mathcal{P} \subseteq \mathbb{R}^n$. Let the transport cost be

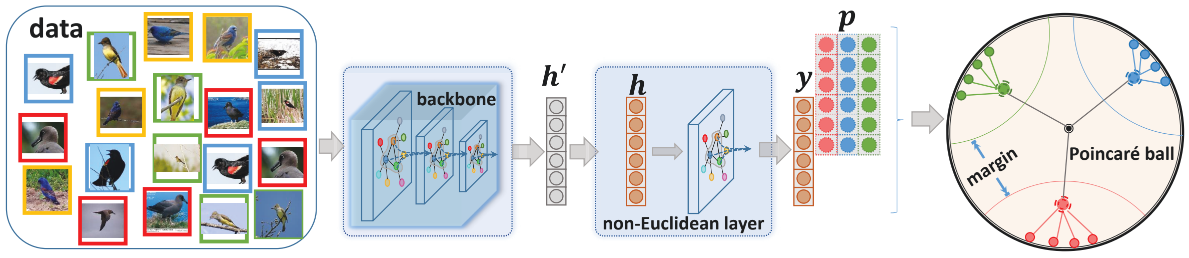

Figure 3: The outputs of backbone $\boldsymbol{h}'$ are transformed to non-Euclidean embeddings $\boldsymbol{y}$. The proxies $\boldsymbol{p}$ and the decision hyper-planes are in the same non-Euclidean space. The data representations in Poincaré ball are scattered and hierarchical. Colored lines denote the decision hyper-planes for different clusters. Similar to the Euclidean case, the margin between different hyper-planes — characterized by geodesics in non-Euclidean space — are constrained.

$c_{ij} = 1 - f(d_p(\boldsymbol{y}_i, \boldsymbol{p}_j))$, *i.e.* the edge weights indicating the data similarity. We have the optimal transport cost $\mathcal{W}(\boldsymbol{Y}, \boldsymbol{P})$ concerning the embeddings and proxies,

$$\mathcal{W}(\boldsymbol{Y}, \boldsymbol{P}) = \min_{x_{ij}} \sum_{i=1}^{k} \sum_{j=1}^{c} c_{ij} x_{ij}, \tag{1}$$

$$s.t. \qquad x_{ij} \geq 0, \sum_{i=1}^{k} x_{ij} = s_j, \sum_{j=1}^{c} x_{ij} = d_i, \forall i, j$$

here x_{ij} is the flow encoding the relationship between data point $\boldsymbol{y}_i$ and proxy $\boldsymbol{p}_j$. During the training stage, we have the data labels, denoted as t_i. As such, x_{it_i} encodes the positive relationship between $\boldsymbol{y}_i$ and $\boldsymbol{p}_{t_i}$, and x_{ij} when $j \neq t_i$ encodes the negative relationship. Compared to hard-coded pairs, the optimal transport problem explicitly considers the data-wise — via $\sum_{i=1}^{k} x_{ij} = s_j$ — and proxy-wise— via $\sum_{j=1}^{c} x_{ij} = d_i$ — relations in the constraints, which inherits the accurate local structure estimation in pair-based methods. We interpret x_{ij} as the probability that data i is with cluster j. As such, d_i is naturally set to 1 for all i. We let $s_j = \sum_{i=1}^{k} \mathbb{I}(t_i = j)$ where $\mathbb{I}$ is an indicator function, which corresponds to the ideal match of positive proxies and data labels. (1) can be solved via a differentiable layer, and we reserve the discussion to §3.4.

Given i, $\{x_{ij} | j \in \{1, \dots, c\}\}$ represents the relationship between data point i and all proxies. The intra-grouplet (*i.e. data-wise*) relationship is decomposed by the constrained optimal transport. We can safely insert x_{ij} into proxies-based losses. In this paper, we adopt the Proxy-Anchor loss [17], and our grouplet loss can be written as,

$$\ell(X, P) = \frac{1}{|\mathcal{C}_p^+|} \sum_{j \in \mathcal{C}_p^+} f_0(\{d_{ij}^+\}, \{x_{ij}\}) + \frac{1}{|\mathcal{C}_p|} \sum_{j \in \mathcal{C}_p} f_0(\{d_{ij}^-\}, \{x_{ij}\}), \tag{2}$$

here $\mathcal{C}_p^+ = \{j | \exists i, t_i = j\}$, $d_{ij}^+ = -\alpha(d_p(\boldsymbol{x}_i, \boldsymbol{p}_j) - \delta)$, $\mathcal{C}_p = \{1, \dots, c\}$, $d_{ij}^- = \alpha(d_p(\boldsymbol{x}_i, \boldsymbol{p}_j) + \delta)$, $f_0(\{d_{ij}\}, \{x_{ij}\}) = \log\left(1 + \sum_{\{i|t_i=j\}} (1 + x_{ij}) \exp(d_{ij})\right)$ and $|\cdot|$ on set denotes the cardinality. α and δ are parameters introduced by proxy anchor, and we follow the default settings in the original paper. Note that compared to the naive Proxy-Anchor loss, we substitute the positive proxy-embedding pairs with the computed x_{it_i}, and insert the negative pairs for unmatched x_{ij}.

Our grouplet paradigm explicitly and automatically determines the data-wise relationships via the constrained optimal transport problem. Moreover, our method is based on randomly selected data points. The construction of grouplets doesn't require specific positive or negative pairs selection, nor iterate all possible combinations of grouplets. The flow computed from the optimal transport also allows our method to benefit from the accelerated training of proxy-based methods.

3.3. Non-Euclidean Data Embeddings

To inherit a tree-like hierarchical data structure from proxies-based methods, we propose to employ non-Euclidean embeddings, which boosts the model performances empirically. In prior works, the data embeddings and proxies are in Euclidean spaces. Recently, non-Euclidean spaces are shown to be suitable for hierarchical data. For example, hyperbolic embeddings for images [16] can be effectively exploited by few-shot learning tasks. Motivated by the success of non-Euclidean representation learning, we propose to learn the data embeddings and proxies and define the similarity $d_p(\cdot)$ in non-Euclidean spaces. Specifically, we use the Poincaré ball model which is proven to be successful in image embeddings, which also empirically works well in our model. Poincaré ball model is derived from the Riemann manifold. Poincaré ball can be characterized by Möbius gyrovector space, in which the parallel operations to Euclidean space are defined, including exponential/logarithmic maps, Möbius addition, *etc.*. A comprehensive evaluation regarding several different geometric structures is summarized in the ablation studies §4.3.2. For more details, we refer the readers to related literature [11, 22].

The detailed structure of our non-Euclidean embeddings is illustrated in Fig. 3. For a fair comparison, minimal modification towards the structure in prior works is involved. We only substitute the Euclidean layer with a non-Euclidean layer of the same size and leave the rest part untouched. The only additional hyperparameter is the radius of the Poincaré ball. Of note, the backbone structure yields Euclidean representations. As such, the intermediate representations should be transformed before being fed to the non-Euclidean layer. For simplicity, given a Poincaré ball $\mathbb{D}^c$ of radius c, we assume the intermediate representations $\boldsymbol{y}'$ is in the tangent space of the Poincaré ball at the origin, $T_{\mathbf{0}}^c$. The transformation can be written as $\boldsymbol{y} = \text{proj}^c\left(\exp_{\mathbf{0}}^c\left(\boldsymbol{y}'\right)\right)$. Here, $\exp_{\mathbf{0}}^c(\cdot)$ is the exponential map to $T_{\mathbf{0}}^c$, $\text{proj}^c(\cdot)$ projects the the point in $T_{\mathbf{0}}^c$ to $\mathbb{D}^c$. After the non-Euclidean layer, the embeddings are also non-Euclidean. For consistency, the proxies are also represented in the same Poincaré ball, and we define $d_p(\boldsymbol{y}, \boldsymbol{p}) : T_{\mathbf{0}}\mathbb{D}^c \times T_{\mathbf{0}}\mathbb{D}^c \rightarrow \mathbb{R}$ in (2) as the non-Euclidean cosine similarity $d_p(\boldsymbol{y}, \boldsymbol{p}) = g_{\mathbf{0}}^{\mathbb{D}}(\log_{\mathbf{0}}^c(\boldsymbol{y}), \log_{\mathbf{0}}^c(\boldsymbol{p}))$. Here, $\log_{\mathbf{0}}^c(\cdot)$ is the logarithmic map to $T_{\mathbf{0}}^c$, and $g_{\mathbf{0}}^{\mathbb{D}}(\cdot)$ is the Riemannian metric on $\mathbb{D}^c$. It should be also stressed that for either the transformation or the non-Euclidean cosine similarity computation, no additional parameters are required.

Via the non-Euclidean representation learning and the associated similarity, we can take advantage of the hierarchical structure of data distributions. Specifically, there is only one single additional hyper-parameter introduced by our non-Euclidean DML, which allows a minimal effort for tuning our model.

3.4. Differentiable Optimal Transport Layer

Directly integrating (1) in our deep model will lead to intractable gradients. To solve this challenging problem, we resort to a deep implicit layer encoding the optimal transport computation. (1) is a linear programming problem computing the 1-Wasserstein distance between the grouplet and the proxies. c_{ij}, s_j and d_i are parameters related to the embeddings, which can be viewed as functions defined on some input $\boldsymbol{\theta}$. In our case, $\boldsymbol{\theta}$ can be interpreted as the intermediate embeddings of the backbone model and other involved model parameters. A deep implicit layer exploits the KKT (Karush–Kuhn–Tucker) conditions to solve a convex problem with the parameters fixed and computes the gradients *w.r.t.* $\boldsymbol{\theta}$ using the implicit function theorem. As such, we can build an end-to-end trainable deep neural network by inserting the implicit optimal transport layer to compute x_{ij} in (1). For self-containedness, the detailed derivation of the deep implicit layer encoding the optimal transport is given in the following.

We first re-write (1) in a compact form,

$$\mathcal{W} = \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{\theta}),\ s.t.\ G(\boldsymbol{\theta})\boldsymbol{x} \leq \mathbf{0}, h(\boldsymbol{x}, \boldsymbol{\theta}) = 0, \tag{3}$$

here, $N = c \times k$, $f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{c}^\intercal \boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^N$, and the k-th entry is x_{ij} with $i = \lfloor k/n \rfloor$ and $j = k \mod n$. $\boldsymbol{c} : \boldsymbol{\Theta} \rightarrow \mathbb{R}^N$ is the vectorized form of the transport cost. $G(\boldsymbol{\theta}) = diag\left(-\mathbf{1}_N\right)$, where $diag\left(\cdot\right)$ maps the given vector to a diagonal matrix. Note that $G(\boldsymbol{\theta})$ is usually a function on $\boldsymbol{\theta}$, and in our problem we take a a constant function. Let $\boldsymbol{s} \in \mathbb{R}^m$ and $\boldsymbol{d} \in \mathbb{R}^n$ be the vectorized $s_j(\boldsymbol{\theta})$ and $d_i(\boldsymbol{\theta})$. $h(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{A}\boldsymbol{x} - [\boldsymbol{s}, \boldsymbol{d}]$, where $[\boldsymbol{s}, \boldsymbol{d}]$ is the concatenation of $\boldsymbol{s}$ and $\boldsymbol{d}$. $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, where $M = c + k$, is defined as,

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1}_k^\intercal & & \\ & \ddots & \\ & & \mathbf{1}_k^\intercal \\ diag\left(\mathbf{1}_k\right) & \cdots & diag\left(\mathbf{1}_k\right) \end{bmatrix}. \tag{4}$$

(3) is a constrained convex problem and its Lagrangian is,

$$L\left(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta}\right) = \boldsymbol{c}^\intercal \boldsymbol{x} + \boldsymbol{\lambda}^\intercal G(\boldsymbol{\theta})\boldsymbol{x} + \boldsymbol{v}^\intercal h(\boldsymbol{x}, \boldsymbol{\theta}), \tag{5}$$

here $\boldsymbol{v}$ and $\boldsymbol{\lambda}$ are the dual variables for the equalities and inequalities in the constraints, respectively.

It is easy to verify that (3) satisfies the Slater's condition and the twice differentiability. As such, the KKT conditions of (5) are the necessary and sufficient optimality conditions for (3). More specifically, let $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda})$, we define

$$g\left(\boldsymbol{z}, \boldsymbol{\theta}\right) = \left[\nabla_{\boldsymbol{x}} L(\boldsymbol{z}, \boldsymbol{\theta}), diag(\boldsymbol{\lambda})G(\boldsymbol{\theta})\boldsymbol{x}, h(\boldsymbol{x}, \boldsymbol{\theta})\right]^\intercal. \tag{6}$$

If $g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) = 0$ for some $\tilde{\boldsymbol{z}} = (\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}})$ where $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{v}}$ are both feasible, then the KKT conditions are satisfied and $\tilde{\boldsymbol{x}}$ is optimal. The partial Jacobian regarding $\boldsymbol{z}$ is,

$$D_{\boldsymbol{z}} g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) = \begin{bmatrix} D_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} L(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) & G(\boldsymbol{\theta})^\intercal & D_{\boldsymbol{x}} h(\tilde{\boldsymbol{x}}, \boldsymbol{\theta})^\intercal \\ diag(\tilde{\boldsymbol{\lambda}})G(\boldsymbol{\theta}) & G(\boldsymbol{\theta})\tilde{\boldsymbol{x}} & 0 \\ D_{\boldsymbol{x}} h(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) & 0 & 0 \end{bmatrix}, \tag{7}$$

and the partial Jacobian regarding $\boldsymbol{\theta}$ is,

$$D_{\boldsymbol{\theta}} g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}) = \left[D_\theta \nabla_{\boldsymbol{x}} L(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}), 0, D_\theta h(\tilde{\boldsymbol{x}}, \boldsymbol{\theta})\right]^\intercal. \tag{8}$$

The following theorem characterizes the differentiability of convex optimization.

Theorem 1. *(Differentiable Convex Optimization [3]) Given a convex problem, assume (1) Slater condition holds, (2) all derivative exists, (3) $\{i | \lambda_i = 0 \text{ and } f_i(x, \theta)\} = \emptyset$, and (4) $D_{\boldsymbol{x}} g(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\theta})$ is non-singular. If $g(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\theta}) = 0$, the solution mapping has a single-valued localization s around $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{\lambda}}$ that is continuously differentiable in a neighborhood $\mathcal{Q}$ of $\boldsymbol{\theta}$ with Jacobian satisfying,*

$$D_{\boldsymbol{\theta}} s(\boldsymbol{\theta}) = -D_{\boldsymbol{z}} g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta})^{-1} D_{\boldsymbol{\theta}} g(\tilde{\boldsymbol{z}}, \boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \mathcal{Q}. \tag{9}$$

Remark. *Theorem 1 is an immediate result from the Implicit Function Theorem [20]. Recall that in our problem, $\boldsymbol{\theta}$ are the embeddings of subgraphs. Theorem 1 states that the gradient w.r.t. $\boldsymbol{\theta}$ can be computed via combining* (7) *and* (8)*. In other words, backward propagation is feasible.*

The advantage of our optimal transport problem is that the data-wise relationships are explicitly considered in the constraints. Meanwhile, the above results allow the differentiable optimal transport problem to be formulated as a trainable layer inserted into our grouplet loss, which makes our paradigm end-to-end trainable.

4. Experimental Results

4.1. Experimental Settings

Datasets The experiments are conducted as image retrieval task on CUB-200-2011 [42], Cars196 [21], and Stanford Online Products [29] datasets. We split the training and test sets following the conventional protocol [29].

CUB-200-2011 [42] contains 200 species of birds with 11,788 images. The first 100 species (5864 images) are used for training and the rest 100 species (5924 images) for testing.

Cars196 [21] contains 196 classes with 16,185 car images. The first 98 classes (8054 images) are used for training and the rest 98 classes (8131 images) for testing.

Stanford Online Products [29] contains 22,634 classes with 120,053 product images. The first 11,318 classes (59,551 images) are used for training and the rest 11,316 classes (60,502 images) are used for testing.

Evaluation Metrics We calculate Recall@n for the image retrieval task. For each query image, top-n most similar images are returned identified, and the retrieved images sharing the label with the query are positive, with the rest be negative. Additionally, we evaluate the clustering performance using the Normalized Mutual Information (NMI) based on the K-means algorithms.

Implementation Details: We use ResNet-50 as the backbone model initialized with an unsupervised pre-trained model on ImageNet dataset [5]. The non-Euclidean layer is adopted from the implementation of Hyperbolic Graph Convolutional Networks [6] using a Poincaré ball manifold with $c = 4$. We adopt the default setting in ProxyAnchor [17], $\alpha = 32$ and $\delta = 0.1$, without any tuning. The grouplet size is 4. The training batch is of the same meaning in regular proxy-based methods. For example, a batch size of 64 means we sample 64 images from the training set. The images are then split into 16 grouplets. We emphasize that our approach doesn't require constructing all combinations of grouplets, instead, it only inserts a re-grouping step concerning the data in each batch. We adopt the OptNet [2] to solve the optimal transport problem. OptNet is designed for solving quadratic programming (QP) problems. The backward propagation is accomplished via automatic differentiation through a customized QP solver [1]. To tailor our problem to fit into OptNet, we only need to insert a small quadratic term $10^{-4}\,(\boldsymbol{x}^\intercal \boldsymbol{x})$ into the objective function. The optimal transport can be viewed as part of the objective, which is not involved in the test stage. The class numbers of CUB-200-2011 [42] and Cars196 [21] are moderate, which can be efficient solved using the proposed scheme. The class number of Stanford Online Products [29] is significantly larger, which may result in the out-of-memory problem. As such, for each batch, we only compute the optimal transport using the proxies sharing the labels with the batch, and the rest are excluded from the computation.

Our approach is implemented in PyTorch and trained with an NVIDIA P40 GPU. The input image sizes for the training and the testing are both in 224×224. During the training, we use random cropping and horizontal mirroring for data augmentation. For the test phase, we resize images to 256×256 then compute the embeddings based on the center cropping. We use Adam optimizer with a learning rate of 0.0001 for the backbone model. The learning rate for the proxies is multiplied by 100 to accelerate the training. We train the models using batch size 64 for 60 epochs. We use a cosine annealing scheduler with 5 steps warm-up. Besides, we use 5 epochs warm-up for the proxies, which can stabilize the training.

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

Table 1 summarizes the comparison of our approach with state-of-the-art DML methods. The image size has a significant influence on the model performance, therefore, we only include those methods with crop size 224×224. On all datasets, our method is superior to previous arts for various embedding sizes.

The results of our approach are substantially superior to all previous methods on CUB-200-2011 and Cars-196. For example, on CUB-200-2011 our method (embedding size of 512) outperforms the second-best method by 5.7% in R@1. Cars-196 is a relatively simple dataset compared to CUB-200-2011. On this dataset, our method also yields an improvement of 3.8% compared to the second-best method. Note that the results of our approach are computed on a small batch size of 64, while the best performance of Proxy-Anchor is obtained on a large batch size of 160, which requires large memory. Therefore, our method not only achieves the best performance for different embedding sizes but also has lower hardware requirements for training.

The data statistics of SOP are significantly different from CUB-200-2011 and Cars-196. There are about 5 images for each class in SOP by average, while CUB-200-2011 and Cars-196 have more than 50 images. The difference may

[1]https://github.com/locuslab/qpth

		CUB-200-2011				Cars196				Stanford Online Products			
		NMI	R@1	R@2	R@4	NMI	R@1	R@2	R@4	NMI	R@1	R@10	R@100
LiftedStruct^{64} [29]	G	56.6	43.6	56.6	68.6	56.9	53.0	65.7	76.0	88.7	62.5	80.8	91.9
Clustering^{64} [28]	I	59.2	48.2	61.4	71.8	53.4	51.5	63.8	73.5	89.5	66.7	82.4	91.9
ProxyNCA^{64} [25]	I	59.5	49.2	61.9	67.9	64.9	73.2	82.4	86.4	90.6	73.7	–	–
SmartMining^{64} [13]	G	59.9	49.8	62.3	74.1	59.5	64.7	76.2	84.2	–	–	–	–
SoftTriplet^{64} [31]	I	$\underline{66.2}$	60.1	71.9	81.2	$\underline{69.3}$	65.4	76.4	84.5	**91.7**	76.3	89.1	95.3
ProxyGML^{64} [53]	I	65.1	59.4	70.1	80.4	67.9	$\underline{78.9}$	$\underline{87.5}$	91.9	89.8	76.2	$\underline{89.4}$	$\underline{95.4}$
ProxyAnchor^{64} [17]	I	–	$\underline{61.7}$	$\underline{73.0}$	$\underline{81.8}$	–	78.8	87.0	$\underline{92.2}$	–	$\underline{76.5}$	89.0	95.1
Proposed^{64}	R	**71.5**	**66.5**	**77.3**	**85.2**	**73.8**	**83.9**	**90.4**	**94.1**	$\underline{90.7}$	**79.1**	**90.8**	**96.2**
Margin^{128} [47]	R	$\underline{69.0}$	$\underline{63.6}$	$\underline{74.4}$	$\underline{83.1}$	$\underline{69.1}$	$\underline{79.6}$	$\underline{86.5}$	$\underline{91.9}$	$\underline{90.7}$	$\underline{72.7}$	$\underline{86.2}$	$\underline{93.8}$
Proposed^{128}	R	**73.5**	**70.0**	**79.9**	**87.0**	**75.6**	**87.8**	**92.5**	**95.5**	**90.9**	**80.8**	**91.9**	**96.7**
HDC^{384} [51]	G	–	53.6	65.7	77.0	–	73.7	83.2	89.5	–	69.5	84.4	92.8
ProxyGML^{384} [53]	I	$\underline{68.4}$	$\underline{65.2}$	$\underline{76.4}$	$\underline{84.3}$	$\underline{70.9}$	$\underline{84.5}$	$\underline{90.4}$	$\underline{94.5}$	$\underline{90.1}$	$\underline{77.9}$	$\underline{90.0}$	$\underline{96.0}$
Proposed^{384}	R	**75.7**	**74.0**	**82.8**	**89.2**	**78.7**	**91.0**	**94.5**	**96.8**	**90.9**	**82.1**	**92.8**	**97.2**
HDML^{512} [52]	G	62.6	53.7	65.7	76.7	69.7	79.1	87.1	92.1	89.3	68.7	83.2	92.4
HTL^{512} [12]	I	–	57.1	68.8	78.7	–	81.4	88.0	92.7	–	74.8	88.3	94.8
RLL-H^{512} [45]	I	–	57.4	69.7	79.2	–	74.0	83.6	90.1	–	76.1	89.1	95.4
A-BIER^{512} [30]	G	–	57.5	68.7	78.3	–	82.0	89.0	93.2	–	74.2	86.9	94.0
ABE^{512} [18]	G	–	60.6	71.5	79.8	–	85.2	90.5	94.0	–	76.3	88.4	94.8
SoftTriplet^{512} [31]	I	69.3	65.4	76.4	84.5	70.1	84.5	90.7	94.5	**92.0**	78.3	90.3	95.9
MS^{512} [44]	I	–	65.7	77.0	86.3	–	84.1	90.4	94.0	–	78.2	90.5	96.0
TML^{512} [50]	R	–	62.5	73.9	83.0	–	86.3	92.3	95.4	–	78.0	91.2	96.7
CircleLoss^{512} [35]	I	–	66.7	77.4	86.2	–	83.4	89.8	94.1	–	78.3	90.5	96.1
GL^{512*} [10]	G	69.0	65.5	77.0	85.0	72.7	85.6	91.2	94.9	$\underline{91.1}$	75.7	88.2	94.8
ProxyGML^{512} [53]	I	$\underline{69.8}$	66.6	77.6	86.4	$\underline{72.4}$	85.5	91.8	95.3	90.2	78.0	90.6	$\underline{96.2}$
ProxyAnchor^{512} [17]	I	–	68.4	79.2	86.8	–	86.1	91.7	95.0	–	$\underline{79.1}$	$\underline{90.8}$	$\underline{96.2}$
ProxyAnchor^{512} [17]	R	–	$\underline{69.7}$	$\underline{80.0}$	$\underline{88.1}$	–	$\underline{87.7}$	$\underline{93.0}$	$\underline{96.1}$	–	–	–	–
Proposed^{512}	R	**76.4**	**75.4**	**83.4**	**90.1**	**77.6**	**91.5**	**94.8**	**96.9**	$\underline{91.1}$	**82.0**	**92.7**	**97.2**

Table 1: Comparison with the state-of-the-art methods. Superscript denotes embedding dimension. The best results are boldfaced, and the runner-ups are underlined. If the results are unavailable from the original paper, the cell is filled with "-". The backbone networks is denoted by abbreviations next to the baselines: I — Inception with batch normalization [15], G — GoogleNet [38], and R — ResNet-50 [14]. Note that GL^{512*} [10] uses 227×227 images instead of 224×224.

result in the performance improvement gap between different datasets. Nevertheless, our approach has a better performance compared to most of the state-of-the-art methods.

Of note, TML [50] and GL [10] are two related methods that also use randomly generated groups of data to computing data-wise similarity. However, TML [50] requires iterating all combinations of tuplets and specialized mining techniques, which is far less efficient nor scalable compared to our method. Compared to GL [10], our method automatically computes the data-wise relationships and learns non-Euclidean embeddings. Moreover, our model performs significantly better than TML [50] (12.9% higher R@1 on CUB-200-2011) and GL [10] (9.9% higher R@1 on CUB-200-2011).

To further demonstrate the performance of our approach, we present some qualitative results in Fig. 4. The embeddings of our approach can accurately retrieve the similar instances under various challenges, including the pose variation and background clutter in CUB-200-2011 and Cars-196, and the view-point changes in SOP.

4.3. Ablation Study

4.3.1 Model Sensitivity and Efficiency

As aforementioned, our method inherits the computational efficiency of proxy-based methods. Moreover, compared to the base method Proxy-Anchor, our method doesn't require a large batch to achieve the best results, which reduces the memory requirement. Table 2 compares the computational resources for Proxy-Anchor and our approach. For Proxy-Anchor, we include two settings. The batch size has an enormous influence on the model performance, and to achieve the best performance, up to 16 G memory is required. Compared to Proxy-Anchor, our method has comparable per-epoch training time and consumes significantly less memory. For example, *Proposed* 16×4 (equal to batch size 64) performs much better and consumes only

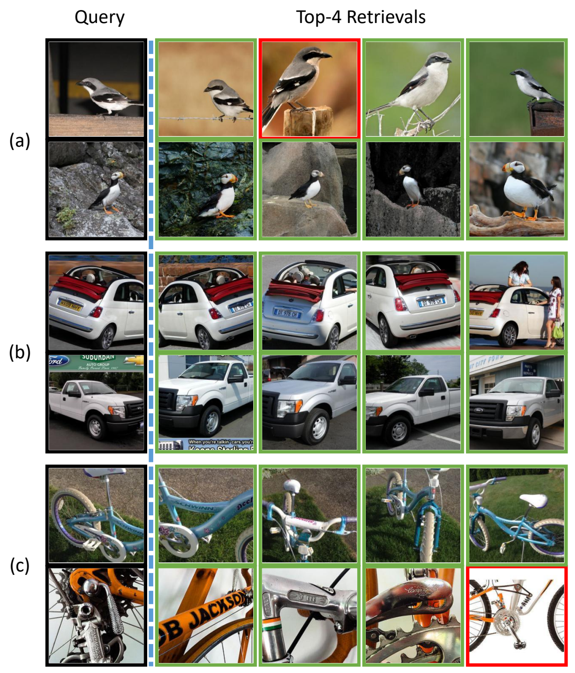

Figure 4: Qualitative results on (a) CUB-200-2011, (b) Cars-196, and (c) SOP. We present the top 4 retrievals for each query image. The retrievals with green boundary are positive cases, and those with red boundary are failure cases. Even for failed retrievals, substantially similar visual appearances are shared with queries.

	Batch	Mem.	Time	R@1
ProxyAnchor^{512} [17]	64	6.0	43	67.1
ProxyAnchor^{512} [17]	180	16.1	40	69.0
Proposed^{512}	8×4	3.2	81	73.6
Proposed^{512}	16×4	6.1	65	75.4
Proposed^{512}	32×4	11.9	63	75.5
Proposed^{512}	8×8	6.1	74	74.9
Proposed^{512}	12×6	6.8	70	74.7

Table 2: The resource requirements on CUB-200-2011. For the proposed method, *Batch* is denoted by *number of grouplet* $\times$ *grouplet size*, and their product is actual batch size of the same sense in Proxy-Anchor. *Mem.* is the max memory for training in Gigabytes, and *Time* is the per-epoch training time in seconds.

30% memory compared to *ProxyAnchor* 64.

The sensitivity of the size of grouplet k is also included in Table 2. For similar batch size (16×4, 8×8 and 12×6), the model performance is not very sensitive to k.

4.3.2 Choice of Embedding Manifold

To demonstrate the impact of the non-Euclidean layer, in this section we consider two important questions and provide the empirical answers on CUB-200-2011.

First, *whether non-Euclidean layer improves the embedding learning*. To demonstrate the impact of integrating a non-euclidean layer, in Table 3 we examine the model performance of two state-of-the-art methods, ProxyGML and ProxyAnchor, under different embedding manifolds. We use the original implementation provided by the authors and alter the embedding layers. Here, Euclidean refers to the original models, Hyperboboid and Poincarè Ball are two different non-Euclidean manifolds. *The results show that non-Euclidean layers lead to better embeddings regardless of the specific types of losses.*

	Euc.	Hyper.	P. Ball
proxyGML^{512} [53]	66.6	65.9	67.1
ProxyAnchor^{512} [17]	68.4	68.1	68.8
Proposed^{512}	74.8	73.6	75.4

Table 3: Comparison of embedding layers with different manifolds. *Euc.*: Euclidean; *Hyper.*: Hyperboboid; *P. Ball*: Poincarè ball. *Euc.* is identical to the standard setting.

Second, *how sensible the performance is concerning the manifold parameters.* Non-Euclidean layers introduce an additional hyperparameters, the radius c of Poincarè ball. In Fig. 5, we illustrate the model performance *v.s.* c. With confidence we believe that *the model performance is not sensitive if* $c \geq 2$.

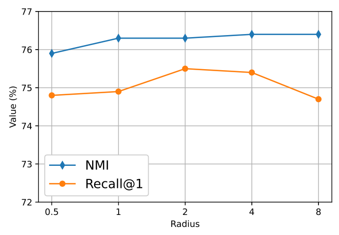

Figure 5: Model performance *v.s.* c on CUB-200-2011.

Based on the above results, we justify that deep metric learning presumably benefits from a properly defined non-euclidean layer. Particularly, we recommend the Poincarè ball model with $c = 4$ for general DML tasks.

5. Conclusion

In this paper, we proposed a new DML paradigm, inheriting the accurate pair-wise relationship of pair-based methods, and the computational efficiency of proxy-based methods. Our method utilizes the hierarchical structure of data embedding via non-Euclidean representation learning and the associated similarity, and employs a differentiable optimal transport layer to learn the data relationships automatically. The experimental results demonstrate our method significantly outperforms previous arts on several standard benchmark datasets, and the ablation study also shows the effectiveness of our design.

References

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019. 3

[2] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017. 3, 6

[3] Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018. 5

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, pages 737–737, 1994. 2

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 6

[6] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32:4869, 2019. 2, 6

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018. 3

[8] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2017. 1

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005. 2

[10] Ismail Elezi, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, and Laura Leal-Taixé. The group loss for deep metric learning. In *European Conference on Computer Vision*, pages 277–294. Springer, 2020. 7

[11] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*, 2018. 2, 5

[12] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 7

[13] Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2829, 2017. 7

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 7

[16] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020. 2, 4

[17] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3238–3247, 2020. 1, 2, 4, 6, 7, 8

[18] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 736–751, 2018. 7

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[20] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012. 6

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 6

[22] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *arXiv preprint arXiv:1910.12892*, 2019. 5

[23] Lei Luo, Jie Xu, Cheng Deng, and Heng Huang. Robust metric learning on grassmann manifolds with generalization guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4480–4487, 2019. 2

[24] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936. 1

[25] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017. 1, 2, 7

[26] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020. 2

[27] Maximilian Nickel and Douwe Kiela. Poincar\'e embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*, 2017. 2

[28] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017. 7

[29] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016. 2, 6, 7

[30] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):276–290, 2018. 7

[31] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6450–6458, 2019. 2, 7

[32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 2

[33] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 1

[34] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016. 1, 2

[35] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6398–6407, 2020. 7

[36] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 1

[37] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 3

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 7

[39] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*. Springer, 2020. 1, 2

[40] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. *arXiv preprint arXiv:1611.00822*, 2016. 2

[41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016. 2

[42] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[43] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1386–1393, 2014. 2

[44] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 2, 7

[45] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5207–5216, 2019. 7

[46] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009. 1

[47] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017. 7

[48] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3415–3424, 2017. 1

[49] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, volume 15, page 12. Citeseer, 2002. 1

[50] Baosheng Yu and Dacheng Tao. Deep metric learning with tuplet margin loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6490–6499, 2019. 7

[51] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 814–823, 2017. 7

[52] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 72–81, 2019. 7

[53] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is more: A deep graph metric learning perspective using fewer proxies. *arXiv preprint arXiv:2010.13636*, 2020. 7, 8