Ensemble Deep Manifold Similarity Learning using Hard Proxies

Nicolas Aziere and Sinisa Todorovic
School of EECS, Oregon State University
azieren@oregonstate.edu, sinisa@oregonstate.edu

Abstract

This paper is about deep image-similarity learning such that images of the same class have more similar deep feature representations than those belonging to different classes. For learning, prior work typically specifies loss in terms of ℓ_2-distances or dot-products between deep features, despite the well-known non-Euclidean nature of deep feature spaces. Our first contribution is in specifying the N-pair loss using a manifold similarity of deep features. We introduce a new time- and memory-efficient estimation of the manifold similarities that uses a closed-form convergence solution of the Random Walk algorithm. We randomly partition the deep feature space, and express the manifold similarities via representatives of the resulting subspaces, a.k.a. proxies. Multiple random partitions of the deep feature space gives an ensemble of proxies which can be jointly used for estimating image similarity. Our second contribution is aimed at reducing overfitting by estimating hard proxies that are as close to one another as possible, but remain in their respective subspaces. We outperform the state of the art in both image retrieval and clustering on the CUB-200-2011, Cars196, and Stanford Online Products datasets with the same complexity as related ensemble methods.

1. Introduction

This paper presents an approach to deep metric learning. Our objective is to learn deep representations of images such that images belonging to the same class have more similar representations than those belonging to different classes. This is an important problem with a wide range of applications, including image retrieval [16, 17], image clustering [18], and fine-grained image classification [4, 11].

Recent work uses a convolutional neural network (CNN) to compute the image's deep feature which satisfies the above objective. The CNN is typically trained with the contrastive loss [2], triplet loss [7, 21], or N-pair loss [17]. These loss functions are usually specified in terms of ℓ_2-distances or dot-products which are ill-suited to the highly non-Euclidean deep feature space.

Recent works address this issue by: (i) Specifying loss as a function of manifold distances of deep features [1, 8, 9]; or (ii) Partitioning the deep feature space first into subspaces, then projecting data to a new space spanned by representatives of the subspaces, and finally estimating loss in the new space [10, 12, 13, 23]. Both groups of methods have shortcomings that we seek to alleviate.

The first group of methods uses the Random Walk algorithm [24] for estimating a geodesic distance (or similarity) between data points on a manifold, referred to as manifold distance (or manifold similarity). However, incorporating Random Walk in deep learning is difficult, because the former requires access to all data whereas deep learning organizes training in mini-batches. Also, Random Walk, being iterative, significantly increases complexity of training.

Weaknesses of the second group of methods pertain to specification of the new embedding space. For example, deep features are projected onto the new space spanned by: (a) One-hot vectors [23], or (b) Randomly sampled vectors, a.k.a. proxies [12]. Both one-hot vectors and randomly sampled proxies are heuristic and not learned end-to-end.

Toward addressing the aforementioned shortcomings, we make two contributions in training of our CNN, as illustrated in Fig. 1 (left). **Our first contribution** is in specifying a time- and memory-efficient algorithm for estimating loss in terms of manifold similarities between deep features. Unlike prior work, we adapt Random Walk to estimate the manifold similarities of only a small number of data in each mini-batch of deep learning, rather than on all training data. This allows for efficiently computing the manifold similarities using the closed-form convergence solution of Random Walk, rather than running its many iterations.

Following the above second group of approaches, we randomly partition the training dataset, where each partition represents a meta-class of images. A meta-class may comprise only a part of images belonging to one class, or images from several distinct image classes. Similar to [12], we take representatives of the meta-classes to stand in as proxies for images when estimating the N-pair loss in training. Specifically, the N-pair loss on all images that got assigned to a meta-class is computed as a loss of the proxy vector repre-

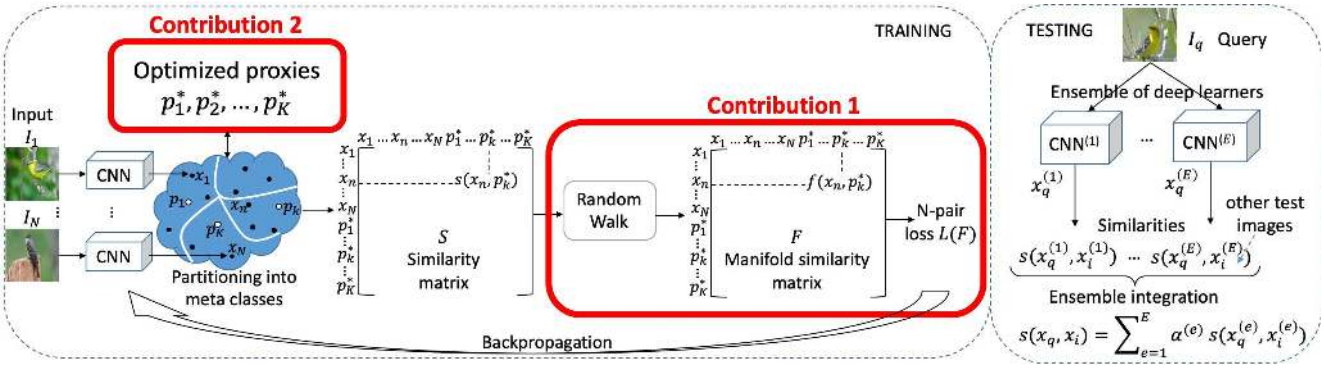

Figure 1. (Left) An overview of our two contributions in training of a CNN whose output deep representations of images x should satisfy the objective that images belonging to the same class have more similar deep representations than those belonging to different classes. (Right) Our training produces an ensemble of CNNs, where each is learned on a particular random partition of the deep feature space into meta-classes. For testing, we integrate all learners in the ensemble for computing similarity between the query and other test images.

senting that meta-class. As the proxy set is much smaller than the training dataset, the number of image triplets commonly considered when estimating the N-pair loss can be significantly reduced. Also, as shown in [12], minimizing loss expressed in terms of proxies in place of images amounts to minimizing an upper bound of the ranking loss between images, and hence effectively enforces the desired distance relationships of images.

Our second contribution is in specifying a new algorithm for estimating *hard proxies*. This is aimed at tightening the upper bound on the desired distance relationships between training images. When the related work [12, 23] randomly samples proxies, they risk overfitting to a relatively "loose" upper bound of the ranking loss between images. Rather than using random sampling, we optimize proxies so that they incur a maximum N-pair loss. Essentially, this means that we search in the deep space for proxies that are as close to one another as possible, but still belong to their respective subspaces of the deep feature space corresponding to the meta-classes. Because the proxies are made very similar to one another, violating the distance relationships between the images is made easier, and thus our end-to-end training of deep features is enforced to be more precise toward reducing these violations.

Some of the issues related to overfitting and random partitioning of the deep feature space can be overcome with ensemble learning [10, 13, 23]. We follow this line of work, and resort to multiple random splits of the deep feature space, as the optimal partitioning is unknown. Each random partitioning gives the corresponding dictionary of hard proxies that are used to define the N-pair loss for learning the CNN – that is, one learner in the ensemble.

In testing, the deep image representation is computed by concatenating or averaging the deep features produced by every CNNs of the ensemble, as illustrated in Fig. 1 (right).

Our evaluation demonstrates that we outperform the state of the art in both image retrieval and clustering on the benchmark datasets, including CUB-200-2011 [20], Cars196 [11], and Stanford Online Products [19], with the same computational complexity per one CNN in the ensemble as non-ensemble methods.

In the following, Sec. 2 reviews prior work, Sec. 3 gives an overview of our approach, Sec. 4 formulates hard proxies, Sec. 5 explains how to compute manifold similarities with Random Walk, Sec. 6 formulates two manifold loss functions, Sec. 7 describes complexity, Sec. 8 specifies implementation details, and Sec. 9 presents evaluation.

2. A Review of Related Work

Distance Metric Learning is a long-standing problem. The section reviews only the most closely related work.

Loss formulations: Loss is usually defined on triplets of images for taking into account image distances both within a class and across distinct classes in training. For example, the N-pair loss [17, 19] is computed on a mini-batch of training images comprising one anchor, one positive image from the same class of the anchor's, and many negative images from different classes. The angular loss [22] represents a variation of the N-pair loss, aimed at both minimizing the angles between intra-class features and maximizing the angles between inter-class features. The facility location function is specified to improve image clustering quality measured by normalized mutual information (NMI) rather than directly optimize image distances [18]. A class-level tree capturing intrinsic and contextual information of the dataset is used to adaptively estimate the margin in the triplet loss [5]. These loss functions are usually specified in terms of ℓ_2-distances or dot-products of deep features. In this work, we use the N-pair loss for training, and advance the related work by specifying the loss in terms of manifold similarities between deep features as more appropriate for the highly non-Euclidean deep feature space.

Addressing the large number of image triplets: Selecting optimal image triplets for more efficient and effective distance metric learning can be done, e.g., by smart mining in the large space of training triplets [7], or by adversarial metric learning on synthetic hard negatives generated from the observed negative samples [3].

The most closely related method to ours in this group uses proxies to substitute for the original data points (hence reducing the large sampling space of image triplets), such that a loss over the proxies is a tight upper bound of the original loss over the images [12]. However, they use the same or double the number of proxies as the number of images classes, whereas we use significantly fewer proxies to avoid overfitting. Also, their randomly sampled proxies are fixed, while learning enforces the distribution of deep features to choose the proxies as their cluster centers. In contrast, we optimize the proxies to become hard examples that are difficult to learn. That is, we enforce the proxies to move away from centers of their respective subspaces, and maximize the N-pair loss, in order to avoid overfitting.

Manifold distance learning: A few approaches seek to estimate a manifold structure of the image dataset for distance metric learning. To this end, they use the PageRank algorithm [24], or the diffusion process on a region manifold [1, 9]. Some of these methods also consider selection of optimal training examples via unsupervised manifold guided selection of image triplets [8]. However, all these approaches estimate the manifold distances in a post-processing step by fixing the deep features. In contrast, we integrate estimation of the manifold distances into our end-to-end training of deep features.

Ensemble Learning: Ensemble learning is aimed at reducing variance among a family of learners, which typically leads to performance improvements. In deep metric learning, each member of the ensemble votes for a distance between two points, and the final distance is estimated by integrating all of the votes. For example, the last embedding layer of a deep network can be divided into an embedding ensemble, and trained using the online gradient boosting [13]. Also, different learners can be defined using a family of attention masks, resulting in an attention-based ensemble [10]. Similar to [23], we use randomized ensembles, where each learner is defined by a particular random partitioning of the deep feature space.

3. Our Approach

Our approach learns an ensemble of CNNs, $\mathcal{E} = \{\text{CNN}^{(e)} : e = 1, \dots, E\}$, where each $\text{CNN}^{(e)}$ embeds an input image, I, to the *normalized* deep feature $x^{(e)} = \frac{\tilde{x}^{(e)}}{|\tilde{x}^{(e)}|}$, as illustrated in Fig. 1 (right). Thus, in this work, all image embeddings are normalized to the unit sphere. The CNNs in $\mathcal{E}$ have the same deep architecture, but are independently learned on a given random partitioning of the training set of images. Randomized ensembles like ours, have been shown to improve performance over individual members of the ensemble [23].

In testing, we first pass every test image I_n through $\mathcal{E}$, resulting in the deep representation $\boldsymbol{x}_n = [x_n^{(1)}, \cdots, x_n^{(E)}]$. These deep features are then used for estimating image similarities, and finally evaluation on the image retrieval or clustering problems. We specify similarity between the query image I_q and another test image I_n as

$$s(\boldsymbol{x}_q, \boldsymbol{x}_n) = \sum_{e=1}^{E} \alpha^{(e)} \; s(x_q^{(e)}, x_n^{(e)}), \tag{1}$$

where $\{\alpha^{(e)}\}$ are relative importance weights of the CNNs in the ensemble, and $s(x_q, x_n)$ is defined as the dot-product of input features:

$$s(x_q, x_n) = x_q \cdot x_n. \tag{2}$$

Note that when $\alpha^{(e)} = 1$, for $e = 1, \dots, E$, our ensemble integration amounts to concatenating all outputs of the CNNs in the ensemble for computing the image similarity in (1), $s(\boldsymbol{x}_q, \boldsymbol{x}_n) = \boldsymbol{x}_q \cdot \boldsymbol{x}_n$.

While $\{\alpha^{(e)}\}$ could be estimated on a validation set using various boosting algorithms, in our experiments, we did not observe significant differences in our performance from the case when the relative weights are all set to 1. Therefore, in this paper, we use $\alpha^{(e)} = 1$, for $e = 1, \dots, E$.

In training, we learn independently each of the CNNs in the ensemble on a given training set of images and their class labels, $\mathcal{D} = \{(I_n, y_n)\}$. After the CNN computes deep features of training images, we estimate their manifold similarity relationships. Any violations of the desired manifold similarity relationships incurs loss, which is then backpropagated for training the CNN.

Following recent approaches [17, 19, 22], in this paper, we use the smooth, differentiable N-pair loss which efficiently takes advantage of all training images in a mini-batch, rather than taking into account individual image triplets. Specifically, in [17, 19, 22], each training mini-batch consists of N samples, where one image x is called anchor, another positive image x^+ comes from the same class as the anchor, and the remaining $N-2$ negative images $\{x_n^-\}$ belong to classes that are different from the anchor's. These approaches define the N-pair loss so as to reduce similarity between the anchor x and the negatives $\{x_n^-\}$, and simultaneously increase similarity between the anchor x and the positive image x^+:

$$L_{\text{N-pair}}(x, x^+, \{x_n^-\}) = \log\left(1 + \sum_{n=1}^{N-2} e^{s(x, x_n^-) - s(x, x^+) + m}\right), \tag{3}$$

where $m \geq 0$ is a constant margin, and $s(\cdot)$ is a similarity function, e.g., given by (2).

Our extension of prior work is two-fold. Our N-pair loss uses manifold similarity instead of their similarity, and an optimized set of proxies in place of actual training images. Before we specify our N-pair loss in Sec. 6, we first describe how to estimate the proxies in Sec. 4, and how to compute manifold similarities of a training mini-batch in Sec. 5.

4. Hard Proxies

We seek to address the following two challenges in our CNN training: (1) How to accurately estimate similarity between images for computing the N-pair loss given by (3) in the highly non-Euclidean deep feature space; and (2) How to efficiently select optimal training images for (3) from the large sampling space of image triplets.

4.1. Proxy N-pair Loss

To address the first challenge, we follow [23] and randomly partition the training dataset $\mathcal{D}$ into K disjoint subsets, $\mathcal{D} = \cup_{k=1}^{K} \mathcal{D}_k$, where K is significantly less than the number of image classes (e.g., 10%). We expect that deep features of each partition $\mathcal{D}_k$ will exhibit properties closer to a Euclidean space than the entire deep space of D. Images in $\mathcal{D}_k$ may come from one or more classes, and we say that $\mathcal{D}_k$ defines a meta-class. Under such a partitioning, we generalize the notion of positive and negative images of an anchor, mentioned in Sec. 3. Specifically, for an anchor image x from $\mathcal{D}_k$, positive images x^+ belong to the same subset $\mathcal{D}_k$, and negative images x^- belong to the other subsets $\mathcal{D}_j$, $j \neq k$.

For the second challenge, we use a similar strategy as that introduced in [12]. In every subset $\mathcal{D}_k$, we randomly select an image to represent this meta-class, and use its *normalized* deep feature $p_k = \frac{\tilde{p}_k}{|\tilde{p}_k|}$ as a proxy for all other images in $\mathcal{D}_k$ when estimating the N-pair loss. In this way, we form the initial set of proxies $\mathcal{P} = \{p_k : k = 1, \ldots, K\}$.

After obtaining $\mathcal{P}$, we estimate the proxy N-pair loss $L_{\mathcal{P}}(x, x^+, \{x_n^-\})$ in a similar way to the expression in (3). For an anchor image x in $\mathcal{D}_k$, we replace its positive x^+ with p_k, and the negatives $\{x_n^-\}$ with their respective proxies $\{p_j\}$, $j \neq k$, resulting in the proxy N-pair loss:

$$L_{\mathcal{P}}(x, x^+, \{x_n^-\}) = \log\Big(1 + \sum_{j=1, j\neq k}^{K} e^{s(x,p_j)-s(x,p_k)+m}\Big). \tag{4}$$

From (4), we effectively alleviate the issue of optimal selection of the positive and negative images for the training mini-batch, since the loss depends only of the anchor image and the significantly fewer proxies than the number of original image classes, $L_{\mathcal{P}}(x, x^+, \{x_n^-\}) = L_{\mathcal{P}}(x)$.

Importantly, the proxy loss $L_{\mathcal{P}}$ in (4) maintains the characteristics of the N-pair loss $L_{\text{N-pair}}$ in (3) and enforces the desired similarity relationships between training images. This is because, our $L_{\mathcal{P}}$ represents an upper bound of $L_{\text{N-pair}}$, so minimizing $L_{\mathcal{P}}$ effectively reduces $L_{\text{N-pair}}$. This is straightforward to show by following very similar derivation steps to those presented in [12]. From (3)–(4) and the triangle inequality, an absolute difference of the two losses $\Delta_L = |L_{\text{N-pair}} - L_{\mathcal{P}}|$ for an image triplet (x, x^+, x^-) can be bounded as

$$\begin{aligned} \Delta_L &= \left|\log \frac{1 + e^{s(x,x^-)-s(x,x^+)+m}}{1 + e^{s(x,p_j)-s(x,p_k)+m}}\right|, \\ &\approx |[s(x,x^-) - s(x,x^+)] - [s(x,p_j) - s(x,p_k)]|, \\ &= |[d(x,x^+) - d(x,x^-)] - [d(x,p_k) - d(x,p_j)]|, \\ &\leq 2\epsilon, \end{aligned} \tag{5}$$

where we define feature distance $d(x, x') = 1 - s(x, x')$ as all our features are normalized to the unit sphere, and $\epsilon = \max_x d(x, p(x))$, and $p(x)$ is the proxy of x. It follows that the expectation of the N-pair loss can be bounded over training images as

$$E[L_{\text{N-pair}}] \leq E[L_{\mathcal{P}}] + Pr[|d(x,p_k) - d(x,p_j)| \leq 2\epsilon]. \tag{6}$$

Since our deep features and proxies are normalized to the unit sphere, the upper bound in (6) is tight.

4.2. Estimation of Hard Proxies

In this work, the initial set of images selected as proxies is fixed for the entire duration of our training. But, in every epoch of training, we first recompute their deep features $\mathcal{P}$, and then estimate an optimal set of proxies $\mathcal{P}^*$ for computing the optimal proxy loss $L_{\mathcal{P}^*}$ over all training mini-batches in the next epoch.

This is our main difference from prior work [12], since their proxies $\mathcal{P}$ remain unchanged in learning, and their CNN is trained to produce deep features which cluster well around the proxies with respect to a distance metric. In our experiments, however, we observe that this leads to overfitting, because, in part, clustering in the non-Euclidean deep space using a distance metric gives suboptimal results. Another reason for overfitting comes from the random outcome of selecting proxies, which may make the proxy loss $L_{\mathcal{P}}$ a looser upper bound of the N-pair loss $L_{\text{N-pair}}$. Hence, minimizing such a $L_{\mathcal{P}}$ may have little effect on enforcing the desired similarity relationships between training images.

To address overfitting, our key idea is to estimate optimal proxies, $\mathcal{P}^* = \{p_k^* : k = 1, \ldots, K\}$, so as to maximally reduce the difference in (6), $|d(x,p_k^*) - d(x,p_j^*)|, j \neq k$, and in this way make $L_{\mathcal{P}^*}$ a tighter upper bound of $L_{\text{N-pair}}$ than the initial $L_{\mathcal{P}}$. One way to achieve this is to make all proxies *similarly distant* from all data points in the deep feature space, resulting in ideal $|d(x,p_k) - d(x,p_j)| \approx 0$, for all x and $p_k \neq p_j$.

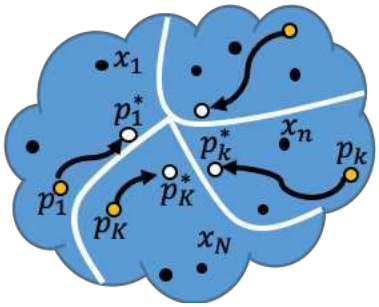

Figure 2. Optimization of proxies: the white lines mark meta-classes, black dots represent images x_n, yellow dots mark the initial randomly sampled proxies p_k, and white dots indicate the estimated hard proxies p_k^*. Our optimization "pushes" p_k^* away from images x_n it represents toward the other meta-classes, while regularizes that p_k^* remains close to p_k.

As illustrated in Fig. 2, this objective can be achieved by *maximizing* distances between p_k^* and images in $\mathcal{D}_k$, for every meta-class $\mathcal{D}_k$. In turn, this will make every p_k^*, $k = 1, \ldots, N$, closer to the other meta-classes. To avoid a trivial solution, where all proxies are equal, we regularize this objective such that the optimal p_k^* is not too far from the initial p_k. This gives the following optimization for every meta-class $\mathcal{D}_k$:

$$p_k^* = \arg\min_{p \in \mathbb{R}^l} \log\Big(1 + \sum_{x_n \in \mathcal{D}_k \setminus \{p_k\}} e^{s(p,x_n) - s(p,p_k)}\Big), \quad (7)$$

where l is the length of deep features, the first term in the exponent "pushes" the optimal proxy away from images x_n it represents toward other meta-classes, and the second term in the exponent regularizes against trivial solutions. We efficiently solve (7) with gradient descent, with the initial value set to p^k of a randomly selected image in $\mathcal{D}_k$.

One consequence of making every p_k^* become closer to the other meta-classes $\mathcal{D}_j$, $j \neq k$, is that all $p_k^* \in \mathcal{P}^*$ become close to each other. This makes minimization of the optimal proxy loss $L_{\mathcal{P}^*}$ given by (4) difficult, because the CNN has to produce more accurate deep features for respecting the desired similarity relationships that x from $\mathcal{D}_k$ should be more similar to p_k than to the other proxies p_j, $j \neq k$. Therefore, we call $\mathcal{P}^*$ the set of hard proxies.

5. Manifold Similarity Estimation

We have empirically observed that estimating similarities between images and the proxies in (4), $\exp(s(x, p_j^*) - s(x, p_k^*) + m)$, $j \neq k$, often gives inaccurate results, because meta-classes $\{\mathcal{D}_k\}$ are highly non-convex sets in the deep feature space. Therefore, rather than using the dot-product for estimating $s(x_n, p_k^*)$, our next contribution is to estimate geodesic similarities, $\{f_{np_k^*} = f(x_n, p_k^*) : k = 1, \ldots, K\}$, on a manifold, which we call manifold similarities, and

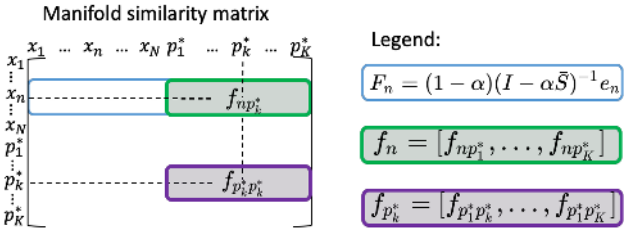

Figure 3. Illustration of vectors of manifold similarities F_n, f_n, and $f_{p_k^*}$, where we use f_n and $f_{p_k^*}$ for estimating the manifold proxy loss.

use them for computing the manifold proxy loss. Importantly, our key novelty is in estimating $\{f(x_n, p_k^*)\}$ for every mini-batch of our end-to-end training, rather than computing manifold similarities pre- or post-training on all data, as common in prior work [1, 8, 9].

We compute the manifold similarities of images in a training mini-batch $\mathcal{B}$ with the Random Walk algorithm [24] on a nearest neighbor graph. Each $\mathcal{B}$ is constructed from N training images and K hard proxies:

$$\mathcal{B} = \{x_1, \ldots, x_n, \ldots, x_N, p_1^*, \ldots, p_k^*, \ldots, p_K^*\}. \quad (8)$$

For $\mathcal{B}$, we first compute the $(N+K) \times (N+K)$ symmetrically normalized adjacency matrix, $\bar{S} = D^{-1/2} S D^{-1/2}$, where elements of S are dot-products $s(x_n, x_{n'}) = x_n \cdot x_{n'}$ or $s(x_n, p_k^*) = x_n \cdot p_k^*$, and D is the diagonal degree matrix with elements $D(n,n)$ equal to a sum of the nth row in S. As similarity of each x_n or p_k^* to itself is irrelevant (and also to avoid loops in Random Walk), we set all diagonal elements of $\bar{S}$ to 0. Then, for each image $x_n \in \mathcal{B}$, we estimate a vector of its manifold similarities to images in $\mathcal{B}$, $F_n = [f_{n1}, \ldots, f_{nN}, f_{np_1}, \ldots, f_{np_K}]$, using the closed-form convergence solution of Random Walk [24] as

$$F_n = (1-\alpha)(I - \alpha\bar{S})^{-1} e_n \quad (9)$$

where $\alpha \in (0,1)$ is a probability of restarting Random Walk from nth query point, e_n is the query one-hot vector with 1 at nth location, and I is the $(N+K) \times (N+K)$ identity matrix. Since the size of mini-batch $(N+K)$ is relatively small, designed to fit the available RAM memory, computing the inverse matrix in (9) can be done efficiently.

As explained in the next section, the manifold proxy loss is defined in terms of f_n and $f_{p_k^*}$, which are parts of their respective vectors F_n and $F_{p_k^*}$. f_n and $f_{p_k^*}$ consist of only manifold similarities of x_n and p_k^* to the proxies, respectively, as illustrated in Fig. 3:

$$\begin{aligned} f_n &= [f_{np_1^*}, \ldots, f_{np_k^*}, \ldots, f_{np_K^*}], \\ f_{p_k^*} &= [f_{p_k^*p_1^*}, \ldots, f_{p_k^*p_k^*}, \ldots, f_{p_k^*p_K^*}]. \end{aligned} \quad (10)$$

6. Two Manifold Proxy Loss Functions

We extend the optimal proxy loss $L_{\mathcal{P}^*}$ given by (4) to account for manifold similarities between images and the hard proxies in a mini-batch of N training images. In this paper, we consider two extensions: (1) Intrinsic L_{int}, and (2) Contextual L_{cxt} loss functions.

The intrinsic manifold proxy loss is computed as

$$L_{\text{int}}(\{x_n\}) = \frac{1}{N}\sum_{n=1}^{N}\log\Big(1+\sum_{\substack{j=1,\\ j\neq k}}^{K} e^{f_{np_j^*}-f_{np_k^*}+m}\Big), \quad (11)$$

where p_k^* is the hard proxy of image x_n if $x_n \in \mathcal{D}_k$, and f_{np^*} are elements of the manifold similarity vector f_n given by (10). Having the similar formulation as $L_{\text{N-pair}}$ and $L_{\mathcal{P}^*}$, the intrinsic loss $L_{\text{int}}(\{x_n\}_1^N)$ in (11) inherits advantages of the N-pair loss and the proxy loss in addressing the large sampling space of image triplets, discussed in Sec. 3.

The contextual manifold loss introduces additional constraints relative to L_{int} for a stronger enforcement of the desired similarity relationships. We additionally constrain that each image $x_n \in \mathcal{D}_k$ "sees" the set of hard proxies P^* as its proxy p_k^* "sees" P^*. That is, if P^* represents well the entire set of data $\mathcal{D}$, then the desired similarity relationships between every $x_n \in \mathcal{D}_k$ and other images in $\mathcal{D}$ should be close to the similarity relationships between p_k^* and P^*.

$$L_{\text{cxt}}(\{x_n\}) = \frac{1}{N}\sum_{n=1}^{N}\log\Big(1+\sum_{\substack{j=1,\\ j\neq k}}^{K} e^{s(f_n,f_{p_j^*})-s(f_n,f_{p_k^*})+m}\Big), \quad (12)$$

where p_k^* is the hard proxy of image x_n if $x_n \in \mathcal{D}_k$, f_n and $f_{p_k^*}$ are given by (10), and $s(f_n, f_{p_k^*}) = f_n \cdot f_{p_k^*}$.

7. Complexity Analysis

We are given a training dataset, $\mathcal{D}$, with size $|\mathcal{D}| = M$, and M^3 image triplets for computing the triplet loss. We use E random partitions of $\mathcal{D}$ into K meta classes for ensemble learning. The E distinct CNNs are trained in parallel, so our runtime actually does not increase E times.

Our training complexity per one epoch of a single CNN is derived as follows. As illustrated in Fig. 1, we have three main computational steps. First, we optimize K proxies over M data by minimizing Eq. 7, which amounts to $O(MK)$. Second, in each iteration over $\frac{M}{N}$ batches, where N is the mini-batch size, we compute an inverse of the (N+K)x(N+K) manifold-similarity matrix, which amounts to $O(\frac{M}{N}(N+K)^3) = O(MK^2)$, as we set $K \approx \frac{N}{2}$ (not sensitive, see Sec. 8). Third, we compute the N-pair loss over M training data and K proxies, which amounts to $O(MK^2)$. Finally, our total complexity is $O(MK) + O(MK^2) + O(MK^2) = O(MK^2)$.

Tab. 1 compares our training complexity with that of the state of the art. As can be seen, our training complexity is the same as that of [23], and higher than that of [12] only for the additional ensemble learning.

Methods	Training Complexity
N-Pair [17]	$O(M^3)$
Proxy-NCA [12]	$O(MK^2)$
DREML [23]	$O(EMK^2)$
Our approach	$O(EMK^2)$

Table 1. Training complexity of the most related methods to ours.

8. Implementation Details

For implementation, we use Pytorch [14]. In preprocessing, images are normalized and re-sized to 256×256 pixels. We used standard data augmentation techniques including random image cropping and rotation. The mini-batch size is set to $N = 128$. The size of our ensemble of CNNs is $E = 25$. The Random Walk parameter $\alpha = 0.8$, and the margin $m = 5\times10^{-4}$. For the hard proxy optimization, the learning rate is set to 10^{-3}. The empirically-found best number of proxies (one per meta-class) is $K = 50$ for all datasets. For comparison with prior work, we implemented both ResNet18 and GoogLeNet pre-trained on ImageNet [15]. The last fully-connected layer is modified to set the dimension of deep features $l = 128$. We used an Adam optimizer parameterized with the weight decay factor of 10^{-5}. The learning rate is initialized at 10^{-4}, and decreased by a factor of 0.1 every three epochs, over a total of 10 epochs per training. Overall, the training of one CNN takes about 1 hour on a Tesla K80 GPU.

9. Results

Datasets: Evaluation is performed on the image retrieval and clustering problems using the following three benchmark datasets. *CUB* 200-2011 [20] has 11,788 images showing 200 bird classes. The data is split into 5,864 images of the first 100 classes are used for training, and 5,924 images of the remaining classes for testing. *Cars196* [11] has 16,185 images of 196 car classes. The data is split into 8,054 training images of the first 96 classes and 8,131 test images of the remaining car classes. *Stanford Online Product* [19] has 120,053 images with 22,634 classes. The data is split into 59,551 training images of the first 11,318 classes and 60,502 testing images of the remaining classes from the dataset. The aforementioned training-test splits are standard and used by all prior work that we compare with.

Evaluation Metrics: In image retrieval, given a query test image, we find its K nearest neighbors from the test set. We calculate a percentage of the retrieved images $R@K$

that have the same class as the query. Image clustering is performed using the K-means algorithm, where K is the number of image classes, and evaluated with the normalized mutual information (NMI). For the obtained set of clusters, $\hat{\Omega} = \{\hat{\omega}_1, ..., \hat{\omega}_K\}$, and the ground-truth image clustering by their classes, $\Omega = \{\omega_1, ..., \omega_K\}$, NMI is defined as $\text{NMI}(\hat{\Omega}, \Omega) = \frac{2I(\hat{\Omega},\Omega)}{H(\hat{\Omega})+H(\Omega)}$, where $I(\cdot)$ denotes the mutual information, and $H(\cdot)$ is entropy.

Ablation Study: We test performance effects of various components of our approach, especially our claimed contributions, using the following variants of our approach:

- EDMS (RW, P*) is our full approach Ensemble Deep Manifold Similarity learning using Random Walk (RW) and the hard proxies for estimating L_{cxt}. For testing, we use the dot-product similarity, where the ensemble of our CNNs is fused as specified in (1) (see Sec. 3 and Fig. 1).
- EDMS (RW, P) does not optimize the proxies, but computes L_{cxt} using the initial P, and thus tests the effect of using P^* vs. P on performance.
- EDMS (RW-int, P*) replaces L_{cxt} with L_{int}, and thus evaluates the contextual vs. intrinsic manifold loss.
- EDMS (P*) replaces L_{cxt} with $L_{\mathcal{P}^*}$ given by (4), and thus tests the effect of Random Walk on performance.
- EDMS (w/o) replaces L_{cxt} with $L_{\text{N-pair}}$ given by (3). We still perform random partitioning of the training set, and compute $L_{\text{N-pair}}$ with respect to meta-classes.
- EDMS (RW) is similar to EDMS (w/o) but computes $L_{\text{N-pair}}$ with manifold similarities $f(x_n, x_{n'})$ with respect to meta-classes.

Baselines: We compare with with the following state-of-the-art approaches using the same testing setup as theirs. *Proxy-NCA* [12] uses the proxies to estimate the NCA loss [6]. *Lifted Structure* [19], *N-pair* [17] and *Angular* [22] use the N-pair loss. *BIER* [13], *ABE* [10] and *DREML* [23] use ensemble learning. For comparison with *DREML* [23], we use their latest results published on arXive and linked on the Github page.

Method	NMI	R@1	R@2	R@4	R@8
EDMS (w/o)	66.4	58.7	71.4	81.2	89.4
EDMS (RW)	63.4	56.1	69.2	79.5	87.6
EDMS (RW, P)	66.8	60.3	71.5	81.3	89.1
EDMS (RW-int, P*)	66.9	61.1	72.2	81.7	89.3
EDMS (P*)	67.2	63.7	74.2	82.9	89.7
EDMS (RW, P*)	**68.9**	**66.1**	**76.7**	**85.5**	**91.4**

Table 2. Our ablation study on the CUB dataset: Image clustering and retrieval results for different variants of our approach using ResNet18.

9.1. Quantitative Results

Fig. 4 shows how performance of our EDMS (RW, P*) changes as a function of the ensemble size, number of proxies, and deep-feature dimension on CUB-200-2011. We observe on Fig. 4 (left) that the accuracy saturates after a certain ensemble size, and as a good trade off between complexity and accuracy we set the number of CNNs in the ensemble $E = 25$. From Fig. 4 (right), we get the best results for $K = 50$ proxies, and deep-feature dimension of $l = 128$. We use these parameters for all variants of our approach on all three datasets.

Tab. 2 presents our ablation study with the six variants of our approach on CUB-200-2011. For EDMS (RW) our recall decreases relative to EDMS (w/o) when no proxies are used. This suggests that our estimation of manifold similarities on a relatively small mini-batch may not be able to reliably capture the true geodesic distances between images without the help of the proxies. This is further seen in EDMS (RW, P), where by adding the proxies to EDMS (RW), we get performance improvement over both EDMS (w/o) and EDMS (RW). A good performance of EDMS (P*) suggests that our random partitioning of the training set and the use of proxies help estimate image similarities reliably, even without Random Walk. Using the contextual loss in EDMS (RW, P*) gives much better recall than using the intrinsic loss in EDMS (RW-int, P*). Finally, our optimization of hard proxies in EDMS (RW, P*) improves performance relative to that of EDMS (RW, P).

Tables 3 and 4 compare our best performing EDMS (RW, P*) with the baselines. When using ResNet18 as the CNN, we outperform the state-of-the-art ensemble learning methods BIER, ABE and DREML. We observe that GoogLeNet gives lower results than ResNet18 for our approach.

From Fig. 4, we can use small $E < 5$, and still significantly outperform [12] (see Tab. 3). Our gains in performance over non-ensemble-learning methods justify the slight increase in runtime for fusing E CNNs.

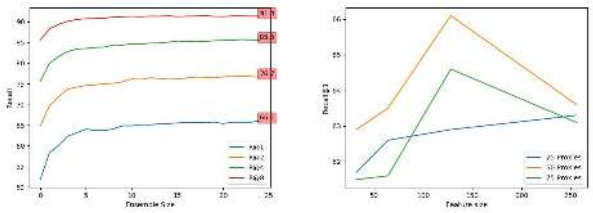

Figure 4. Optimal selection of the ensemble size, number of proxies, and deep-feature dimension for EDMS (RW, P*) on CUB-200-2011 with respect to Recall. (Left) Our recall as a function of the ensemble size. (Right) Recall@1 as a function of the number of proxies, and size of deep features.

Dataset		CUB-200-2011					Car196				
Method	Network	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8
Lifted [19]	GoogLeNet	55.38	47.2	58.9	70.2	80.2	55.1	48.3	61.1	71.8	81.1
Proxy-NCA [12]	InceptionBN	59.5	49.2	61.9	67.9	72.4	64.9	73.2	82.4	86.4	88.7
N-pair [17]	GoogLeNet	60.4	51.0	63.3	74.3	83.2	64.0	71.1	79.7	86.5	91.6
Angular [22]	GoogLeNet	61.1	54.7	66.3	76.0	83.9	63.2	71.4	81.4	87.5	92.1
BIER [13]	GoogLeNet	-	55.3	68.4	76.9	85.1	-	78.0	85.8	91.1	95.1
ABE [10]	GoogLeNet	-	60.6	71.5	79.8	87.7	-	85.2	90.5	94.0	96.1
DREML [23]	ResNet18	67.8	63.9	75.0	83.1	89.7	76.4	86.0	91.7	95.0	97.2
EDMS (RW, P*)	GoogLeNet	64.5	61.6	72.1	81.8	88.9	75.1	85.6	90.8	94.8	96.1
EDMS (RW, P*)	ResNet18	**68.9**	**66.1**	**76.7**	**85.5**	**91.4**	**76.7**	**87.6**	**92.1**	**95.2**	**97.3**

Table 3. Image clustering and retrieval results on the CUB-200-2011 and Cars196 datasets.

Method	NMI	R@1	R@10	R@100	R@1000
Lifted [19]	87.4	63.0	80.5	91.7	97.5
N-pair [17]	87.9	67.7	83.8	93.0	97.8
Angular [22]	88.6	70.9	85.0	93.5	98.0
BIER [13]	-	74.2	86.9	94.0	97.8
ABE [10]	-	76.3	88.4	94.8	98.2
Ours+G	**89.0**	**77.2**	**89.1**	**94.9**	**98.1**
Ours+R	**90.1**	**78.5**	**90.7**	**95.2**	**98.5**

Table 4. Image clustering and retrieval results on Stanford Online Products. All the competing approaches use the GoogLeNet. Ours+G = EDMS (RW, P*) with GoogLeNet, and Ours+R = EDMS (RW, P*) with ResNet18.

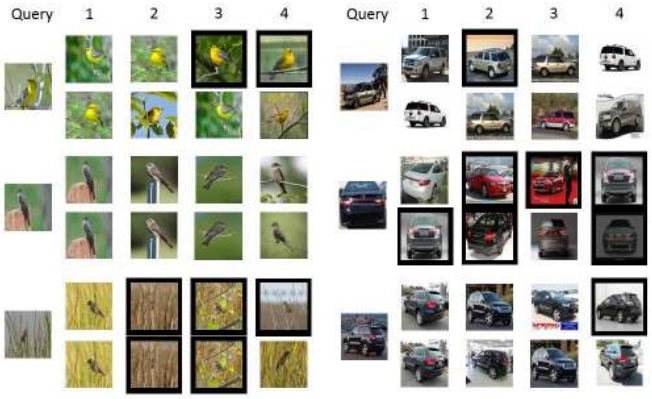

Figure 5. Our sample retrieval results on the CUB-200-2011 (Left) and Car196 datasets (Right). For each query, we show 4 top retrieved images, where the top row shows the results with EDMS (P*), and the bottom row shows the results with EDMS (RW, P*). Using manifold similarities gives more visually accurate results. Errors in our retrieval are highlighted with the black frame.

9.2. Qualitative Results

Fig. 5 shows a few sample retrieval results on CUB-200-2011 and Car196 for EDMS (P*) and EDMS (RW, P*). As can be seen, using manifold similarities in EDMS (RW, P*) gives more visually accurate results not only in terms of the correctly retrieved images from the same class as the query, but also regarding the 3D pose and background. There are also some failure cases, which seem to be due to confusing foreground and background in the images. This could be addressed in the future by incorporating recent visual attention techniques.

10. Conclusion

We have presented a new approach to ensemble learning of deep image representations that should respect their desired similarity relationships within and across image classes. Toward addressing the non-Euclidean properties of the deep feature space, we have made two key contributions in training. First, we have specified two new loss functions, called contextual and intrinsic manifold loss, in terms of geodesic similarity of images on a manifold, which is efficiently estimated for each training mini-batch using the closed-form solution of Random Walk. For computing our manifold loss, training images are partitioned into subsets, and their manifold similarity is estimated via randomly selected representatives of the subsets, called proxies. Our second contribution pertains to optimizing the proxies such that the proposed manifold loss enforces stronger constraints on learning of the desired similarity relationships. We have presented an ablation study and comparison with the state of the art on image retrieval and clustering using the CUB-200-2011, Cars196, and Stanford Online Products datasets. Our results suggest that estimation of manifold similarities on a relatively small mini-batch may not be able to reliably capture the true geodesic distances between images without the help of the proxies. Also, our random partitioning of the training set and the use of proxies help estimate image similarities reliably, leading to a competitive performance, even without Random Walk. Our full approach outperforms the state of the art on both image retrieval and clustering, on all three datasets.

Acknowledgment

This work was supported in part by DARPA XAI Award N66001-17-2-4029.

References

[1] S. Bai, Z. Zhou, J. Wang, X. Bai, L. J. Latecki, and Q. Tian. Ensemble diffusion for retrieval. In *ICCV*, pages 774–783, 2017. 1, 3, 5

[2] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005. 1

[3] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018. 3

[4] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, volume 2, page 3, 2017. 1

[5] W. Ge, W. Huang, D. Dong, and M. R. Scott. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 2

[6] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004. 7

[7] B. Harwood, V. K. BG, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. *space*, 9(13):22. 1, 3

[8] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Mining on manifolds: Metric learning without labels. *arXiv preprint arXiv:1803.11095*, 2018. 1, 3, 5

[9] A. Iscen, G. Tolias, Y. S. Avrithis, T. Furon, and O. Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *CVPR*, volume 1, page 4, 2017. 1, 3, 5

[10] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon. Attention-based ensemble for deep metric learning. *arXiv preprint arXiv:1804.00382*, 2018. 1, 2, 3, 7, 8

[11] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 1, 2, 6

[12] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. *arXiv preprint arXiv:1703.07464*, 2017. 1, 2, 3, 4, 6, 7, 8

[13] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. BIER–boosting independent embeddings robustly. In *International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3, 7, 8

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 6

[15] 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*, 115(3):211–252, 2015. 6

[16] O. Seddati, S. Dupont, and S. Mahmoudi. Quadruplet networks for sketch-based image retrieval. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 184–191. ACM, 2017. 1

[17] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 1, 2, 3, 6, 7, 8

[18] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[19] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 6, 7, 8

[20] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 6

[21] J. Wang, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, et al. Learning fine-grained image similarity with deep ranking. *arXiv preprint arXiv:1404.4661*, 2014. 1

[22] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2612–2620. IEEE, 2017. 2, 3, 7, 8

[23] H. Xuan, R. Souvenir, and R. Pless. Deep randomized ensembles for metric learning. *arXiv preprint arXiv:1808.04469*, 2018. 1, 2, 3, 4, 6, 7, 8

[24] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Advances in neural information processing systems*, pages 169–176, 2004. 1, 3, 5