Deep Embedding Learning with Discriminative Sampling Policy

Yueqi Duan[1,2,3], Lei Chen[1,2,3,4], Jiwen Lu[1,2,3,*], Jie Zhou[1,2,3]
[1]Department of Automation, Tsinghua University, China
[2]State Key Lab of Intelligent Technologies and Systems, China
[3]Beijing National Research Center for Information Science and Technology, China
[4]School of Electrical and Information Engineering, Tianjin University, China
duanyq14@mails.tsinghua.edu.cn; chen_lei@tju.edu.cn; lujiwen@tsinghua.edu.cn;
jzhou@tsinghua.edu.cn

Abstract

Deep embedding learning aims to learn a distance metric for effective similarity measurement, which has achieved promising performance in various tasks. As the vast majority of training samples produce gradients with magnitudes close to zero, hard example mining is usually employed to improve the effectiveness and efficiency of the training procedure. However, most existing sampling methods are designed by hand, which ignores the dependence between examples and suffer from exhaustive searching. In this paper, we propose a deep embedding with discriminative sampling policy (DE-DSP) learning framework by simultaneously training two models: a deep sampler network that learns effective sampling strategies, and a feature embedding that maps samples to the feature space. Rather than exhaustively calculating the hardness of all the examples for mining through forward-propagation, the deep sampler network exploits the strong prior of relations among samples to learn discriminative sampling policy in a more efficient manner. Experimental results demonstrate faster convergence and stronger discriminative power of our DE-DSP framework under different embedding objectives.

1. Introduction

Embedding learning for effective distance metric estimation has aroused much attention over the past few decades [5, 10, 36, 16]. With the recent success of deep learning [18, 23, 12, 15], deep embedding learning methods present strong discriminative power in various tasks due to the high nonlinearity, such as visual search [31, 30, 35], biometric verification [3, 25, 1] and zero-shot learning [2, 41]. The basic goal of deep embedding learning is to minimize intra-class variations and maximize inter-class distances, where numbers of objectives have been presented in the literature [3, 25, 33, 4, 29, 31, 30, 22, 35], including the most commonly-used contrastive loss [3] and triplet loss [25].

While the sample sizes are usually quadratic or cubic in deep embedding learning, the meaningful hard examples only account for the tiny minority. Training with numerous easy examples may suffer from inefficiency and poor performance as they contribute little to the training process by producing gradients with magnitudes close to zero [29, 14]. To this end, several hard example mining approaches have been proposed for effective sample selection [25, 42, 38, 11, 40, 39]. Recent studies have shown that sampling plays an equal or even more crucial role than the objective function in deep embedding learning [38].

An ideal sampling strategy should be *targeted* and *adaptive*. On one hand, the selected samples should target at the requirements of the current state of embedding. On the other hand, the sampling strategy should be adaptively updated during the training process of deep embedding learning. Unfortunately, it is not easy to achieve the goals for most existing sampling methods designed by hand as a huge number of candidates are required to perform forward-propagation and exhaustive search (to be *targeted*) at every step (to be *adaptive*) of deep embedding learning. To avoid the infeasible computation, most exsiting sampling strategies employ an online method by selecting effective examples within a small mini-batch as a suboptimal solution [25, 38]. However, limited sampling space may lead to slow convergence and poor local optima.

In this paper, we consider that exhaustive search is far from required for sampling. For example, our humans only need a few attempts to acquire that there are large inter-class distances between two specific classes with 100 samples in each class, which are not probably able to construct hard triplets. However, exhaustive search suffers from 1,980,000 ($200 \times 99 \times 100$) hardness calculations. It should be noted

* Corresponding author

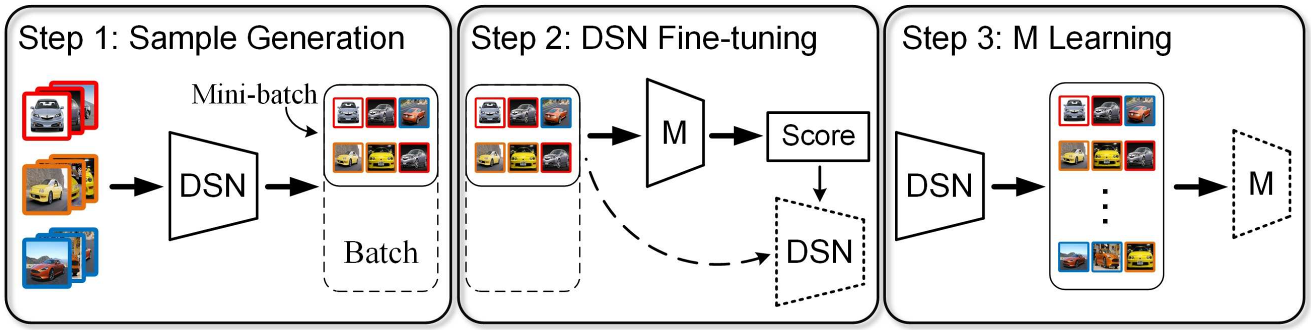


Figure 1. An overview of the proposed DE-DSP framework with the widely-used triplet embedding for an easy illustration. In the figure, DSN is the deep sampling network, and M represents the feature embedding. Deep models in solid lines indicate fixed parameters, and the ones in dashed lines are fine-tuned. We first select a mini-batch of samples with high posibilities through the deep sampling network, where the size of the mini-batch is better selected as an integral divisor of the embedding batchsize. Then, we receive the scores of the samples from the feature embedding through forward-propagation, which are utilized to fine-tune the deep sampler network. We iteratively perform Step 1 and Step 2 until having generated enough triplets for the whole batch. Lastly, we train the feature embedding with the batch of training samples.

that each hardness calculation requires to perform forward-propagation through the deep embedding network, which is usually very deep to achieve strong discriminative power. The main reason is that our humans have the ability to capture the relations among candidates, while searching based methods process each candidate independently. To this end, we propose a deep embedding with discriminative sampling policy (DE-DSP) learning framework, where we design a deep sampler network (DSN) to learn effective sampling strategies rather than exhaustive search. With the learned sampling policy, we obtain the possibility for each candidate sample of being selected, and the deep embedding provides the scores for the selected samples to optimize the sampling policy. Figure 1 shows an overview of the proposed DE-DSP. We observe that the number of hardness calculation is equal to the batchsize, which is independent to the candidate size and much less than exhaustive search. Compared with the existing hand-crafted sampling methods, there are two key advantages of DE-DSP:

1) To our best knowledge, it is the first attempt to learn a deep neural network for discriminative sampling policy. Compared with existing searching based methods which are especially designed for a few losses by hand (usually contrastive and triplet losses), DE-DSP is more general to various embedding structures and does not require strong prior sampling knowledge of human by directly consuming the embedding objective as the reward function. Moreover, DSN provides the possibilities of candidates of being chosen rather than a straightforward decision. As only choosing hard samples may be sub-optimal [38], easy samples will still have chances to be selected by DSN.

2) Existing methods assume that the hardness of each candidate is independent and have to perform forward-propagation on all (or a subset of) the examples through a very deep embedding network. Instead, DSN learns an effective policy with only a few score tests, which can be generalized to those untested samples. We consider that DSN possesses such generalization ability as it exploits the relation of distribution consistency between the tested and untested samples as strong prior, so that the untested samples obtain reasonable score estimations through the policy learned from the tested ones. For example, an untested sample will obtain a close score estimation to a similar tested sample. Also, while most existing sampling methods are independent at varying training steps and have to re-compute all the distances at each step, DSN has stronger adaptability as we can simply update the parameters based on the last step.

Due to the advantages above, DE-DSP is able to select more effective training samples from larger candidate size. Moreover, at the end of the training process where most samples are invalid for embedding learning, DE-DSP is more likely to capture the remaining effective samples with the discriminative sampling policy. As a general framework, we develop the widely-used triplet loss [25] and the state-of-the-art N-pair loss [29] to demonstrate the effectiveness of the proposed DE-DSP. Experimental results show that DE-DSP successfully boosts the performance of the original objectives and outperforms existing hand-crafted sampling methods on the CUB-200-2011 [34], Cars196 [17], Stanford Online Products [31] and In-Shop Clothes Retrieval [21] datasets.

2. Related Work

Metric Learning: Metric learning aims to obtain effective similarity measurements between input samples, where great progress has been made over the past few decades. Conventional methods learn a linear Mahalanobis distance to replace the simple Euclidean distance [26, 27, 9, 5, 10, 36, 19, 16]. For example, Weinberger *et al.* [36] separated samples from different classes by a large margin for k-nearest neighbor classification. Jain *et al.* [16] studied low-dimensional metrics as a regularization including low-rank and sparse metrics. While kernel tricks are employed to address the nonlinear correlations of samples [37, 7], recent deep metric learning methods present stronger discriminative power with the development of deep learning [3, 25]. Contrastive loss [3] focused on the absolute distances within a margin for the pairwise input samples. Triplet loss [25] constructed triplet input samples to ensure the relative distance ordering between positives and negatives. In recent years, several effective objectives have been proposed by constraining on more samples [33, 14, 31, 29, 8]. Representative methods include histogram loss [33] and position-dependent deep metric (PDDM) [14] for quadruplets, and lifted structure [31] and N-pair loss [29] for the whole batches. However, most training examples contribute little for deep metric learning and data sampling is required to enhance the effectiveness and efficiency.

Hard Example Mining: Hard example mining is widely applied to many tasks for effective model training [28, 42, 38, 6, 20]. In deep embedding learning, hard example mining acts as bootstrapping by gradually selecting hard samples in the embedding space [28, 11]. For example, Schroff *et al.* [25] trained FaceNet with "semi-hard" triplets, where negative-anchor pairs had small distances but still farther than positive-anchor pairs. Harwood *et al.* [11] presented a smart mining strategy to improve the efficiency of data sampling. Yuan *et al.* [40] adaptively trained the feature embedding at multiple hard levels in a hard-aware deeply casaded (HDC) manner. Wu *et al.* [38] proposed a margin based loss for distance weighted sampling and demonstrated the significant importance of sampling in deep embedding learning. However, these methods are hand-crafted and suffer from limited searching space. Rather than searching for effective training examples at every training step, we focus on adaptively learning discriminative sampling policies.

3. Proposed Approach

Let $\mathbf{X} = \{x_1, x_2, \cdots, x_n\}$ be the input data and $y_i \in \{1, 2, \cdots, C\}$ be the label of x_i. Deep embedding learning aims to train a deep forward network $f(\cdot;\theta) : \mathbf{X} \rightarrow \mathbb{R}^d$, where x is the input and θ is the learnable parameters. For simplicity, we omit x and θ from $f(x;\theta)$ with superscripts and subscripts inherited. We define the distance between a pair of embedded features as $D_{ij} = ||f_i - f_j||_2$, where $||\cdot||_2$ represents the Euclidean norm. In general, the goal of deep embedding learning is to minimize the distances between positive pairs and maximize the distances between negative pairs. To achieve this, most deep embedding approaches train the network with a well-designed objective $L_{\text{emb}}(\cdot;\theta)$, where representative objectives include contrastive loss and triplet loss.

Contrastive loss focuses on absolute distances by taking pairwise samples $\{x_i, x_j\}$ as inputs. The objective function of contrastive loss is shown as follows:

$$L_{\text{cont}} = \mathbf{1}\{y_i = y_j\}D_{ij}^2 + \mathbf{1}\{y_i \neq y_j\}[\alpha - D_{ij}]_+^2, \quad (1)$$

where α is the margin and the operation of $[\cdot]$ represents the hinge function $\max(0, \cdot)$.

Triplet loss constructs triplet samples $\{x_a, x_p, x_n\}$ for training, which represent anchor, positive and negative samples, respectively. Compared with contrastive loss, triplet loss only requires the relative ordering of distances:

$$L_{\text{tri}} = [D_{ap}^2 - D_{an}^2 + \alpha]_+. \quad (2)$$

Optimizing with all training pairs or triplets suffer from infeasible computation as the sizes of samples are quadratic or cubic. Hence, hard example mining methods are employed to select meaningful samples within a batch. Simple hard negative mining is able to accelerate the convergence of contrastive loss, while semi-hard criterion is presented by FaceNet [25] for triplet loss:

$$x_n^* = \underset{x_n : D_{ap} < D_{an}}{\arg\min} \; D_{an}, \quad (3)$$

where x_n^* represents the selected semi-hard negative sample given anchor and positive samples.

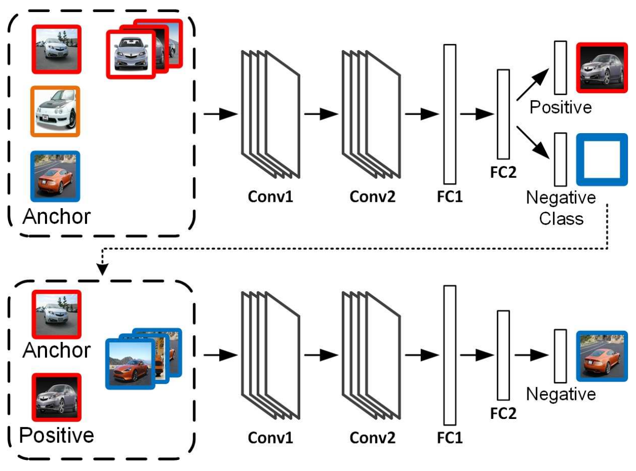
3.1. Deep Sampler Network

In this paper, instead of computing the costly $\arg\min$ and $\arg\max$ functions, we train a deep sampler network to select effective training samples with a softmax policy. The probability of choosing one sample is given by:

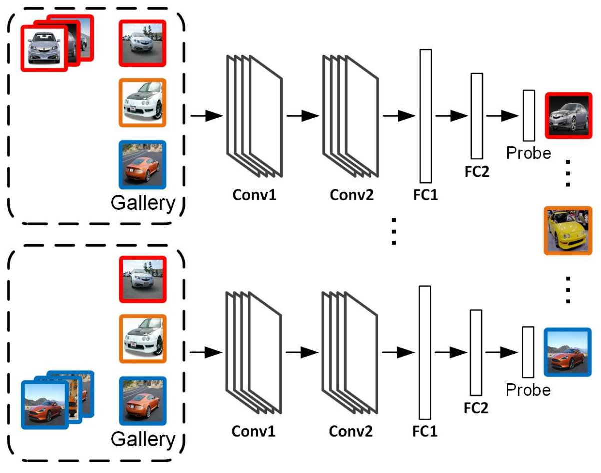
$$p(\mathbf{x}_s|\mathbf{X};\eta) = \frac{\exp\left(H(\mathbf{x}_s;\eta)\right)}{\sum_{\mathbf{x}\in\{\mathbf{x}_s\}} \exp\left(H(\mathbf{x};\eta)\right)}, \quad (4)$$

where $\mathbf{x}_s = \{x_{(1)}, x_{(2)}, \cdots, x_{(L)}\}$ represents a selected sample, e.g. $\{x_i, x_j\}$ for pairwise inputs and $\{x_a, x_p, x_n\}$ for triplet inputs, and $\{\mathbf{x}_s\}$ is the set of candidate training samples. $H(\mathbf{x}_s;\eta)$ is defined as the instructiveness of the selected training sample $\mathbf{x}_s$, which measures how effective it is to train the embedding. Previous studies show that $H(\mathbf{x}_s;\eta)$ should be highly related to the hardness of the selected sample [25, 38], and the deep sampler network $g(\cdot;\eta)$ predicts $H(\mathbf{x}_s;\eta)$ for each candidate sample with the parameters η.

While the learned sampling policy is able to predict the instructiveness of the candidate samples without exhaustive search, the number of candidate training samples $\mathbf{x}_s$ is still



(a) Sampler for Triplet Embedding



(b) Sampler for N-pair Embedding

Figure 2. The architectures of deep sampler network for (a) triplet embedding, and (b) N-pair embedding. For triplet embedding, the first sampler selects the positive example and the negative class, and the second sampler chooses the negative example. For N-pair embedding, the sampler independently selects N probe samples to construct N pairs with the gallery samples. (Best viewed in color.)

exponential. To address this, we employ L networks to sequentially select $x_{(l)}$ for the sample $\mathbf{x}_s$ with the conditional probability formula:

$$\begin{aligned} p(\mathbf{x}_s|\mathbf{X};\eta) &= p(x_{(1)}, x_{(2)}, \cdots, x_{(L)}|\mathbf{X};\eta) \\ &= \prod_{l=1}^{L} p(x_{(l)}|x_{(1)}, \cdots, x_{(l-1)}, \mathbf{X};\eta), \end{aligned} \tag{5}$$

where $\eta = \{\eta_1, \cdots, \eta_L\}$ is the parameters of the networks.

In order to train the deep sampler network, the feature embedding should judge the quality of each selected sample $\mathbf{x}_s$ with a score function $S(\mathbf{x}_s;\theta)$, where more instructive samples gain higher scores. The goal of the deep sampler network is to maximize the expected scores with the sampling policy:

$$L_s(\mathbf{X}; g, f) = -\mathbb{E}_{\mathbf{x}_s \sim p(\mathbf{x}_s|\mathbf{X};\eta)}[S(\mathbf{x}_s;\theta)], \tag{6}$$

and we omit the subscripts of $\mathbb{E}_{\mathbf{x}_s \sim p(\mathbf{x}_s|\mathbf{X};\eta)}$ for short.

When f is fixed, (6) has a global optima of

$$p(\mathbf{x}_s|\mathbf{X};\eta) = \begin{cases} 1 & \text{if } \mathbf{x}_s = \arg\max_{\mathbf{x}} S(\mathbf{x};\theta), \\ 0 & \text{else.} \end{cases} \tag{7}$$

The global optima of (6) is equal to the result of exhaustive search in hand-crafted methods under a proper definition of S, such as a score of hardness or semi-hardness. However, the proposed deep sampler network requires much less hardness tests and allows larger sampling space. In this paper, we set $S(\mathbf{x}_s;\theta) = L_{\text{emb}}(\mathbf{x}_s;\theta)$ for simplicity, where L_{emb} is the loss function employed for deep embedding learning.[1] Moreover, it is necessary to avoid repetitively selecting the same sample with a high score in a mini-batch. A simple yet effective solution is to uniformly specify $x_{(1)}$ (e.g. anchors in the triplet embedding) from all the classes by hand, and select the others to construct training examples with the learned sampling policy.

[1] In the experiments, we minus the average score of the mini-batch for each $S(\mathbf{x}_s;\theta)$ to provide both positive and negative scores.

While it is usually hard to score each $x_{(l)}$ in the training sample, we train all the L networks with the holistic score $S(\mathbf{x}_s;\theta)$ considering L is relatively small. With the equality $\frac{\partial}{\partial\eta}p(\mathbf{x}_s|\mathbf{X};\eta) = p(\mathbf{x}_s|\mathbf{X};\eta)\frac{\partial}{\partial\eta}\ln p(\mathbf{x}_s|\mathbf{X};\eta)$, we obtain the derivative of the expected reward as follows:

$$\frac{\partial}{\partial\eta}\mathbb{E}[S(\mathbf{x}_s;\theta)] = \mathbb{E}[S(\mathbf{x}_s;\theta)\frac{\partial}{\partial\eta}p(\mathbf{x}_s|\mathbf{X};\eta)]. \tag{8}$$

We first sample $\mathbf{x}_s$ according to the current policy, and compute its score $S(\mathbf{x}_s;\theta)$ through embedding. In this way, we obtain a mini-batch of $\mathbf{x}_s$ with scores. Then, we rewrite $p(\mathbf{x}_s|\mathbf{X};\eta)$ with (5) as the product of the outputs of L networks. With each selected $\mathbf{x}_s$, η_l only affects the output of the l-th network, and we can compute the derivative of each network respectively.

3.2. Joint Deep Sampler and Embedding Learning

In DE-DSP, we jointly train the deep sampler network and the feature embedding in an iterative manner, which is a general framework and a wide range of existing supervised embedding losses $L_{\text{emb}}(\cdot;\theta)$ are applicable. The deep sampler network learns discriminative sampling policy to generate targeted examples for the feature embedding, and the parameters of embedding are optimized with the effective training examples. Algorithm 1 details the training procedure of DE-DSP. In the following, we discuss the proposed DE-DSP framework with two different losses: triplet embedding and N-pair embedding.

DE-DSP for Triplet Embedding: Triplet loss [25] is widely-used in feature embedding, which is formulated as (2). In order to avoid repetitive sampling within a mini-batch, we randomly select the anchors from varying classes at each time, and learn the sampling policies of positive and negative examples given the anchor with $L = 2$. Figure 2 (a) shows the network architecture of the sampler for triplet embedding, where the two networks share the parameters of the convolutional and fully connected layers. For the first network, we pile all the anchors (with the current anchor for constructing the triplet as the first) and candidate positives up as the input, simultaneously employing two softmax layers for positive example and negative class selections. The reason for negative class determination is to reduce the dimension of the softmax layer for the second network. The second network takes the current anchor, the selected positive example and the candidate negative examples from the choosing class as the input, and obtains the selected negative example through softmax.

DE-DSP for N-pair Embedding: N-pair loss [29] is a recent embedding learning structure, where each batch contains N pairs of probe and gallery samples. Each probe sample together with all the gallery samples would construct an $(N+1)$-tuplet, which includes a positive example and N negative examples. The objective function of N-pair loss is written as follows:

$$L_{\text{npair}} = \frac{1}{N}\sum_{i=1}^{N} \log\big(1 + \sum_{j \neq i} \exp\big(f_i^T f_j^+ - f_i^T f_i^+\big)\big), \quad (9)$$

where x_i and x_i^+ are probe and gallery samples. We suggest the readers referring to [29] for complete details.

While the N-pair loss achieves the state-of-the-art performance, very few works research on mining more effective batches. In DE-DSP, we aim to learn the policy of selecting N probe samples with the pre-set gallery samples. Given N gallery samplers, the selection of each probe sample is independent due to the decomposability of (9). Therefore, we employ the same deep sampler network to select each probe sample separately. Figure 2 (b) shows the structure of deep sampler network for N-pair embedding.

4. Experiments

We conducted experiments on the widely-used CUB-200-2011 [34], Cars196 [17], Stanford Online Products [31] and In-Shop Clothes Retrieval [21] datasets following the standard evaluation protocols. The CUB-200-2011 dataset contains 200 bird species with 11,788 images. We applied the first 100 species with 5,864 images for training, and the rest 100 species with 5,924 images for testing. The Cars196 dataset includes 196 car models with 16,185 images. We applied the first 98 models with 8,054 images for training, and the remaining 98 models with 8,131 images for testing.

Algorithm 1: DE-DSP

```
Input: Training set, number of mini-batches in each batch
    $K$, and iteration number $T$.
Output: Parameters of the deep sampler network $\eta$, and
    parameters of the feature embedding $\theta$.
1:  Initialize the parameters of feature embedding $\theta$.
2:  Pre-train the parameters of deep sampler network $\eta$.
3:  for $iter = 1, 2, \cdots, T$ do
4:      for $k = 1, 2, \cdots, K$ do
5:          Sample mini-batch of $m$ training examples with
            the sampling policy $p(\mathbf{x}_s|\mathbf{X};\eta)$.
6:          Set the mini-batch in the batch.
7:          Compute the scores $S(\mathbf{x}_s;\theta)$ through forward-
            propagation of the feature embedding.
8:          Update the parameters of deep sampler network
            $\eta$ with the mini-batch using (8).
9:      end for
10:     Update the parameters of feature embedding $\theta$ with
        the batch using $L_{\text{emb}}$.
11: end for
12: return $\eta$ and $\theta$.
```

The Stanford Online Products dataset has 22,634 products from eBay.com with 120,053 images. We applied the first 11,318 products with 59,551 images for training, and the rest 11,316 products with 60,502 images for testing. The In-Shop Clothes Retrieval dataset consists of 11,735 classes of clothes with 54,642 images. We applied the first 3,997 classes with 25,882 images for training, 3,985 classes with 14,218 images for query, and the rest 3,985 classes with 12,612 images for gallery.

In the training procedure, We first normalized each image to 256×256 and then performed standard data augmentation by random crop and horizontal mirroring. We initialized all the embedding networks for baseline and proposed methods with GoogLeNet [32] pretrained on the ImageNet ILSVRC dataset [24], adding a fully connected layer with random initialization. We trained the additional fully connected layer with 10 times learning rate compared with other layers. As the performance is not largely affected by embedding sizes [31], we fixed the embedding length as 512 throughout the experiments. The batchsizes are 120 for both triplet embedding and N-pair embedding, and the size of a mini-batch for triplet loss is 30.

In the experiments, we employed both retrieval and clustering tasks for effectiveness demonstration. For the clustering task, we computed the normalized mutual information (NMI) and F_1 metrics on the test set with the K-means algorithm. NMI employs the ground truth classes $\mathbb{C} = \{c_1, \cdots, c_K\}$ and a set of clusters $\Omega = \{\omega_1, \cdots, \omega_K\}$ as the input, where c_i represents the samples with the label i, and ω_j is the samples belonging to the jth cluster. NMI is

Table 1. Experimental results (%) on the CUB-200-2011 and Cars196 datasets compared with the widely-used baseline methods.

Method	CUB-200-2011					Cars196				
	NMI	F_1	R@1	R@2	R@4	NMI	F_1	R@1	R@2	R@4
DDML	47.3	13.1	31.2	41.6	54.7	41.7	10.9	32.7	43.9	56.5
Lifted	56.4	22.6	46.9	59.8	71.2	57.8	25.1	59.9	70.4	79.6
Clustering	59.2	-	48.2	61.4	71.8	59.0	-	58.1	70.6	80.3
Angular	61.0	30.2	53.6	65.0	75.3	62.4	31.8	71.3	80.7	87.0
DAML	61.3	29.5	52.7	65.4	75.5	66.0	36.4	75.1	83.8	89.7
Triplet	49.8	15.0	35.9	47.7	59.1	52.9	17.9	45.1	57.4	69.7
Semi-hard (Triplet)	50.3	16.4	37.9	50.4	63.0	53.3	18.5	52.4	65.2	75.1
DE-DSP (Triplet)	**53.7**	**19.8**	**41.0**	**53.2**	**64.8**	**55.0**	**22.3**	**59.3**	**71.3**	**81.3**
N-pair	60.2	28.2	51.9	64.3	74.9	62.7	31.8	68.9	78.9	85.8
DE-DSP (N-pair)	**61.7**	**30.5**	**53.6**	**65.5**	**76.9**	**64.4**	**33.3**	**72.9**	**81.6**	**88.8**

Table 2. Experimental results (%) on the Stanford Online Products and In-Shop Clothes Retrieval datasets compared with the widely-used baseline methods.

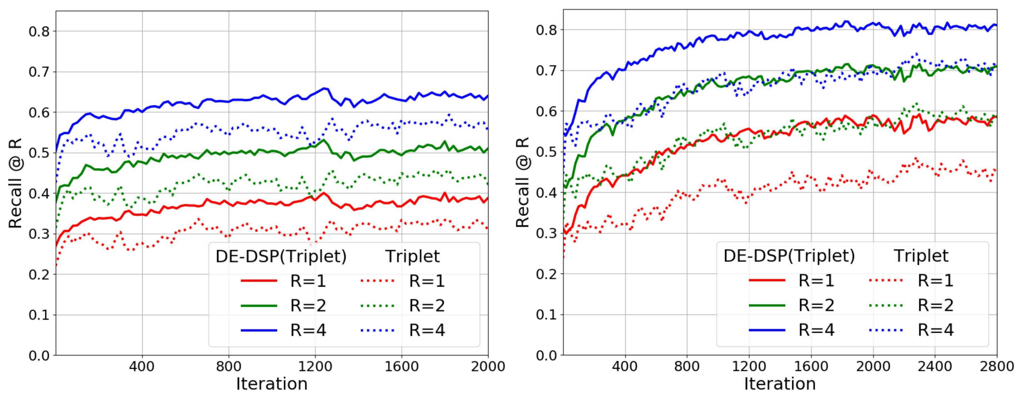
Method	Stanford Online Products					In-Shop Clothes Retrieval				
	NMI	F_1	R@1	R@10	R@100	R@1	R@10	R@20	R@30	R@40
DDML	83.4	10.7	42.1	57.8	73.7	24.4	47.8	55.6	60.4	64.2
Lifted	87.2	25.3	62.6	80.9	91.2	75.3	93.1	95.5	96.4	97.0
Clustering	89.5	-	67.0	83.7	93.2	-	-	-	-	-
Angular	87.8	26.5	67.9	83.2	92.2	80.4	93.9	95.7	96.5	97.1
DAML	89.4	32.4	68.4	83.5	92.3	78.9	93.8	95.7	96.6	97.1
Triplet	86.3	20.2	53.9	72.1	85.7	56.1	82.0	86.8	89.2	90.6
Semi-hard (Triplet)	86.9	21.7	55.9	73.5	86.7	57.0	82.8	87.2	90.1	91.6
DE-DSP (Triplet)	**87.4**	**22.7**	**58.2**	**75.8**	**88.4**	**58.7**	**84.4**	**89.3**	**91.5**	**92.8**
N-pair	87.9	27.1	66.4	82.9	92.1	76.4	93.6	94.7	95.6	96.2
DE-DSP (N-pair)	**89.2**	**30.6**	**68.9**	**84.0**	**92.6**	**78.6**	**93.8**	**95.5**	**96.2**	**96.7**

the ratio of mutual information and the mean entropy of the ground truth and clusters $\mathrm{NMI}(\Omega, \mathbb{C}) = \frac{2I(\Omega;\mathbb{C})}{H(\Omega)+H(\mathbb{C})}$, and F_1 metric is defined as the harmonic mean of precision and recall $F_1 = \frac{2PR}{P+R}$. For the retrieval task, we calculated the percentage of test samples that had at least one sample from the same class in R nearest neighbors.

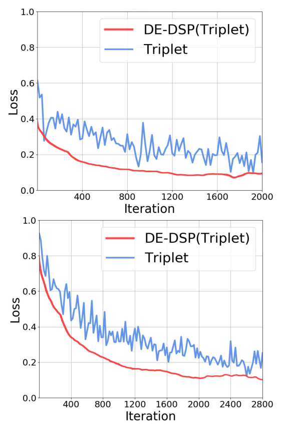
4.1. Quantitative Results

We applied the DE-DSP framework on triplet loss [25] and N-pair loss [29] for direct comparisons, and also compared with widely-used baseline methods including D-DML [13], lifted structure [31], clustering [30], angular loss [35] and DAML [6]. In the experiments, we also tested the performance of triplet loss with semi-hard negative mining strategy for reference. Among the listed methods, contrastive loss and DDML are weakly supervised as they only require pairwise inputs, while other methods are strongly supervised with full annotations of identities.

Tables 1 and 2 show the experimental results on the CUB-200-2011, Cars196, Stanford Online Products and In-Shop Clothes Retrieval datasets respectively, where bold numbers show the results which are improved with the learned sampling policy compared with random/semi-hard strategies. We observe that the proposed DE-DSP framework successfully boosts the performance of existing triplet embedding and N-pair embedding due to the selection of effective training examples. In general, the improvement in Cars196 is larger than CUB-200-2011, where a possible reason is that the images in Cars196 have smaller variations and require more carefully sampling. For triplet embedding, it is crucial to select effective training examples, and the proposed DE-DSP leads to relatively large improvement. Semi-hard sampling strategy is also tested for training data mining. However, as a hand-crafted method, semi-hard negative mining suffer from exhaustive search and limited searching space. Instead, the proposed DE-DSP exploits the relationships between samples to learn an effective sampling policy without additional prior sampling knowledge of human. For N-pair embedding, while most existing sampling strategies are not applicable as they are especially designed for contrastive loss and triplet loss, DE-DSP is a general framework which can be employed to varying em-



(a) Recall Scores



(b) Loss Plots

Figure 3. Recall scores and loss plots on the CUB-200-2011 and Cars196 datasets. (a) Left: CUB-200-2011. Right: Cars196. Solid lines represent the proposed DE-DSP based method and dash lines represent triplet loss. Lines in the same color share the same value of R. (b) Top: CUB-200-2011. Bottom: Cars196. (Best viewed in color.)

bedding losses. N-pair embedding extends the triplet loss to an $(N+1)$-tuplet loss allowing more comparisons within a batch, which partly addresses the problem of meaningful training sample selection. Therefore, the improvement is relatively small for the proposed DE-DSP compared with the triplet embedding, yet it still improves the results and achieves the state-of-the-art performance on both datasets.

Figure 3 shows the validation recall scores under different R and loss plots of triplet and DE-DSP (triplet) embeddings on the CUB-200-2011 and Cars196 datasets. The loss plots iterate over one batch and we drew each point training the deep embedding with a batch. We observe that the DE-DSP based method selects highly effective training samples, so that the recall scores increase rapidly and continuously outperform triplet loss with random sampling. The loss plots present significant differences between triplet embedding and DE-DSP (triplet) methods, which suffer from severe vibration for the triplet embedding and descend smoothly for the proposed DE-DSP method. While there are fluctuations in training losses for random sampling as it is not guaranteed to select meaningful training examples at each step, DSN has learned the effective sampling policy for the current embedding to ensure the decrease of training losses at each iteration. The smoothness of loss plots demonstrate the discriminativeness of the sampling policy.

We also compared our DE-DSP with recent sampling methods on the Cars196 dataset with the widely-used triplet loss, which included semi-hard sampling, N-pair sampling, and fast approximate nearest neighbour graph (FANNG) as shown in Table 3. Semi-hard sampling is an easy but effective approach by selecting hard negatives which are farther from the anchor than positives. N-pair sampling aims to select triplets using the structure of N-pair data. FANNG is a recent smart mining method with low computational costs.

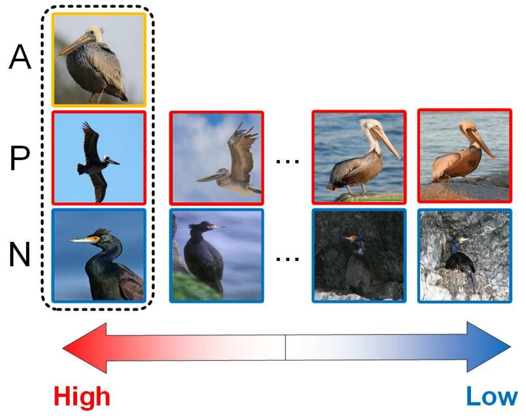
Table 3. Experimental results (%) on the Cars196 dataset compared with varying sampling strategies.

Method	R@1	R@2	R@4	R@8
Triplet	45.1	57.4	69.7	79.2
N-pair sampling	46.3	59.9	71.4	81.3
Semi-hard sampling	52.4	65.2	75.1	84.3
FANNG sampling	58.2	70.6	78.9	86.7
DE-DSP (Triplet)	**59.3**	**71.3**	**81.3**	**88.6**

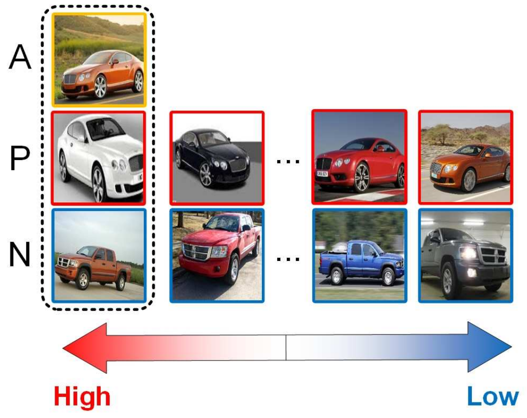
For fair comparisons, we fixed the network architecture as GoogLeNet and guaranteed the only difference as the training data. We observe that DE-DSP achieves better results than the recent sampling methods, which demonstrates the effectiveness of the learned sampling policy. Moreover, DE-DSP can be generally applied to various training losses compared with loss-specific methods.

4.2. Comparison in Sampling Time

We tested the computational time to select one negative sample from 100 classes where each contained 100 images, given an anchor and a positive sample. For the searching method, we needed to obtain 10,000 GoogLeNet features for all the candidates and computed 20,000 Euclidean distances, which took around 4.3s. For DE-DSP, we only performed 102 forward passes of DSN (much shallower than GoogLeNet) to determine the class, and another 102 forward passes to choose the sample. The time was 0.016s, where we bypassed exhaustive GoogLeNet feature extraction and distance computation due to the exploitation of relationships among samples. As the training process of DSN might be costly, we fine-tuned its parameters based on the last step for quick convergence. The average time of



(a) CUB-200-2011



(b) Cars196

Figure 4. Selected triplet samples with the learned sampling policy on (a) CUB-200-2011, and (b) Cars196. In the figure, *A* is the anchor sample, *P* represents the candidate positive samples, and *N* represents the candidate negative samples. Through the learned sampling policy, we obtain the possibility for each sample of being selected, where the images on the left are with high possibilities and right with low possibilities. The deep sampler network selects the samples according to the possibilities for effective embedding learning.

fine-tuning DSN was 5.6s per batch, and the total average processing time for a batch (both sampling and embedding learning) was 20.6s. In general, we train a lightweight DSN model to exploit the relations among training samples, so that it would not be required to perform exhaustive forward passes through the heavyweight GoogLeNet model.

4.3. Qualitative Results

While the proposed DE-DSP successfully boosts the performance and smooths the loss plots on the two datasets, it is still required to visualize the selected samples for demonstrating the effectiveness of the learned sampling policy. In this subsection, we visualize the sampling results of the final deep sampler network for an intuitive observation. Figure 4 shows the sampling examples of triplet embedding on both CUB-200-2011 and Cars196. In the figure, the first row is the selected anchor sample. The second and third rows are candidate positive and negative samples, respectively. According to the sampler for triplet embedding shown in Figure 2 (a), the candidate negative samples are from the same selected class for each anchor sample. As the learned policy is able to obtain the possibility of each sample, we show the first two positive/negative samples with highest/lowest possibilities given the anchor sample. We observe that hard samples, i.e. dissimilar positive samples and similar negative samples, have larger chances of being selected while easy samples are with low possibilities. The visualization results of sample selection demonstrate the discriminativeness of the learned sampling policy, which is able to mine more meaningful training examples for effective embedding learning. It should also be noticed that although hard samples have higher possibilities of being selected as shown in Figure 4, easy samples still have chances to be chosen with less opportunities. As pointed out in [38], training the deep embedding network with only hard examples may lead to sub-optima, and the proposed DE-DSP shares the similar thoughts by selecting samples according to the possibilities.

5. Conclusion

In this paper, we have proposed a deep embedding with discriminative sampling policy (DE-DSP) learning framework, which can be generally applied to varying embedding losses. Through joint learning of sampling policy and feature embedding, DSN gradually selects targeted training examples for effective training at each step. Compared with the existing sampling strategies, DSN exploits the relations among candidate samples to learn discriminative sampling policy rather than exhaustive search, which does not require strong prior sampling knowledge of human by simply setting the embedding objective as the reward of DSN. Moreover, it is more adaptive to different training steps through fine-tuning the parameters based on last step, which avoids repetitive computation at each step. Experimental results show that the proposed deep sampler network is able to learn discriminative sampling policy, and DE-DSP successfully boosts the performance of the widely-used triplet embedding and N-pair embedding on the benchmark datasets.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant 61672306, Grant U1813218, Grant 61822603, Grant U1713214, and Grant 61572271. The authors would like to thank Mr. Qiyuan Dong for valuable discussions.

References

[1] Slawomir Bak and Peter Carr. One-shot metric learning for person re-identification. In *CVPR*, pages 1571–1580, 2017. 1

[2] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *ECCV*, pages 730–746, 2016. 1

[3] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546, 2005. 1, 3

[4] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *CVPR*, pages 1153–1162, 2016. 1

[5] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007. 1, 3

[6] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *CVPR*, pages 2780–2789, 2018. 3, 6

[7] Zheyun Feng, Rong Jin, and Anil Jain. Large-scale image annotation by efficient and robust kernel metric learning. In *ICCV*, pages 1609–1616, 2013. 3

[8] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *ECCV*, pages 269–285, 2018. 3

[9] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *NIPS*, pages 451–458, 2006. 3

[10] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? Metric learning approaches for face identification. In *CVPR*, pages 498–505, 2009. 1, 3

[11] Ben Harwood, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *ICCV*, pages 2821–2829, 2017. 1, 3

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[13] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face and kinship verification. *TIP*, 26(9):4269–4282, 2017. 6

[14] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *NIPS*, pages 1262–1270, 2016. 1, 3

[15] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017. 1

[16] Lalit Jain, Blake Mason, and Robert Nowak. Learning low-dimensional metrics. In *NIPS*, pages 4142–4150, 2017. 1, 3

[17] Jonathan Krause, Michael Stark, Jia Deng, and Li Feifei. 3d object representations for fine-grained categorization. In *ICCVW*, 2013. 2, 5

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[19] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013. 3

[20] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *ECCV*, pages 689–704, 2018. 3

[21] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, pages 1096–1104, 2016. 2, 5

[22] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, pages 360–368, 2017. 1

[23] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *BMVC*, pages 1–12, 2015. 1

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 1, 2, 3, 5, 6

[26] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *NIPS*, pages 41–48, 2004. 3

[27] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *ICML*, 2004. 3

[28] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 3

[29] Kihyuk Sohn. Improved deep metric learning with multi-class N-pair loss objective. In *NIPS*, pages 1849–1857, 2016. 1, 2, 3, 5, 6

[30] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *CVPR*, pages 5382–5390, 2017. 1, 6

[31] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016. 1, 2, 3, 5, 6

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 5

[33] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, pages 4170–4178, 2016. 1, 3

[34] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 5

[35] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *ICCV*, pages 2593–2601, 2017. 1, 6

[36] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb):207–244, 2009. 1, 3

[37] Kilian Q Weinberger and Gerald Tesauro. Metric learning for kernel regression. In *AISTATS*, pages 612–619, 2007. 3

[38] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *ICCV*, pages 2840–2848, 2017. 1, 2, 3, 8

[39] Rui Yu, Zhiyong Dou, Song Bai, Zhaoxiang Zhang, Y-ongchao Xu, and Xiang Bai. Hard-aware point-to-set deep metric for person re-identification. In *ECCV*, pages 196–212, 2018. 1

[40] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *ICCV*, pages 814–823, 2017. 1, 3

[41] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, pages 6034–6042, 2016. 1

[42] Jiahuan Zhou, Pei Yu, Wei Tang, and Ying Wu. Efficient on-line local metric adaptation via negative samples for person re-identification. In *ICCV*, pages 2420–2428, 2017. 1, 3