

DAS: Densely-Anchored Sampling for Deep Metric Learning

Lizhao Liu^{1,2}, Shangxin Hunag¹, Zhuangwei Zhuang¹,
Ran Yang¹, Mingkui Tan^{1,3†}, and Yaowei Wang^{2†}

¹South China University of Technology ²PengCheng Laboratory
³Information Technology R&D Innovation Center of Peking University
{selizhaoliu,sevtars,z.zhuangwei,msyangran}@mail.scut.edu.cn,
mingkuitan@scut.edu.cn, wangyw@pcl.ac.cn

Abstract. Deep Metric Learning (DML) serves to learn an embedding function to project semantically similar data into nearby embedding space and plays a vital role in many applications, such as image retrieval and face recognition. However, the performance of DML methods often highly depends on sampling methods to choose effective data from the embedding space in the training. In practice, the embeddings in the embedding space are obtained by some deep models, where the embedding space is often with barren area due to the absence of training points, resulting in so called “missing embedding” issue. This issue may impair the sample quality, which leads to degenerated DML performance. In this work, we investigate how to alleviate the “missing embedding” issue to improve the sampling quality and achieve effective DML. To this end, we propose a Densely-Anchored Sampling (DAS) scheme that considers the embedding with corresponding data point as “anchor” and exploits the anchor’s nearby embedding space to densely produce embeddings without data points. Specifically, we propose to exploit the embedding space around single anchor with Discriminative Feature Scaling (DFS) and multiple anchors with Memorized Transformation Shifting (MTS). In this way, by combing the embeddings with and without data points, we are able to provide more embeddings to facilitate the sampling process thus boosting the performance of DML. Our method is effortlessly integrated into existing DML frameworks and improves them without bells and whistles. Extensive experiments on three benchmark datasets demonstrate the superiority of our method.

Keywords: Deep Metric Learning · Missing Embedding · Embedding Space Exploitation · Densely-Anchored Sampling

1 Introduction

Deep Metric learning (DML) is the foundation of various applications, including face recognition, verification [10, 42], image retrieval [21], image clustering [17],

[†] Corresponding authors.

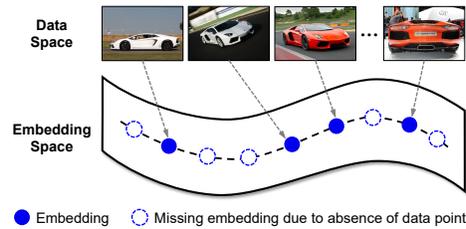


Fig. 1. Illustration of the “missing embedding” issue. The data points of similar semantics are mapped into the nearby embedding space that is often with barren area due to the absence of data points, resulting in the “missing embedding” issue

image classification [12], few-shot learning [32], video representation learning [5] and sound generation [6] *etc.* Since it was introduced, it has sparked considerable interest in the community, where academics have offered a variety of methods [18, 36, 38, 42, 45, 53, 56] and have made substantial progress [37, 41]. The goal of DML is to learn a deep model that is capable of mapping semantically similar data points to similar embeddings in the embedding space. To accomplish this, most existing approaches [7, 18, 42, 48, 53, 56] train the deep model with loss functions that bring the embeddings from semantically similar data points close to each other and vice versa. However, some embeddings may have limited contribution or bring no improvement to train the deep model [56], or even lead to bad local minima early on in training (such as a collapsed model) [42]. Thus, sampling informative and stable embeddings is very important to facilitate the training of deep model [56]. As a result, improving the sample quality is of significance to achieve effective DML. There are two commonly used measures for this goal: designing more effective sampling methods or providing more embeddings.

Pioneering efforts have made substantial progress toward the design of effective sampling methods upon embedding pairs [41, 42, 56] or a full batch of embeddings [36, 38]. These methods typically perform sampling on a batch of embeddings, which often leads to inaccurate sampling results due to the following reasons. First, the batch size is typically constrained by the memory of a single GPU as the sampling process typically cannot cross different GPU devices [41]. Second, even with GPU that has sufficient memory to support a larger batch size, the embedding space that contains the embeddings embedded by deep models may still with barren area due to the absence of data points, resulting in a “missing embedding” issue (as shown in Fig. 1). Thus, the limited amount of embeddings may impair the sample quality and the performance of DML. Based on the above analyses, we ask: “Can we overcome the inaccurate sampling issue brought by the absence of data points?”

Very recently, a few attempts [13, 16, 28, 54, 64] have been committed to answering this question by pseudo embedding generation. Hard example generation approaches [13, 64] generate hard embeddings from easy embeddings with an additional generative adversarial network or auto-encoder. Embedding expansion [28] performs interpolation between embeddings to achieve augmentation in embedding space. Cross batch memory [54] maintains embeddings

from previous iterations and considers them are still informative in the current batch in terms of sampling. However, these approaches either leverage additional sub-network [13, 64], which introduce extra training cost, or need further modification to the sampling and loss computation process [28, 54] in standard DML [18, 20, 42, 45, 53], which may limit their applicability to other tasks.

In this paper, we seek to densely produce embeddings without data points to alleviate the “missing embedding” issue. In this way, with the combination of embeddings with and without data points, we are able to provide more embeddings for sampling to improve the sample quality and achieve effective DML. Our motivation stems from a fundamental hypothesis of metric learning: *the embeddings that are close to each other in the embedding space have similar semantics*. Unfortunately, how to exploit the embedding space to produce effective embeddings without data points remains an unsolved problem. To this end, we propose a Densely-Anchored Sampling (DAS) scheme to consider the embedding with data point as “anchor” and densely exploit the anchor’s nearby embedding space to produce embeddings that have no corresponding data points. The proposed DAS is consist of two modules, namely, Discriminative Feature Scaling (DFS) and Memorized Transformation Shifting (MTS), which exploit the embedding space around single and multiple embeddings respectively to produce effective embeddings with no corresponding data points. To be specific, based on observations that effective semantics for one embedding are highly activated features [2, 3, 14], DFS identifies these features and applies random scaling on them. In this sense, we are able to exploit the embedding space around a single embedding by enhancing or weakening its effective semantics to produce embeddings. Based on the fact that semantic differences (*i.e.*, transformations) of intra-class embeddings can be added to other embeddings to generate effective embeddings [31], we assume that they can be added in a way like word embeddings [34]: $Queen = Woman + (King - Man)$. Thus, our Memorized Transformation Shifting module exploits the embedding space among multiple embeddings by adding (*i.e.*, shift) intra-class embeddings’ semantic differences to other embeddings of the same class to produce effective embeddings.

Our main contributions are summarized as follows. **First**, we propose a novel and plug-and-play Densely-Anchored Sampling (DAS) scheme that exploits embeddings’ nearby embedding space and densely produces embeddings without data points to improve the sampling quality and performance of DML. **Second**, we propose two modules, namely Discriminative Feature Scaling (DFS) and Memorized Transformation Shifting (MTS) to exploit embedding space around a single and multiple embeddings to produce embeddings. **Last**, extensive experiments demonstrate the effectiveness of the proposed method.

2 Related Work

Sampling Methods in DML. Sampling informative and stable embeddings is vital to train the deep model in DML [41, 42, 56]. Thus, various sampling approaches [15, 41, 42, 53, 56] have been tailored to effectively sample the em-

beddings to train the deep model. To further improve sampling efficiency, some researchers propose to leverage a whole batch of embeddings [20, 25, 36, 53, 67]. Even though sophisticated sampling methods improve DML, due to the absence of data points, sampling embeddings that are often with “missing embedding” leads to inaccurate sampling, thereby degenerating the final performance. In this paper, we propose a DAS scheme to produce embeddings with no data points by exploiting embeddings’ nearby embedding space to achieve effective DML.

Loss Functions for DML. Studies on DML losses can be grouped into two categories: pair-based and proxy-based. The pair-based losses [18, 22, 26, 42, 45, 51–53, 56, 58] are constructed upon the pairwise distance between embeddings. Although pair-based approaches mine the rich information among vast embedding pairs, they typically encounter the sampling effective embedding pairs issues. Instead, proxy-based [1, 10, 25, 33, 36, 40, 60] losses introduce the concept of “proxy” as a class representation and avoid the sampling issue by optimizing the embedding close to its proxy. However, proxy-based methods are very difficult to train when the number of classes is extremely large [38], limiting their applicability to real-life scenarios. Thus, in this paper, we focus on developing an effective technique to alleviate the general data sampling issue for pair-based methods that have wider applicability and delivers boosted performance for them.

Pseudo Embedding Generation. Methods on synthesizing pseudo embeddings have been recently shown as an important technique to improve DML [13, 28, 31, 54, 62, 64]. DAML [13] uses an additional generative adversarial network to generate only hard negatives to improve the model training. HDML [62] leverages the inter-class information for embedding generation on the sampled embeddings. DVML [31] apply extra generator and decoder to model the class centers that may have inaccurate distance estimation to the real embeddings and employs them to generate embeddings. Embedding expansion [28] linearly interpolates the embeddings to obtain more embeddings. Cross batch memory [54] stores embeddings from previous batches and considers them beneficial for the upcoming sampling process. However these approaches [28, 54] suffer from the following limitations: **First**, the generated embeddings can only be considered as negative during sampling, which may limit the power of them. **Second**, the sampling or loss computation process need to be modified to dock with them, limiting their applicability. **Third**, an additional sub-network is introduced in order to generate embeddings, which brings heavy computation cost. **Last**, information source that may have inaccurate distance measurement to real embeddings such as class centers and inter-class differences are considered to produce embeddings. Different from them, DAS is a light-weight module, which produces embeddings by densely sampling around the “anchor” and serves as a plug-and-play component to facilitate the sampling process in standard DML.

Data Augmentation. Augmentation in data space such as image [44] has been widely studied and is considered an important technique to avoid overfitting. Recently, many efforts have been made to design effective augmentation methods [8, 11, 49, 55, 57] in feature space, aiming to provide more features for training when the source data (*e.g.*, images) is scarce. These methods often in-

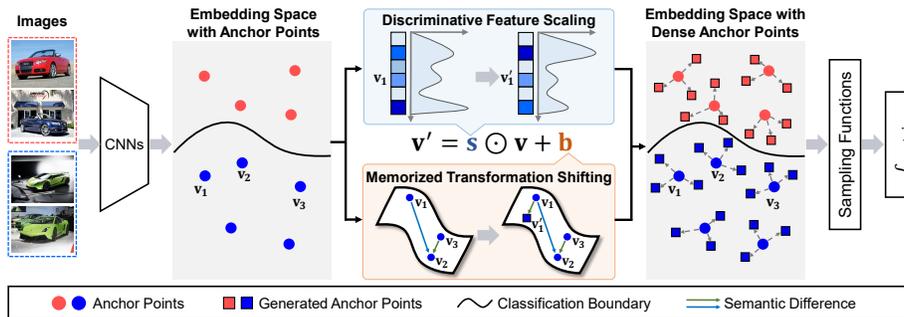


Fig. 2. Illustration of our Densely-Anchored Sampling (DAS) scheme, which leverages two modules to exploit anchors’ (*i.e.*, embeddings with data points) nearby embedding space to densely produce embeddings without data points: DFS performs random scaling on the discriminative features to produce embeddings around a single embedding; MTS exploits the embedding space among multiple embeddings by adding the intra-class semantic differences to embeddings of the same class to produce embeddings. With DAS, we alleviate the “missing embedding” issue by providing more embeddings for sampling, thereby, achieving effective DML

introduce complicated training processes [8, 49]. Wang *et al.* [55] propose ISDA that estimates semantic differences with covariance matrices and develops an improved version of cross-entropy loss. The computation and memory consumption of covariance matrices are much heavier than DAS. Moreover, ISDA can not be trivially extended to pair-based DML loss. Yin *et al.* [57] introduce FTL that requires extra networks (*e.g.*, decoder and feature transfer module) and a carefully designed bi-stage training strategy to achieve feature translation, which is less efficient and general than DAS. Unlike these methods, we view the feature space augmentation as an approach to fill the “missing embedding” in the embedding space and propose a simpler solution under the context of DML.

3 Densely-Anchored Sampling

In this paper, we seek to improve DML by alleviating the “missing embedding” issue incurred in sampling. The overall process of DAS scheme is shown in Fig. 2. **Notation.** Let $\mathbf{v} = f_{\theta}(\mathbf{I}) \in \mathbb{R}^d$ be an embedding with data point \mathbf{I} , where $f_{\theta}(\cdot)$ is a deep model (*e.g.*, CNNs) with learnable parameters θ . Let $y_{\mathbf{v}}$ be the label of the embedding \mathbf{v} . Let \mathbf{v}' denote embedding without data point. Following previous methods [42], we normalize both \mathbf{v} and \mathbf{v}' to the d -dimensional hypersphere (*i.e.*, $\|\mathbf{v}\|_2, \|\mathbf{v}'\|_2 = 1$). We omit the normalization process for brevity. Let C denote the number of training classes.

3.1 Problem Definition and Motivation

DML seeks to learn a deep model that keeps similar data points close, and vice versa. Formally, we define the distance between two embeddings as follows [56]:

$$\mathbf{D}_{ij} = \|f_{\theta}(\mathbf{I}_i) - f_{\theta}(\mathbf{I}_j)\|, \quad (1)$$

where $\|\cdot\|$ denotes the ℓ_2 norm. For any positive pair of embeddings ($y_i = y_j$), the distance should be small; Whilst for negative pair ($y_i \neq y_j$), it should be large. In practice, limited by computing resources, it is infeasible to optimize every element in \mathbf{D}_{ij} . Therefore, it is necessary to sample effective embedding pair for the objective construction (take the contrastive loss as an example):

$$\mathcal{L} = \sum_{(i,j) \in \mathcal{Q}} \mathbb{I}\{y_i = y_j\} \mathbf{D}_{ij} + \mathbb{I}\{y_i \neq y_j\} [\alpha - \mathbf{D}_{ij}]_+, \quad (2)$$

where $\mathbb{I}\{\cdot\}$ is the indication function, α is a margin, $\mathcal{Q} = S(D)$ indicates indexes of the sampled embedding pairs and S denotes some sampling function.

Without causing ambiguity, we denote the points on the embedding space as “**anchor points**”, based on which we will conduct sampling to train the deep model. Note that each data point shall have an anchor point on the embedding space. However, due to the absence of training data, the embedding space may have a lot of “barren area”. Thus, it is very difficult to provide sufficient anchor points for sampling and learn a deep model with good performance. To this end, we propose to densely produce anchor points with no data points to facilitate the sampling, thereby improving the training in DML.

Specifically, we first propose to exploit the embedding space around a single embedding by enhancing or weakening its semantics. We term this process as semantic scaling. Second, we propose to add (*i.e.*, shift) intra-class differences (*i.e.*, transformations) to embeddings to exploit the embedding space among multiple embeddings. We term this process as semantic shifting. Based on the above analyses, the formulation of DAS scheme is formulated as

$$\mathbf{v}' = \text{DAS}(\mathbf{v}; \mathbf{s}, \mathbf{b}) = \underbrace{\mathbf{s} \odot \mathbf{v}}_{\text{scaling}} + \underbrace{\mathbf{b}}_{\text{shifting}}, \quad (3)$$

where \mathbf{v} and \mathbf{v}' are embeddings with and without data points, respectively. \odot denotes the Hadamard product. $\mathbf{s}, \mathbf{b} \in \mathbb{R}^d$ are semantic scaling and semantic shifting factors, respectively. Moreover, given a set of semantic scaling and shifting factor pairs $\{(\mathbf{s}_t, \mathbf{b}_t)\}$, we are able to produce a set of embeddings by

$$\mathbf{v}'_t = \text{DAS}(\mathbf{v}; \mathbf{s}_t, \mathbf{b}_t). \quad (4)$$

Therefore, the semantic scaling and shifting factors is essential to the quality of the produced embeddings and we propose DFS and MTS to acquire them:

$$\mathbf{s} = \text{DFS}(\{\mathbf{v} \mid y_{\mathbf{v}} = c\}), \quad (5)$$

$$\mathbf{b} = \text{MTS}(\{\mathbf{v} \mid y_{\mathbf{v}} = c\}), \quad (6)$$

where DFS and MTS take embeddings from the same class as input and produce the semantic scaling and shifting factors, respectively.

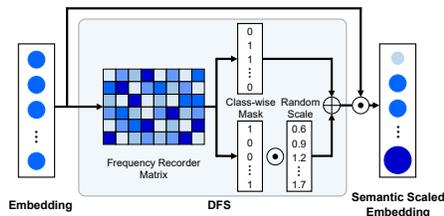


Fig. 3. Illustration of the proposed Discriminative Feature Scaling (DFS) module, which identifies the discriminative features (*e.g.*, channels) and applies different random scaling to them to produce embeddings around a single embedding

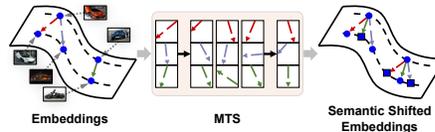


Fig. 4. Illustration of the proposed Memorized Transformation Shifting (MTS) module. MTS adds the intra-class transformations to embeddings of the same class to produce embeddings among multiple embeddings

3.2 Discriminative Feature Scaling

We seek to obtain effective semantic scaling factors to produce anchor points around a single anchor. Thus, as shown in Fig. 3, our Discriminative Feature Scaling (DFS) module carries out the semantic scaling mechanism by finding the discriminative features in an embedding and applying random scaling on them. The feasibility of DFS comes from two aspects. First, visual attributes can be predicted reliably using a sparse number of neurons from CNNs [14]. Second, in CNNs, neurons that match a diverse set of object concepts are highly activated [2, 3]. In practice, the embedding typically occupies a high dimensional embedding space, which contains semantics for all training classes and semantics for each class are diverse. Thus, the semantics for one class are more likely to be noise for another. In this sense, it is very important to find out the effective *e.g.*, discriminative features for each class in order to perform semantic scaling. To this end, we propose to identify the effective semantics by counting the number of occurrences of the highly-activated neurons for each class.

To be specific, we first initialize a Frequency Recorder Matrix (FRM) $\mathbf{P} \in \mathbb{R}^{C \times d}$ as a zero matrix. Then, given a set of embeddings $\{\mathbf{v} \mid y_{\mathbf{v}} = c\}$ from class c , we update \mathbf{P} as follows:

$$\mathbf{P}[c, k] = \begin{cases} \mathbf{P}[c, k] + 1, & \text{if } \mathbf{v}[k] \in \text{Top}(\mathbf{v}, K), \\ \mathbf{P}[c, k], & \text{otherwise,} \end{cases} \quad (7)$$

where $\text{Top}(\mathbf{v}, K)$ is an operator to select the top- K elements from the vector \mathbf{v} . During training, we constantly update the FRM by recording the position of highly-activated neurons from the embeddings of the same class. In this way, FRM¹ serves as a stable, accurate and effective semantics identifier for training classes. Given the FRM, we compute the class-wise binary channel mask $\mathbf{M} \in \mathbb{R}^{C \times d}$ by

$$\mathbf{M}[c, k] = \begin{cases} 1, & \text{if } \mathbf{P}[c, k] \in \text{Top}(\mathbf{P}[c], K), \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

¹ Visualization of the frequency recorder matrix is in the supplementary.

Given an embedding \mathbf{v} , we attain the semantic scaling factor by

$$\mathbf{s} = \boldsymbol{\gamma} \odot \mathbf{M}[y_{\mathbf{v}}] + \mathbf{1}^d \odot (1 - \mathbf{M}[y_{\mathbf{v}}]), \quad (9)$$

where $\boldsymbol{\gamma} \in \mathbb{R}^d$ and $\boldsymbol{\gamma} \sim \text{Uniform}[1 - r_s, 1 + r_s]^d$ and $r_s \in (0, 1)$ is a hyper-parameter to be set. Note that we only randomly scale the discriminative features while leaving the indiscriminative ones intact. To produce more than one scaling factor, we repeatedly sample $\boldsymbol{\gamma}$ from the uniform distribution.

3.3 Memorized Transformation Shifting

To produce anchors without data points among multiple anchors, we propose a module to provide effective semantic shifting factors. As shown in Fig. 4, our Memorized Transformation Shifting (MTS) module exploit intra-class embeddings’ nearby embedding space by leveraging the differences between embeddings and adding them to other embeddings of the same class. On the basis of that semantic differences of embeddings can be added to other embedding to generate effective embeddings [31], the motivation of our MTS comes from the semantic relations of word embedding [34]: *Woman + (King - Man) = Queen*, where a “woman” pluses “royal” semantics (*e.g.*, transformation) becomes a “Queen”.

The transformations from both inter-class and intra-class embeddings are candidates for our design choices. However, the majority of the inter-class transformations typically are not transferable due to large inter-class differences. Thus, we only consider intra-class embeddings to attain the transformations. As suggested by the latest research [41], sampling only two images for each class in a batch consistently achieves good performance, which leads to very limited transformations we can obtain in one batch *i.e.*, two transformations. To address this issue, we construct a bank to memorize the transformations from previous iterations to ensure the diversity of the transformations. Specifically, we construct a transformation bank $\mathbf{B} \in \mathbb{R}^{C \times Z \times d}$, where Z is the bank capacity for each class. Then, during training, once we obtain a set of embeddings $\mathcal{V}_c = \{\mathbf{v} \mid y_{\mathbf{v}} = c\}$ from the class c , we calculate the transformations between them by

$$\mathbf{t}_z = \mathbf{v}_i - \mathbf{v}_j, \quad \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}_c, i \neq j. \quad (10)$$

Then, the transformations are en-queued into \mathbf{B} according to the FIFO principle to ensure that the transformations in the bank are in a relatively fresh state:

$$\mathbf{B}[y_{\mathbf{v}}, z] = \mathbf{t}_z, \quad z \in \{1, 2, \dots, Z\}. \quad (11)$$

Note that z is reset to 1 when it reaches Z . Finally, with the assistance of the bank \mathbf{B} , given an embedding \mathbf{v} , we retrieve the semantic shifting factor as follow

$$\mathbf{b} = r_b \mathbf{t}, \quad \mathbf{t} \sim \{\mathbf{B}[y_{\mathbf{v}}, z] \mid z=1, 2, \dots, Z\}. \quad (12)$$

r_b is a hyper-parameter. Multiple shifting factors are formed by repeat sampling.

Algorithm 1 Training method of DAS-based DML

Require: Training image-label pairs $\mathcal{S} = \{(\mathbf{I}_i, y_i)\}_{i=1}^N$; the embedding function f_θ ; number of embeddings to produce T ; number of training classes C ; transformation bank capacity Z ; learning rate α .

Ensure: Optimized embedding function f_θ^* .

- 1: Initialize θ from ImageNet pretrained model.
- 2: Initialize the frequency recorder matrix $\mathbf{P} \in \mathbb{R}^{C \times d} = \mathbf{0}$.
- 3: Initialize the transformation bank $\mathbf{B} \in \mathbb{R}^{C \times Z \times d} = \mathbf{0}$.
- 4: **while** not converge **do**
- 5: Obtain a batch image-label pairs $\{(\mathbf{I}_i, y_i)\}_{i=1}^B$ from \mathcal{S} .
- 6: Compute embeddings $\mathbf{v}_i \leftarrow f_\theta(\mathbf{I}_i), i = 1, 2, \dots, B$.
- 7: // *perform semantic scaling by Discriminative Feature Scaling*
- 8: Update the frequency recorder matrix \mathbf{P} by Eqn. (7).
- 9: Acquire semantic scaling factors $\{\mathbf{s}_j\}_{j=1}^{B \times T}$ by Eqn. (9).
- 10: // *perform semantic shifting by Memorized Transformation Shifting*
- 11: Obtain intra-class transformations $\{\mathbf{t}\}$ by Eqn. (10).
- 12: Update the transformation bank \mathbf{B} by Eqn. (11).
- 13: Attain semantic shifting factors $\{\mathbf{b}_j\}_{j=1}^{B \times T}$ by Eqn. (12).
- 14: // *perform densely-anchored sampling*
- 15: Produce embeddings $\{\mathbf{v}'_j\}_{j=1}^{B \times T}$ by Eqn. (4).
- 16: Sample positive and negative embedding sets.
- 17: Compute the training loss $\mathcal{L}_{\text{DAS-DML}}$ by Eqn. (13).
- 18: Update the parameters θ by $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{DAS-DML}}$.
- 19: **end while**

3.4 DML with Densely-Anchored Sampling

The overall algorithm of integrating DAS into DML is detailed in Algorithm 1. Given anchor-label pairs $\{(\mathbf{v}, y_{\mathbf{v}})\}$, we produce embedding-label pairs $\{(\mathbf{v}', y'_{\mathbf{v}})\}$ with no data points by DAS scheme, where $y'_{\mathbf{v}} = y_{\mathbf{v}}$ since the class semantic is preserved. Then, embedding-label pairs with or without data points are fed into the sampling module to obtain the positive and negative embedding sets (*e.g.*, pairs, triplets, *etc.* specified by the sampling and loss functions): $\{(\mathcal{P}, \mathcal{N})\} = \text{Sample}(\{(\mathbf{v}, y_{\mathbf{v}})\} \cup \{(\mathbf{v}', y'_{\mathbf{v}})\})$. Last, given a DML loss function \mathcal{L}_{DML} ², DAS-based DML objective function is formulated as:

$$\mathcal{L}_{\text{DAS-DML}} = \mathcal{L}_{\text{DML}}(\{(\mathcal{P}, \mathcal{N})\}). \tag{13}$$

4 Experiments

Datasets. We use three popular benchmarks: 1) CUB2011-200 (CUB) [50], a fine-grained bird dataset with the first 100 categories for training and another 100 categories for testing. 2) CARS196 (CARS) [29], a fine-grained vehicle dataset with the first 98 classes for training and another 98 classes for testing. 3) Stanford Online Products (SOP) [38], a large-scale online products dataset with the train and test partitions as 11,318 classes and another 11,316 classes, respectively.

² See supplementary for detailed DML sampling methods and loss functions.

Table 1. Comparisons with SoTA methods on CUB, CARS and SOP. The best results are in **bold**. * indicates the reimplementaion by [4]. † denotes our reimplementaion

Method	Backbone	CUB				CARS				SOP			
		R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@100	R@1000
Margin [56]	R ¹²⁸	63.60	74.40	83.10	90.00	79.60	86.50	91.90	95.10	72.70	86.20	93.80	98.00
HDC [59]	G ³⁸⁴	53.60	65.70	77.00	85.60	73.70	83.20	89.50	93.80	69.50	84.40	92.80	97.70
A-BIER [39]	G ³⁸⁴	57.50	68.70	78.30	86.20	82.00	89.00	93.20	96.10	74.20	86.90	94.00	97.80
ABE [27]	G ⁵¹²	60.60	71.50	79.80	87.40	85.20	90.50	94.00	96.10	76.30	88.40	94.80	98.20
HTL [15]	IBN ⁵¹²	57.10	68.80	78.70	86.50	81.40	88.00	92.70	95.70	74.80	88.30	94.80	98.40
RLL-H [52]	IBN ⁵¹²	57.40	69.70	79.20	86.90	74.00	83.60	90.10	94.10	76.10	89.10	95.40	N/A
SoftTriple [40]	IBN ⁵¹²	65.40	76.40	84.50	90.40	84.50	90.70	94.50	96.90	78.30	90.30	95.90	N/A
MS [53]	IBN ⁵¹²	65.70	77.00	86.30	91.20	84.10	90.40	94.00	96.50	78.20	90.50	96.00	98.70
ProxyGML [67]	IBN ⁵¹²	66.60	77.60	86.40	N/A	85.50	91.80	95.30	N/A	78.00	90.60	96.20	N/A
ProxyAnchor [25]	IBN ⁵¹²	68.40	79.20	86.80	91.60	86.10	91.70	95.00	97.30	79.10	90.80	96.20	98.70
Contrastive + XBM [54]	IBN ⁵¹²	65.80	75.90	84.00	89.90	82.00	88.70	93.10	96.10	79.50	90.80	96.10	98.70
MS* [53]	IBN ⁵¹²	64.50	76.20	84.60	90.50	82.10	88.80	93.20	96.10	76.30	89.70	96.00	98.80
MS + EE* [28]	IBN ⁵¹²	65.10	76.80	86.10	91.00	82.70	89.20	93.80	96.40	77.00	89.50	96.00	98.80
ProxyAnchor + MemVir [4]	IBN ⁵¹²	69.00	79.20	86.80	91.60	86.70	92.00	95.20	97.40	79.70	91.00	96.30	98.60
MS† [53]	IBN ⁵¹²	65.72	77.19	85.74	91.56	83.86	90.41	94.64	96.99	76.89	89.58	95.59	98.60
MS + DAS ($K = 8$) (Ours)	IBN ⁵¹²	67.07	78.11	86.43	91.88	85.66	91.60	95.27	97.37	78.16	90.26	95.99	98.76
MS† [53]	R ⁵¹²	66.46	77.28	85.85	91.69	83.99	90.39	94.51	96.80	79.53	91.06	96.30	98.83
MS + DAS ($K = 8$) (Ours)	R ⁵¹²	69.19	79.25	87.09	92.62	87.84	93.15	95.99	97.85	80.59	91.80	96.68	98.95

Implementation details³ We leverage two popular backbones: ResNet50 [19] (R ^{d}) and Inception BN [23] (IBN ^{d}), where their parameters are initialized from ImageNet [9] pre-trained models and d denotes the embedding dimension. Note that some approaches also consider GoogleNet [46] (G ^{d}) as the backbone. Here, we mainly consider the settings of $d = 128, 512$. R¹²⁸ is used as the default backbone. The embedding layer is randomly initialized. Regarding evaluation metrics, Recall at k (R@ k) [24], Normalized Mutual Information (NMI) [43] and F1 score (F1) [45] are used, where R@ k measures the image retrieval performance while F1 and NMI measure the image clustering performance. For hyper-parameters in DAS, we set $(T, K, Z, r_s, r_b) = (3, 4, 10, 1e^{-2}, 1e^{-2})$ by default.⁴ Our source code is publicly available at <https://github.com/lizhaoliu-Lec/DAS>.

4.1 Comparison with State-of-the-arts

In this section, we compare our method with state-of-the-art competitors to investigate the effectiveness of DAS. The results are shown in Table 4. For fair comparisons, the results of the closely related baseline EE are from the reimplementaion by [4] using the stronger IBN⁵¹² backbone (G⁵¹² in the original paper). Our approach is based on MS loss and achieves superior performance on all datasets and evaluation metrics. **First**, when combining the R⁵¹² backbone and DAS, we are able to boost the R@1 metrics by 3.49% on CUB and 1.74% on CARS, which shows that DAS is able to deliver more accurate image retrieval results even with higher embedding dimension (*i.e.*, 512). **Second**, for our closely relative opponent, EE, its improvements on MS are marginal, showing that simply performing interpolation to generate embeddings is inferior to

³ See supplementary for more details.

⁴ Experiments on hyper-parameters T, r_s, r_b are in the supplementary.

Table 2. Comparisons with pair-based methods on CUB, CARS and SOP. [S] and [D] denote semi-hard and distance weighted sampling, respectively

Method	CUB			CARS			SOP		
	R@1	F1	NMI	R@1	F1	NMI	R@1	F1	NMI
Triplet [S] [42]	60.25	32.82	64.64	74.64	31.98	63.22	73.51	33.47	89.33
Triplet [S] + DAS	60.82	33.86	65.67	77.21	33.88	64.84	73.99	33.91	89.42
Triplet [D] [56]	62.68	36.39	67.03	78.86	35.80	65.85	77.54	37.10	90.05
Triplet [D] + DAS	64.28	38.16	68.06	82.63	39.14	68.12	77.95	37.64	90.18
Contrastive [D] [56]	61.65	35.23	66.58	76.03	32.77	64.09	73.13	35.60	89.78
Contrastive [D] + DAS	63.67	36.25	67.15	80.74	36.07	65.93	74.80	36.21	89.89
Margin [56]	62.61	37.33	67.58	80.10	37.85	67.15	78.69	39.20	90.50
Margin + DAS	64.50	37.86	68.04	82.29	38.22	67.94	79.14	39.52	90.56
GenLifted [20]	58.81	34.64	65.50	72.45	32.43	64.00	76.18	37.26	90.13
GenLifted + DAS	59.94	35.09	66.07	73.55	32.85	64.11	76.92	37.64	90.21
N-Pair [45]	60.55	36.94	67.19	77.35	36.26	66.74	77.71	37.13	90.15
N-Pair + DAS	62.81	38.37	68.43	79.93	38.06	68.20	77.98	37.82	90.28
MS [53]	62.63	38.88	68.19	82.04	40.85	69.45	78.89	37.53	90.12
MS + DAS	64.13	39.18	69.08	83.31	42.78	70.77	79.44	38.77	90.40

DAS. **Last**, even for a strong baseline, ProxyAnchor that leverages the advanced training techniques, and sophisticated loss, we still outperform it considerably.

4.2 Effectiveness of DAS on Pair-based Loss

Quantitative results. To investigate the efficacy of DAS, we conduct experiments with R¹²⁸ backbone on the widely used pair-based losses. We reimplement all baselines under the same settings for a fair comparison. The results are presented in Table 2. For considered pair-based losses, DAS is able to improve their performance on both image retrieval and clustering metrics. Notably, for approaches such as GenLifted and N-Pair that leverage the whole batch of embeddings for loss computation, DAS still improves their performance, showing its effectiveness. Last, even for a very strong baseline, MS, that considers different kinds of relationships among embedding pairs and designs sophisticated weighting mechanism, DAS is able to greatly improve it without bells and whistles.

Convergence analyses. We provide the results of training loss and test set R@1 in Fig. 5 to analyze the training behaviors of DAS.⁵ The loss curve with DAS decreases smoother than that without DAS, showing that DAS provides more embeddings to facilitate sampling, thereby, stabilizing the training. Moreover, with DAS, the training loss is higher than the baseline, one possible reason is that DAS is able to act as a regularizer to avoid overfitting, which is consistent with the result that DAS achieves a higher test set R@1. Similar phenomenon is also observed in other embeddings generation methods [28].

4.3 Effectiveness of DAS on Sampling Method

In this section, we investigate the effectiveness of DAS by evaluating it with different sampling approaches. We choose two popular loss functions: triplet and

⁵ Results on more pair-based losses are in the supplementary.

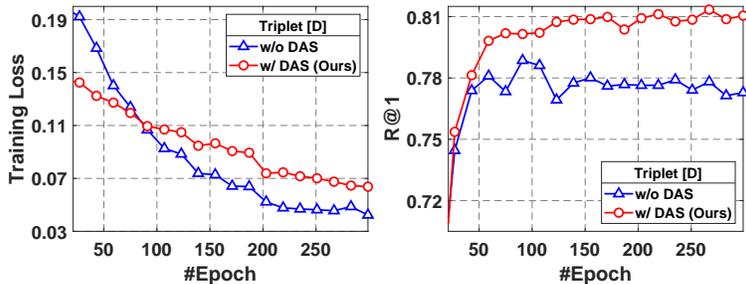


Fig. 5. The training loss and test set R@1 on CARS. The sampling method and loss function are triplet loss and distance weighted sampling, respectively

Table 3. Comparisons with various sampling methods on CARS

Method	Sampling	DAS	R@1	F1	NMI
Triplet	Random		74.21	33.41	64.28
		✓	76.79	35.21	65.49
	Semi-hard [42]	✓	77.10	33.82	65.03
	Soft-hard [41]		74.64	31.98	63.22
		✓	80.54	37.52	66.77
Distance [56]	✓	81.34	37.27	67.21	
Contrastive	Random		42.44	15.83	48.87
		✓	50.79	19.40	52.71
	Distance [56]	✓	80.70	35.47	66.01

Table 4. Comparisons with more related works on CARS

Method	Backbone	R@1	R@2	R@4	R@8
N-Pair + HDML [63]	G ⁵¹²	68.90	78.90	85.80	90.90
N-Pair + HDML-A [63]	G ⁵¹²	81.10	88.80	93.70	96.70
N-Pair + DAS (Ours)	G ⁵¹²	83.70	90.33	94.47	96.77
MS + SEC [61]	IBN ⁵¹²	85.73	91.96	95.51	97.54
MS + DAS (Ours)	IBN ⁵¹²	85.66	91.60	95.27	97.37
MS + SEC + DAS (Ours)	IBN ⁵¹²	87.80	93.16	96.18	98.01
Margin + DiVA [35]	IBN ⁵¹²	83.10	90.00	N/A	N/A
Margin + DAS (Ours)	IBN ⁵¹²	84.85	90.32	93.99	96.40
ProxyNCA++ [47]	R ⁵¹²	86.50	92.50	95.70	97.70
Margin + DiVA	R ⁵¹²	82.20	89.00	N/A	N/A
Margin + DRML [66]	R ⁵¹²	73.30	83.00	89.80	94.40
Margin + DCML [65]	R ⁵¹²	85.20	91.80	96.00	98.00
Margin + DAS (Ours)	R ⁵¹²	88.34	93.21	95.92	97.59

contrastive losses that are sensitive to the sampling methods. Therefore, various sampling approaches are tailored for them. The experiment results are presented in Table 3. For two loss functions, despite the choice of sampling approaches, DAS is able to improve them considerably on both image retrieval and clustering metrics. Notably, when applying DAS, triplet loss with random sampling outperform the one with the semi-hard sampling. This indicates that producing more embeddings for sampling is as important as the sampling approach.

4.4 Qualitative Results

To better understand our method, we compare contrastive loss (with distance weighted sampling) w/ or w/o DAS and visualize image retrieval results on both CARS (Fig. 6) and SOP (Fig. 6) datasets.⁶ On CARS, the model performs better with DAS despite the background noises or the interference from the car’s color, demonstrating that DAS enforces the model to focus on real semantics. On SOP, the model with DAS is insensitive to drastic viewpoint changes. These results verify the generalization ability and robustness of DAS under various scenes.

⁶ More qualitative results are in the supplementary.



Fig. 6. Top 3 retrieved results from the R^{128} trained w/ or w/o DAS. green and red rectangles indicate desired and undesired results, respectively

4.5 Ablation Studies

Effect of DFS and MTS. In this section, we perform ablation studies to evaluate the performance gain by each module in DAS. The loss function and sampling method are triplet loss and distance weighted sampling, respectively. The results are in Table 5. First, DFS, alone, boosts $R@1$ by +2.42%, verifying that producing embeddings around a single embedding is able to force the model to focus on real semantics and achieve better image retrieval results. Second, with MTS only, the clustering metrics are greatly improved, indicating that producing embeddings among multiple embeddings are beneficial to the image clustering task. Last, with DFS and DAS, all metrics are further improved, suggesting that DFS and MTS reinforce and complement each other.

Table 5. Ablation studies on CARS

DFS	MTS	$R@1$	F1	NMI
		78.86	35.80	65.85
✓		81.28 (+2.42)	36.22 (+0.42)	66.84 (+0.99)
	✓	81.83 (+2.97)	38.26 (+2.46)	67.81 (+1.96)
✓	✓	82.63 (+3.77)	39.14 (+3.34)	68.12 (+2.27)

Effect of K in DFS. Multi-dimensional embeddings have a diverse set of semantic features [2, 3]. The top- K mask is effective to discover the discriminative features. We perform experiments on SOP, a large-scale and diverse dataset with margin loss to support our claim. With $K = 1, 2, 4, 8, 16, 32$, we obtain results $R@1 = 78.76, 78.86, \mathbf{79.14}, 78.09, 77.94, 77.96$, where substantial improvements are observed when $K = \{1, 2, 4\}$ and larger K (*i.e.*, $K > 4$) leads to worse results.

Effect of Z in MTS. The larger bank capacity (Z) allows us to access intra-class transformations from current and previous iterations, which improves the diversity of the produced embeddings. We conduct experiments on CARS with margin loss by setting $Z = 1, 2, 3, 4, 5$ and obtain $R@1 = 82.35, 82.38, 82.42, \mathbf{82.75}, 82.33$, showing that history transformations ($Z > 2$) bring slight improvements.

4.6 Further Discussions

More discussions on DFS. One may question whether the proposed DFS requires the linear assumption on high dimensional feature space. In fact, we do not make this assumption and our method is built on a very basic hypothesis of metric learning: the embeddings that are close to each other in the embedding space have similar semantics. More critically, DFS does not rely on the linearity assumption. Instead, DFS is based on the observations that effective semantics for one embedding are highly activated features [2, 3].

More discussions on MTS. Li *et al.* [30] also apply a memory module is leveraged to store abundant features and conduct neighborhood search upon them to enhance the discriminative power of a general CNN feature on the image search and few-shot learning tasks. Unlike them, DAS constructs a memory bank with the intra-class embedding transformations, which allows us to access intra-class transformations from current and previous iterations, and thus improves the diversity of produced embeddings for DML.

Comparisons with more related works. In Table 4, we compare DAS with HDML [63], SEC [61], DiVA [35], ProxyNCA++ [47], DRML [66], DCML [65] on CARS under the same settings. We can see that DAS outperforms HDML, DiVA, DRML and DCML, and achieves comparable performance to SEC. Note that DAS can be also incorporated into SEC. As a result, using MS loss, SEC + DAS outperforms SEC by 2.07% in R@1. These results further verify the applicability of DAS on some regularization techniques in DML.

5 Conclusion

In this paper, we propose a Densely-Anchored Sampling (DAS) scheme to alleviate the “missing embedding” issue incurred during DML sampling. To this end, we propose to produce embeddings with no data points by exploiting the embeddings’ nearby embedding space. Specifically, we propose a DFS module that identifies an embedding’s discriminative features and performs random scaling on them to exploit the embedding space around it. Moreover, we propose a MTS module to exploit embedding space among multiple embedding by adding the intra-class semantic differences to embeddings of the same class. By combining the embeddings with and without data points, DAS provides more embeddings for sampling to improve the sampling quality and achieve effective DML. Extensive experiments with various loss functions and sampling methods on three public available benchmarks show that DAS is effective. In the future, we plan to apply DAS to other areas such as self-supervised learning that require sampling.

Acknowledgements. This work was partially supported by Peng Cheng Laboratory Research Project No. PCL2021A07, National Natural Science Foundation of China (NSFC) 62072190, Program for Guangdong Introducing Innovative and Entrepreneurial Teams 2017ZT07X183.

References

1. Aziere, N., Todorovic, S.: Ensemble deep manifold similarity learning using hard proxies. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7299–7307 (2019) [4](#)
2. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6541–6549 (2017) [3](#), [7](#), [13](#), [14](#)
3. Bau, D., Zhu, J.Y., Strobel, H., Lapedriza, A., Zhou, B., Torralba, A.: Understanding the role of individual units in a deep neural network. Proceedings of the National Academy of Sciences **117**(48), 30071–30078 (2020) [3](#), [7](#), [13](#), [14](#)
4. Byungsoo Ko, G.G., Kim, H.G.: Learning with memory-based virtual classes for deep metric learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) [10](#)
5. Chen, P., Huang, D., He, D., Long, X., Zeng, R., Wen, S., Tan, M., Gan, C.: Rspnet: Relative speed perception for unsupervised video representation learning. In: AAAI Conference on Artificial Intelligence, 2021 (2020) [2](#)
6. Chen, P., Zhang, Y., Tan, M., Xiao, H., Huang, D., Gan, C.: Generating visually aligned sound from videos. IEEE Transactions on Image Processing **29**, 8292–8302 (2020) [2](#)
7. Chen, W., Chen, X., Zhang, J., Huang, K.: Beyond triplet loss: a deep quadruplet network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 403–412 (2017) [2](#)
8. Chu, P., Bian, X., Liu, S., Ling, H.: Feature space augmentation for long-tailed data. In: Proceedings of the European Conference on Computer Vision. pp. 694–710. Springer (2020) [4](#), [5](#)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255. IEEE (2009) [10](#)
10. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2019) [1](#), [4](#)
11. DeVries, T., Taylor, G.W.: Dataset augmentation in feature space. arXiv preprint arXiv:1702.05538 (2017) [4](#)
12. Ding, Z., Fu, Y.: Robust transfer metric learning for image classification. IEEE Transactions on Image Processing **26**(2), 660–670 (2016) [2](#)
13. Duan, Y., Zheng, W., Lin, X., Lu, J., Zhou, J.: Deep adversarial metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2780–2789 (2018) [2](#), [3](#), [4](#)
14. Escorcia, V., Carlos Niebles, J., Ghanem, B.: On the relationship between visual attributes and convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1256–1264 (2015) [3](#), [7](#)
15. Ge, W.: Deep metric learning with hierarchical triplet loss. In: Proceedings of the European Conference on Computer Vision. pp. 269–285 (2018) [3](#), [10](#)
16. Gu, G., Ko, B., Kim, H.G.: Proxy synthesis: Learning with synthetic classes for deep metric learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1460–1468 (2021) [2](#)
17. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: International Joint Conference on Artificial Intelligence. pp. 1753–1759 (2017) [1](#)

18. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 1735–1742. IEEE (2006) [2](#), [3](#), [4](#)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016) [10](#)
20. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017) [3](#), [4](#), [11](#)
21. Hoi, S.C., Liu, W., Chang, S.F.: Semi-supervised distance metric learning for collaborative image retrieval and clustering. ACM Transactions on Multimedia Computing, Communications, and Applications **6**(3), 1–26 (2010) [1](#)
22. Hu, J., Lu, J., Tan, Y.P.: Discriminative deep metric learning for face verification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1875–1882 (2014) [4](#)
23. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456. PMLR (2015) [10](#)
24. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 117–128 (2010) [10](#)
25. Kim, S., Kim, D., Cho, M., Kwak, S.: Proxy anchor loss for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3238–3247 (2020) [4](#), [10](#)
26. Kim, S., Seo, M., Laptev, I., Cho, M., Kwak, S.: Deep metric learning beyond binary supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2288–2297 (2019) [4](#)
27. Kim, W., Goyal, B., Chawla, K., Lee, J., Kwon, K.: Attention-based ensemble for deep metric learning. In: Proceedings of the European Conference on Computer Vision. pp. 736–751 (2018) [10](#)
28. Ko, B., Gu, G.: Embedding expansion: Augmentation in embedding space for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7255–7264 (2020) [2](#), [3](#), [4](#), [10](#), [11](#)
29. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 554–561 (2013) [9](#)
30. Li, S., Chen, D., Liu, B., Yu, N., Zhao, R.: Memory-based neighbourhood embedding for visual recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6102–6111 (2019) [14](#)
31. Lin, X., Duan, Y., Dong, Q., Lu, J., Zhou, J.: Deep variational metric learning. In: Proceedings of the European Conference on Computer Vision. pp. 689–704 (2018) [3](#), [4](#), [8](#)
32. Liu, L., Cao, J., Liu, M., Guo, Y., Chen, Q., Tan, M.: Dynamic extension nets for few-shot semantic segmentation. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 1441–1449 (2020) [2](#)
33. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 212–220 (2017) [4](#)
34. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 746–751 (2013) [3](#), [8](#)

35. Milbich, T., Roth, K., Bharadhwaj, H., Sinha, S., Bengio, Y., Ommer, B., Cohen, J.P.: Diva: Diverse visual feature aggregation for deep metric learning. In: European Conference on Computer Vision. pp. 590–607. Springer (2020) [12](#), [14](#)
36. Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 360–368 (2017) [2](#), [4](#)
37. Musgrave, K., Belongie, S., Lim, S.N.: A metric learning reality check. In: European Conference on Computer Vision. pp. 681–699. Springer (2020) [2](#)
38. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4004–4012 (2016) [2](#), [4](#), [9](#)
39. Opitz, M., Waltner, G., Possegger, H., Bischof, H.: Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42**(2), 276–290 (2018) [10](#)
40. Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., Jin, R.: Softtriple loss: Deep metric learning without triplet sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6450–6458 (2019) [4](#), [10](#)
41. Roth, K., Milbich, T., Sinha, S., Gupta, P., Ommer, B., Cohen, J.P.: Revisiting training strategies and generalization performance in deep metric learning. In: International Conference on Machine Learning. pp. 8242–8252. PMLR (2020) [2](#), [3](#), [8](#), [12](#)
42. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 815–823 (2015) [1](#), [2](#), [3](#), [4](#), [5](#), [11](#), [12](#)
43. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to Information Retrieval, vol. 39. Cambridge University Press Cambridge (2008) [10](#)
44. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. *Journal of Big Data* **6**(1), 1–48 (2019) [4](#)
45. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 1857–1865 (2016) [2](#), [3](#), [4](#), [10](#), [11](#)
46. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015) [10](#)
47. Teh, E.W., DeVries, T., Taylor, G.W.: Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In: European Conference on Computer Vision. pp. 448–464. Springer (2020) [12](#), [14](#)
48. Ustinova, E., Lempitsky, V.: Learning deep embeddings with histogram loss. In: Proceedings of the International Conference on Neural Information Processing Systems. pp. 4177–4185 (2016) [2](#)
49. Volpi, R., Morerio, P., Savarese, S., Murino, V.: Adversarial feature augmentation for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5495–5504 (2018) [4](#), [5](#)
50. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011) [9](#)
51. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1386–1393 (2014) [4](#)

52. Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R., Robertson, N.M.: Ranked list loss for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5207–5216 (2019) [4](#), [10](#)
53. Wang, X., Han, X., Huang, W., Dong, D., Scott, M.R.: Multi-similarity loss with general pair weighting for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5022–5030 (2019) [2](#), [3](#), [4](#), [10](#), [11](#)
54. Wang, X., Zhang, H., Huang, W., Scott, M.R.: Cross-batch memory for embedding learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6388–6397 (2020) [2](#), [3](#), [4](#), [10](#)
55. Wang, Y., Pan, X., Song, S., Zhang, H., Huang, G., Wu, C.: Implicit semantic data augmentation for deep networks. *Advances in Neural Information Processing Systems* **32**, 12635–12644 (2019) [4](#), [5](#)
56. Wu, C.Y., Manmatha, R., Smola, A.J., Krahenbuhl, P.: Sampling matters in deep embedding learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2840–2848 (2017) [2](#), [3](#), [4](#), [5](#), [10](#), [11](#), [12](#)
57. Yin, X., Yu, X., Sohn, K., Liu, X., Chandraker, M.: Feature transfer learning for face recognition with under-represented data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5704–5713 (2019) [4](#), [5](#)
58. Yu, B., Tao, D.: Deep metric learning with tuplet margin loss. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6490–6499 (2019) [4](#)
59. Yuan, Y., Yang, K., Zhang, C.: Hard-aware deeply cascaded embedding. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 814–823 (2017) [10](#)
60. Zhai, A., Wu, H.Y.: Classification is a strong baseline for deep metric learning (2019) [4](#)
61. Zhang, D., Li, Y., Zhang, Z.: Deep metric learning with spherical embedding. *Advances in Neural Information Processing Systems* **33**, 18772–18783 (2020) [12](#), [14](#)
62. Zhao, Y., Jin, Z., Qi, G.j., Lu, H., Hua, X.s.: An adversarial approach to hard triplet generation. In: Proceedings of the European Conference on Computer Vision. pp. 501–517 (2018) [4](#)
63. Zheng, W., Lu, J., Zhou, J.: Hardness-aware deep metric learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**(9), 3214–3228 (2021) [12](#), [14](#)
64. Zheng, W., Chen, Z., Lu, J., Zhou, J.: Hardness-aware deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 72–81 (2019) [2](#), [3](#), [4](#)
65. Zheng, W., Wang, C., Lu, J., Zhou, J.: Deep compositional metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9320–9329 (2021) [12](#), [14](#)
66. Zheng, W., Zhang, B., Lu, J., Zhou, J.: Deep relational metric learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12065–12074 (2021) [12](#), [14](#)
67. Zhu, Y., Yang, M., Deng, C., Liu, W.: Fewer is more: A deep graph metric learning perspective using fewer proxies. *Advances in Neural Information Processing Systems* **33**, 17792–17803 (2020) [4](#), [10](#)

DAS: Densely-Anchored Sampling for Deep Metric Learning (Supplementary Materials)

Lizhao Liu^{1,2}, Shangxin Hunag¹, Zhuangwei Zhuang¹,
Ran Yang¹, Mingkui Tan^{1,3†}, and Yaowei Wang^{2†}

¹South China University of Technology ²PengCheng Laboratory

³Key Laboratory of Big Data and Intelligent Robot, Ministry of Education

{selizhaoliu,sevtars,z.zhuangwei,msyangran}@mail.scut.edu.cn,

mingkuitan@scut.edu.cn, wangyw@pcl.ac.cn

We organize our supplementary materials as follows. In Section **A**, we provide the detailed formulations of both pair-based and proxy-based DML loss functions. In Section **B**, we detail the formulations of DML sampling methods. In Section **C**, we provide more implementation details of DAS. In Section **D**, we analyze the overhead of DAS, In Section **E**, we provide experiment results of DAS on widely used proxy-based losses. In Section **F**, we provide results on DAS w/o image augmentation. In Section **G**, we study the effect of batch size on DAS. In Section **H**, we study the effect of embedding dimension on DAS. In Section **I**, we visualize and analyze the frequency recorder matrix. In Section **J**, we provide the evolution of training process w.r.t. more DML losses. In Section **K**, we investigate the effect of hyper-parameters r_s, r_b, T . In Section **L**, we provide qualitative results w.r.t. DFS and MTS. In Section **M**, we provide more qualitative results on different loss functions.

A Detailed Formulations of Loss Function in DML

A.1 Pair-based Loss Function

Contrastive Loss [2]. The goal of contrastive loss is simply pulling the embeddings of the same class as close as possible and separating the embeddings of different classes at least of a given margin. Specifically, contrastive loss requires the index set of the sampled embedding pairs $\mathcal{P} = \{(i, j)\}$ and the pair-wise euclidean distance is calculated as $\mathbf{D}_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\|$. Then the formulation of contrastive loss is as follows

$$\mathcal{L}_{\text{Contrastive}} = \sum_{(i,j) \in \mathcal{P}} \mathbb{I}\{y_i = y_j\} \mathbf{D}_{ij} + \mathbb{I}\{y_i \neq y_j\} [\gamma - \mathbf{D}_{ij}]_+, \quad (\text{I})$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, γ (set to 1.0 in this paper) is the margin.

Triplet Loss [8]. Triplet loss extends the contrastive loss by converting the absolute distance relationship between embeddings into a relative distance relationship (*i.e.*, ranking): the distance between embeddings of different classes

[†] Corresponding authors.

should be farther away than any embeddings of the same class. Specifically, triplet loss requires sampling a set of embedding triplets $\mathcal{T} = \{(a, p, n)\}$, where $y_a = y_p \neq y_n$ and a, p, n are the index of the anchor, positive and negative, respectively. The formulation of triplet loss is as follows

$$\mathcal{L}_{\text{Triplet}} = \sum_{(a,p,n) \in \mathcal{T}} [\mathbf{D}_{ap} - \mathbf{D}_{an} + \gamma]_+, \quad (\text{II})$$

where γ (set to 0.2 in this paper) is the margin.

Margin Loss [11]. Margin loss introduces a more flexible optimization paradigm into the triplet loss. Specifically, a adjustable and learnable margin $\beta \in \mathbb{R}^C$ is proposed to replace the fixed margin (*i.e.*, 0) between embedding of different classes, which converts the triplet ranking problem into a relative ordering of pairs. The formulation of margin loss is as follows

$$\mathcal{L}_{\text{Margin}} = \sum_{(i,j) \in \mathcal{P}} \mathbb{I}\{y_i = y_j\} [\gamma + \mathbf{D}_{ij} - \beta_{y_i}]_+ + \mathbb{I}\{y_i \neq y_j\} [\gamma + \beta_{y_i} - \mathbf{D}_{ij}]_+, \quad (\text{III})$$

where γ (set to 0.2 in this paper) is the margin in the triplet loss and β_{y_i} is the learnable margin for class y_i . Each element in β is initialized with 1.2 and the learning rate for β is set to $5e^{-4}$.

Generalized Lifted Structure Loss [3]. Generalized lifted structure loss extends the standard lifted structure loss [6] by considering all embeddings from the same class w.r.t. the anchor during intra-class distance minimization. Generalized lifted structure loss pulls embeddings of the same class w.r.t. the anchor close while pushing embeddings of different classes apart. To save computation cost, each embedding in a batch is used as the anchor once. To be specific, the index set of the sampled embeddings is $\mathcal{P} = \{(a, \mathcal{Q}, \mathcal{R})\}$, where $a \notin \mathcal{Q}, \mathcal{R}$, and $y_a = y_q \neq y_r, q \in \mathcal{Q}, r \in \mathcal{R}$. Then the formulation of generalized lifted structure loss is as follows

$$\mathcal{L}_{\text{GenLifted}} = \sum_{(a,\mathcal{Q},\mathcal{R}) \in \mathcal{P}} \left[\log \sum_{q \in \mathcal{Q}} \exp(\mathbf{D}_{aq}) + \log \sum_{r \in \mathcal{R}} \exp(\gamma - \mathbf{D}_{ar}) \right]_+ + \nu \|\mathbf{v}_a\|^2, \quad (\text{IV})$$

where γ (set to 1.0 in this paper) is the margin to avoid pushing the embeddings of different classes too large and ν (set to $5e^{-3}$ in this paper) regularizes the embeddings. Note that, in this loss, embeddings for distance computation and producing embeddings with no data points are not normalized.

N-Pair Loss [9]. N-Pair loss extends the triplet loss by considering all embeddings of different classes during inter-class distance maximization. Specifically, the index set of the sampled embeddings is $\mathcal{P} = \{(a, p, \mathcal{R})\}$, where $y_a = y_p \neq y_r, r \in \mathcal{R}$, and the pair-wise distance is calculated as $\mathbf{D}_{ij} = \mathbf{v}_i^T \mathbf{v}_j$. Then the

formulation of N-Pair loss is as follows

$$\mathcal{L}_{\text{N-Pair}} = \sum_{(a,p,\mathcal{R}) \in \mathcal{P}} \log \left(1 + \sum_{r \in \mathcal{R}} \exp(\mathbf{D}_{ar} - \mathbf{D}_{ap}) \right) + \nu \|\mathbf{v}_a\|^2, \quad (\text{V})$$

where ν (set to $5e^{-3}$ in this paper) controls the optimization strength on the embedding regularization. Note that, in this loss, embeddings for distance computation and producing embeddings with no data points are also not normalized.

Multi-similarity Loss [10]. Apart from considering simple anchor-positive, anchor-negative relationships, multi-similarity loss better leverages all embeddings in a batch by additionally considering positive-positive and negative-negative relationship. Also, to save computation cost, each embedding in a batch will only be used as anchor once. For an anchor a , let \mathcal{P}_a and \mathcal{N}_a denote its corresponding positive and negative embedding index sets, respectively. Given the pair-wise distance computed by $\mathbf{D}_{ij} = \mathbf{v}_i^T \mathbf{v}_j$, the sampled embedding index set is constructed as $\mathcal{P} = \{(a, \mathcal{Q}, \mathcal{R})\}$, where $a \notin \mathcal{Q}, \mathcal{R}$, $\mathcal{Q} = \{q \mid y_q = y_a, \mathbf{D}_{aq} > \min_{i \in \mathcal{P}_a} (\mathbf{D}_{ai} - \epsilon)\}$ and $\mathcal{R} = \{r \mid y_r \neq y_a, \mathbf{D}_{ar} < \max_{j \in \mathcal{N}_a} (\mathbf{D}_{aj} + \epsilon)\}$. Then the formulation of multi-similarity loss is as follows

$$\begin{aligned} \mathcal{L}_{\text{MS}} = \sum_{(a, \mathcal{Q}, \mathcal{R}) \in \mathcal{P}} & \frac{1}{\alpha} \log \left[1 + \sum_{q \in \mathcal{Q}} \exp(-\alpha (\mathbf{D}_{aq} - \lambda)) \right] \\ & + \frac{1}{\beta} \log \left[1 + \sum_{r \in \mathcal{R}} \exp(\beta (\mathbf{D}_{ar} - \lambda)) \right], \end{aligned} \quad (\text{VI})$$

where $\alpha, \beta, \lambda, \epsilon$ are hyper-parameters to be set. In this paper, we set $\alpha = 2, \beta = 40, \lambda = 5e^{-1}, \epsilon = 1e^{-1}$.

A.2 Proxy-based Loss Function

Softmax Loss [12]. Different from the pair-based loss function, softmax loss1 introduces a proxy *i.e.*, classifier for each class and optimizes the embedding by pulling it close to its proxy. The formulation of softmax loss is as follows:

$$\mathcal{L}_{\text{Softmax}} = - \sum_i \log \frac{\exp(\mathbf{W}_{y_i}^T \mathbf{v}_i / T)}{\sum_{c \in C} \exp(\mathbf{W}_c^T \mathbf{v}_i / T)}, \quad (\text{VII})$$

where $\mathbf{W} \in \mathbb{R}^{C \times d}$ is the classifier weight for all training classes. Since the embedding \mathbf{v}_i is normalized, a temperature T (set to $5e^{-2}$ in this paper) is used to boost the gradient. Moreover, the learning rate of \mathbf{W} is set to $1e^{-5}$ for CARS and CUB, $2e^{-3}$ for SOP.

ArcFace Loss [1]. ArcFace loss improves the vanilla softmax by adding an angular margin into embedding and its corresponding proxy to achieve more

compact intra-class representation. The formulation of ArcFace loss is as follows

$$\mathcal{L}_{\text{ArcFace}} = - \sum_i \log \frac{\exp(s \cdot \cos(\mathbf{W}_{y_i}^T \mathbf{v}_i + \gamma))}{\exp(s \cdot \cos(\mathbf{W}_{y_i}^T \mathbf{v}_i + \gamma)) + \sum_{c \neq y_i} \exp(s \cdot \cos(\mathbf{W}_c^T \mathbf{v}_i))}, \quad (\text{VIII})$$

where $\mathbf{W} \in \mathbb{R}^{C \times d}$ is the classifier weight for all training classes and γ, s are hyper-parameters to be set. In this paper, we set $\gamma = 5e^{-1}$ and $s = 16$. Moreover, the learning rate of \mathbf{W} is set to $5e - 4$ for all datasets.

B Detailed Formulations of Sampling Method for DML

Random Sampling [4]. Random sampling simply selects the index of positive pair or negative pair in a most trivial way *i.e.*, randomly selecting. To be specific, given an embedding \mathbf{v}_i , its index of positive is randomly draw from $\{j \mid y_i = y_j, i \neq j\}$ and its index of negative is randomly draw from $\{k \mid y_i \neq y_k, i \neq k\}$.

Semi-hard Sampling [8]. Semi-hard sampling is proposed to effectively sample embedding triplets that grows cubically to batch size. In the training process, most of the triplets satisfy the objective function and they provide limited (or no) training signal to train the model, thereby impeding the model learning [8]. Thus, given an anchor \mathbf{v}_a and its positive \mathbf{v}_p (randomly sampled), semi-hard sampling carefully choose negative embedding’s index as follows

$$n \sim \{i \mid y_i \neq y_a, \|\mathbf{v}_a - \mathbf{v}_p\|^2 < \|\mathbf{v}_a - \mathbf{v}_i\|^2\}. \quad (\text{IX})$$

Soft-hard Sampling [7] To avoid selecting “hard” embeddings that impedes model training, semi-hard sampling chooses embeddings that are relatively close to the anchor. Soft-hard triplet sampling shows that a probabilistic (soft) selection of potentially hard embeddings is actually beneficial. Given an anchor embedding \mathbf{v}_a , soft-hard sampling attain the indexes of positive and negative embedding as follows

$$p \sim \{i \mid y_i = y_a, \|\mathbf{v}_a - \mathbf{v}_i\|^2 > \arg \min_{q \in \mathcal{Q}_a} \|\mathbf{v}_a - \mathbf{v}_q\|^2\}, \quad (\text{X})$$

$$n \sim \{j \mid y_j \neq y_a, \|\mathbf{v}_a - \mathbf{v}_j\|^2 < \arg \max_{r \in \mathcal{R}_a} \|\mathbf{v}_a - \mathbf{v}_r\|^2\}, \quad (\text{XI})$$

where $\mathcal{R}_a = \{r \mid y_r \neq y_a\}$, $\mathcal{Q}_a = \{q \mid y_q = y_a\}$ are the positive and negative index sets w.r.t. the anchor a , respectively. In this way, soft-hard sampling explores more triplets than semi-hard sampling to improve the model training.

Distance-weighted Sampling [11]. Different from other sampling strategy that considers a certain distance range of embeddings, distance-weighted sampling considers a wide range of embeddings in a probabilistic way. Since the

embedding space is typically a d -dimensional hypersphere \mathbb{S}^{d-1} , the analytical distribution of pairwise distance on a hypersphere obeys

$$q(\mathbf{D}_{ij}) \propto \mathbf{D}_{ij}^{d-2} \left[1 - \frac{1}{4}\mathbf{D}_{ij}\right]^{\frac{d-3}{2}}, \quad (\text{XII})$$

and $\mathbf{D}_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\|$ for any embedding pairs $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{S}^{d-1}$. To obtain a wide range of negative embeddings that are able to improve the embedding diversity as well as model training, distance-weighted sampling acquires the index of negative embedding based on the inversed distance distribution

$$P(n | a) \propto \min(\lambda, q^{-1}(\mathbf{D}_{an})). \quad (\text{XIII})$$

In this paper, we set $\lambda = 5e^{-1}$ and the largest distance to 1.4.

C More Implementation Details

In this section, we provide more implementation details. As for image augmentation process, random crop (image size 224×224) with random horizontal flip ($p = 0.5$) is applied during training and single center crop (image size 256×256) is used for testing. In terms of training strategy, the number of training epochs is 300. We use Adam [5] as the optimizer. The initial learning rate is $1e^{-5}$, which is reduced by a factor of 0.3 in 200th and 250th epoch, respectively. The weight decay is $4e^{-4}$. For batch preparation, SPC-2 construction [7] is used (2 samples per category). The batch size is set to 112.

D Efficiency and Overhead Analysis

DAS takes extra cost only in the training stage. Specifically, w/ and w/o DAS, the training time cost for [11] are 1.15s *vs.* 0.70s per batch, which includes the cost of DAS and using more embeddings for sampling and loss computation. Moreover, DAS only consumes 13% of the total time, which is efficient compared to the whole training procedure.

E Effectiveness of DAS on Proxy-based Loss

Although DAS is developed for pair-based loss, we perform experiments to evaluate the generalization ability of DAS on classic and widely used proxy-based losses *i.e.*, Softmax and ArcFace. The results are presented in Table I. The improvements are still observed when equipped with DAS for Softmax and ArcFace across different datasets.

F DAS w/o Image Augmentation

We further perform experiments without image augmentation using triplet loss and distance weighted sampling on CARS. The results are shown in Table II. DAS boosts all metrics considerably, showing that DAS is complementary to image augmentation technique.

Table I: Comparisons with proxy-based approaches on various datasets

Method	CUB			CARS			SOP		
	R@1	F1	NMI	R@1	F1	NMI	R@1	F1	NMI
Softmax [12]	61.58	36.12	66.73	79.07	37.11	67.01	77.92	37.20	90.05
Softmax + DAS	62.02	36.24	67.42	81.23	39.95	68.91	79.36	38.72	90.40
ArcFace [1]	61.56	35.73	66.83	79.50	37.75	67.82	78.08	37.79	90.18
ArcFace + DAS	62.80	37.63	67.80	82.22	40.82	69.82	78.12	38.08	90.26

Table II: Comparisons without image augmentation on CARS

DAS R@1	F1	NMI
61.47	22.06	53.88
✓ 65.13 (+ 3.66)	23.77 (+ 1.71)	55.89 (+ 2.01)

G Effect of Batch Size

In this section, we investigate the effect of batch size on the proposed DAS. The results are presented in Fig. I. The loss function and sampling method are margin loss [11] and distance-weighted sampling [11], respectively. From Fig. I, we have the following observations: **First**, under various batch size and image retrieval evaluation metrics, when equipped with DAS, the model is able to consistently obtain better results than the one trained without DAS. **Second**, we observe that the model trained with DAS and batch size = 32 outperforms the one trained without DAS and batch size = 224 in terms of R@1. It shows that producing effective embeddings without datapoints by DAS is as equally important as providing more data points in a batch to achieve improved performance. These results well prove the rationality of our motivation and the efficacy of DAS.

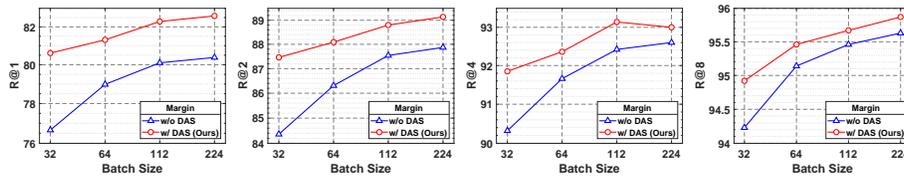
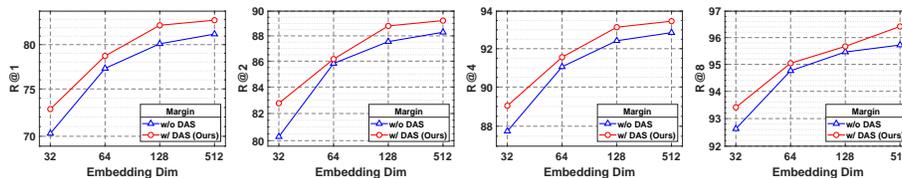


Fig. I: The test set R@{1, 2, 4, 8} on CARS with different batch size

H Effect of Embedding Dimension

In this section, we evaluate the proposed method on different embedding dimensions. The results are shown in Fig. II. We use the margin loss [11] as the

loss function while leveraging the distance-weighted sampling as the sampling method [11]. From Fig. II, we obtain the following results: **First**, under different embedding dimension, DAS consistently reaches the best performance for all image retrieval metrics. **Second**, the model that trained with DAS and embedding dimension = 64 obtains a comparable result like the one trained without DAS and embedding dimension = 128 regarding R@1. It shows that the produced embeddings by DAS are able to force the model to better leverage the model capacity. And covering the barren area in embedding space is important to get improved performance when model capacity is low. These results demonstrate the effectiveness of DAS.

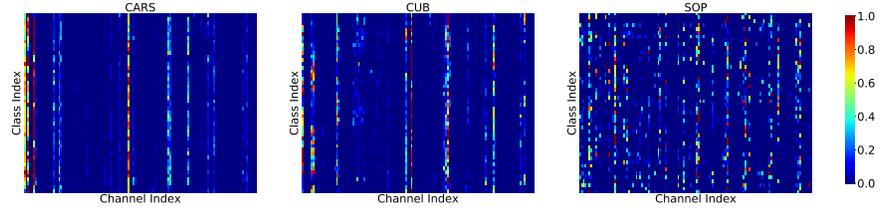


(a) Experiments using different embedding dim.

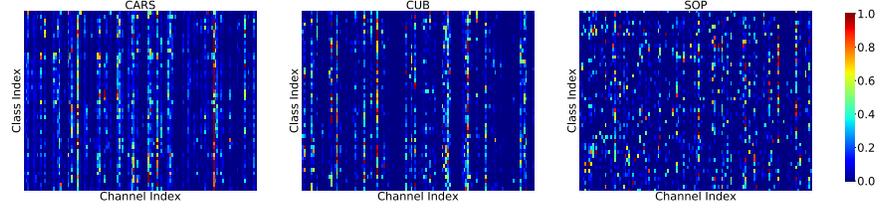
Fig. II: Test set R@{1, 2, 4, 8} on CARS with different embedding dimension d

I Visualization Results on Frequency Recorder Matrix

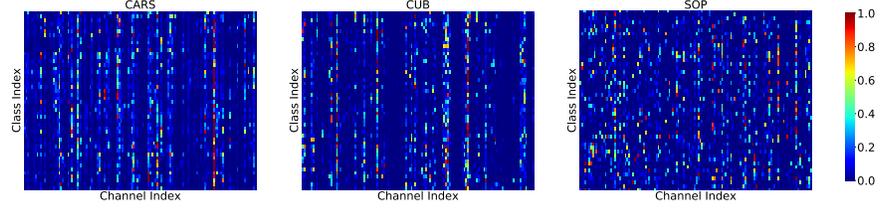
In this section, we visualize the Frequency Recorder Matrix (FRM) \mathbf{P} introduced in the DFS module. The FRM serves as a stable and effective identifier for semantic scaling by considering the top activated features for one class as the effective semantics instead of noises. The loss function and sampling method we used here are triplet loss [8] and distance-weighted sampling [11], respectively. We perform experiments on all three datasets (*i.e.*, CARS, CUB and SOP). The results are depicted in Fig. III, from which, we have the following observations: **First**, for different training stages (*i.e.*, epoch = 1, 150, 300), the number of top activated features for embeddings of the same classes are limited (*i.e.*, around 4 ~ 8) across all datasets. **Second**, as the training process proceeds, more features are likely to be the top activated features. **Third**, for the large scale dataset SOP, more features are likely to be the top activated ones due to the rich semantics covered by adequate data points. In this sense, with the proposed FRM, we are able to figure out channels with more discriminative power to achieve effective semantic scaling. These results demonstrate the rationality of the proposed FRM.



(a) Visualization of the frequency recorder matrix at epoch = 1



(b) Visualization of the frequency recorder matrix at epoch = 150



(c) Visualization of the frequency recorder matrix at epoch = 300

Fig. III: From left to right, the visualized FRM on CARS, CUB and SOP, respectively. Each element in \mathbf{P} is normalized (*i.e.*, divided by the maximum value in its row). Only the first 48 classes are presented due to page limit

J Evolution of Training Process w.r.t. Different Losses

In this section, we provide the evolution of training loss and test set R@1 w.r.t. different losses in the training process. The results are depicted in Fig. IV. We have the following observations: **First**, when training with DAS, the training losses are generally higher and decrease smoother than the baseline, which demonstrates that producing more embeddings by DAS is able to consistently provide training signal to train the model. **Second**, when equipped with DAS, the test set R@1s are higher than the baseline. **Third**, for some loss functions that face severe overfitting problems such as contrastive loss and generalized lifted structure loss, DAS is able to ease the overfitting problem. These results verify the effectiveness of the proposed method across different loss functions and sampling methods.

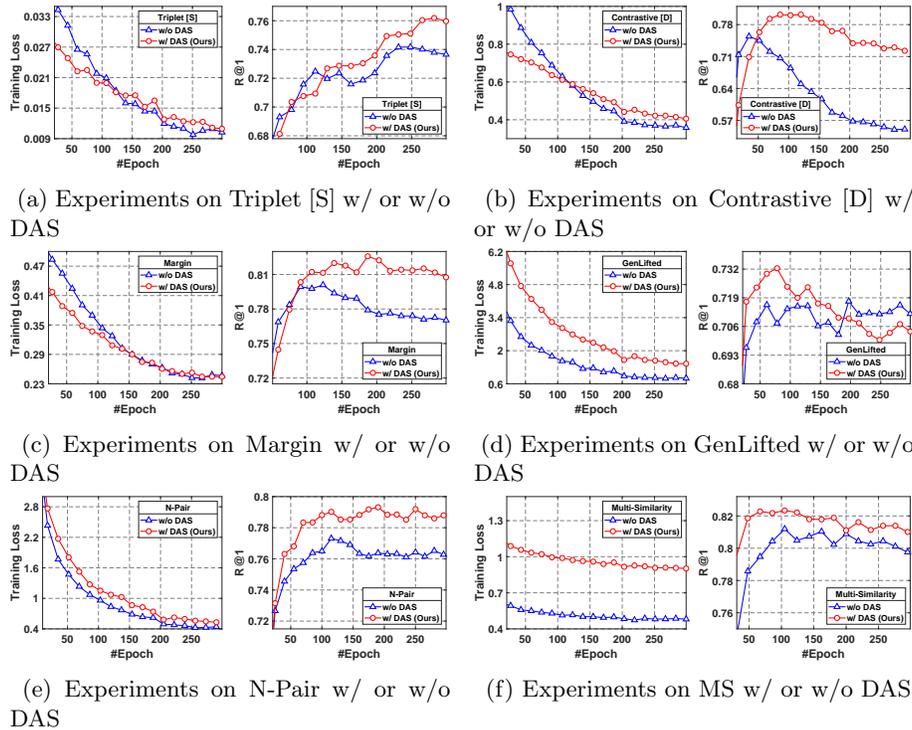


Fig. IV: The training loss and test set R@1 on CARS with different losses

K Ablation Studies on Hyper-parameters

In this section, we investigate the effect of the hyper-parameters r_s, r_b, T in DAS. The loss function and sampling method are margin loss [11] and distance-weighted sampling [11], respectively. The default hyper-parameters' settings are $(r_s, r_b, T) = (1e^{-2}, 1e^{-2}, 3)$.

Random scale in DFS (r_s). The results on different random scale in DFS are shown in Table III (a). Our method is insensitive to a wide range of random scale, showing that scaling the discriminative features is able to provide effective semantics of different strength.

Semantic shifting scale in MTS (r_b). r_b is to provide the flexibility of controlling the strength of adding intra-class semantic differences. The results on different semantic shifting scales in MTS are shown in Table III (b). Our method obtains similar results under different r_b and reaches the best results when $r_b = 1$, which suggests that larger r_b is able to cover more barren area in the embedding space to improve the model training.

Number of the produced embeddings T . The results on different numbers of the produced embeddings are shown in Table III (c). As T increases from 1 to 5, the proposed DAS achieves better results and reaches the best result at $T = 5$. When $T = 10$, the performance is worse than $T = 5$, which indicates that too many produced embeddings with no data points will dominate the optimization direction and impair the learning of embeddings with data points.

Table III: Experiments on different hyper-parameters on CARS

<p>(a) Effect of the random scale r_s in DFS</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="border-right: 1px solid black;">r_s</th> <th>$1e^{-2}$</th> <th>$1e^{-1}$</th> <th>$2e^{-1}$</th> <th>$5e^{-1}$</th> </tr> </thead> <tbody> <tr> <td style="border-right: 1px solid black;">R@1</td> <td>82.29</td> <td>82.25</td> <td>82.30</td> <td>82.43</td> </tr> </tbody> </table>	r_s	$1e^{-2}$	$1e^{-1}$	$2e^{-1}$	$5e^{-1}$	R@1	82.29	82.25	82.30	82.43	<p>(b) Effect of the scale r_b in MTS</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="border-right: 1px solid black;">r_b</th> <th>$1e^{-3}$</th> <th>$1e^{-2}$</th> <th>$1e^{-1}$</th> <th>1</th> </tr> </thead> <tbody> <tr> <td style="border-right: 1px solid black;">R@1</td> <td>82.19</td> <td>82.29</td> <td>82.07</td> <td>82.55</td> </tr> </tbody> </table>	r_b	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	1	R@1	82.19	82.29	82.07	82.55
r_s	$1e^{-2}$	$1e^{-1}$	$2e^{-1}$	$5e^{-1}$																	
R@1	82.29	82.25	82.30	82.43																	
r_b	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	1																	
R@1	82.19	82.29	82.07	82.55																	
<p>(c) Effect of the number of produced embedding (T) in DAS</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="border-right: 1px solid black;">T</th> <th>1</th> <th>3</th> <th>5</th> <th>10</th> </tr> </thead> <tbody> <tr> <td style="border-right: 1px solid black;">R@1</td> <td>80.78</td> <td>82.29</td> <td>83.40</td> <td>81.75</td> </tr> </tbody> </table>		T	1	3	5	10	R@1	80.78	82.29	83.40	81.75										
T	1	3	5	10																	
R@1	80.78	82.29	83.40	81.75																	

L Qualitative Results of DFS and MTS

In this section, we investigate the effectiveness of the proposed DFS and MTS. Specifically, we apply DFS and MTS in the test phase and compare results from the model trained with or without them. Since the training and test classes are different, the DFS and MTS modules used for training are unavailable here.

Thus, the semantic scaling is implemented as randomly scaling the top K features in an embedding; Whilst we perform semantic shifting by adding the transformation (obtained from another two embeddings of the same class) to the embedding. The loss function and sampling method we used here are contrastive loss [2] and distance-weighted sampling [11], respectively. The results for semantic scaling and shifting are shown in Fig. V and Fig. VI, respectively. We have the following observations: **1)** When we apply different semantic scaling to the query, the model trained with DAS consistently retrieves correct results, which is not the case for the baseline. **2)** The model trained with DAS is able to retrieve expected results even with the semantic shifted embedding while the baseline fails to do so. These results show that DAS is able to produce embeddings with effective semantics to train the model, which is insensitive to the semantic differences and consistently achieves good generalization ability after training.



Fig. V: Top 6 retrieved results with different scales on CARS. The expected and unexpected results are framed by green and red rectangles, respectively



Fig. VI: Top 3 retrieved results with MTS on CARS. The expected and unexpected results are framed by green and red rectangles, respectively

M More Qualitative Results

In this section, we provide qualitative results on different losses w/ or w/o DAS. The results for CARS and SOP are in Fig. VII and Fig. VIII, respectively. From those results, we can see that the proposed DAS can enforce the model to focus on real semantics despite the background noises and other semantics' interference such as car's colors, drastic viewpoint changes *etc.* These results show the generalization ability and robustness of the proposed DAS.



Fig. VII: Top 3 retrieved results using the model trained by different loss functions that are equipped w/ or w/o on CARS. The expected and unexpected results are framed by green and red rectangles, respectively



Fig. VIII: Top 3 retrieved results using the model trained by different loss functions that are equipped w/ or w/o on SOP. The expected and unexpected results are framed by green and red rectangles, respectively

References

1. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2019) [3](#), [6](#)
2. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 1735–1742. IEEE (2006) [1](#), [11](#)
3. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017) [2](#), [12](#), [13](#)
4. Hu, J., Lu, J., Tan, Y.P.: Discriminative deep metric learning for face verification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1875–1882 (2014) [4](#)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (2015) [5](#)
6. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4004–4012 (2016) [2](#)
7. Roth, K., Milbich, T., Sinha, S., Gupta, P., Ommer, B., Cohen, J.P.: Revisiting training strategies and generalization performance in deep metric learning. In: International Conference on Machine Learning. pp. 8242–8252. PMLR (2020) [4](#), [5](#)
8. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 815–823 (2015) [1](#), [4](#), [7](#), [12](#), [13](#)
9. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 1857–1865 (2016) [2](#), [12](#), [13](#)
10. Wang, X., Han, X., Huang, W., Dong, D., Scott, M.R.: Multi-similarity loss with general pair weighting for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5022–5030 (2019) [3](#), [12](#), [13](#)
11. Wu, C.Y., Manmatha, R., Smola, A.J., Krahenbuhl, P.: Sampling matters in deep embedding learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2840–2848 (2017) [2](#), [4](#), [5](#), [6](#), [7](#), [10](#), [11](#), [12](#), [13](#)
12. Zhai, A., Wu, H.Y.: Classification is a strong baseline for deep metric learning (2019) [3](#), [6](#)