Learning with Memory-based Virtual Classes for Deep Metric Learning

Byungsoo Ko*
NAVER/LINE Vision
kobiso62@gmail.com

Geonmo Gu*
NAVER/LINE Vision
korgm403@gmail.com

Han-Gyu Kim
NAVER Clova Speech
hangyu.kim@navercorp.com

Abstract

The core of deep metric learning (DML) involves learning visual similarities in high-dimensional embedding space. One of the main challenges is to generalize from seen classes of training data to unseen classes of test data. Recent works have focused on exploiting past embeddings to increase the number of instances for the seen classes. Such methods achieve performance improvement via augmentation, while the strong focus on seen classes still remains. This can be undesirable for DML, where training and test data exhibit entirely different classes. In this work, we present a novel training strategy for DML called MemVir. Unlike previous works, MemVir memorizes both embedding features and class weights to utilize them as additional virtual classes. The exploitation of virtual classes not only utilizes augmented information for training but also alleviates a strong focus on seen classes for better generalization. Moreover, we embed the idea of curriculum learning by slowly adding virtual classes for a gradual increase in learning difficulty, which improves the learning stability as well as the final performance. MemVir can be easily applied to many existing loss functions without any modification. Extensive experimental results on famous benchmarks demonstrate the superiority of MemVir over state-of-the-art competitors. Code of MemVir is publicly available[1].

1. Introduction

Deep metric learning (DML) is of great importance for learning visual similarities in a wide range of vision tasks, such as image clustering [17], unsupervised learning [4, 15, 5], and image retrieval [43, 10, 24, 12]. Learning visual similarity aims to build a well-generalized embedding space that reflects visual similarities of images using a defined distance metric. Typically, training and test data exhibit entirely different classes in DML. Thus, the main challenge is to maximize generalization performance from a training distribution to a shifted test distribution, which differs from classic classification tasks that deal with i.i.d. training and test distributions [31, 37].

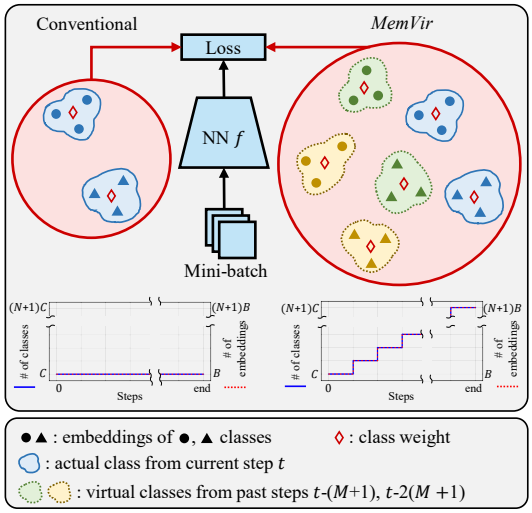


Figure 1. In conventional training, the loss function is computed with actual classes. On the other hand, in MemVir, classes from previous steps (virtual classes) are used to compute the loss function along with the actual classes. Moreover, the number of classes and embeddings are gradually increased by adding virtual classes, where C and B denote number of classes and batch size, N and M are hyper-parameters for MemVir.

Current DML approaches focus on learning visual similarities with objective functions, which considers pair-wise similarity (pair-based losses) [6, 38, 46] or similarity between samples and class representatives (proxy-based losses) [41, 40, 28, 7, 42]. Recent studies propose exploiting additional embeddings from past training steps, which are saved and controlled in the memory queue, to increase the number of samples in a mini-batch and that of hard negative pairs [15, 5, 45, 22]. And yet, these methods of utilizing past embeddings is still constrained to the seen classes of the training data. Thus, the trained model might result to over-fit to the seen classes while under-perform on the unseen classes in test data. Therefore, to learn an embedding space that generalizes, we need to alleviate the strong focus on seen classes during the training phase [37, 31, 30].

*Authors contributed equally.
[1] https://github.com/navervision/MemVir

In this paper, we propose a novel training strategy, which trains a model with Memory-based Virtual classes (MemVir), for DML. In MemVir, we maintain memory queues for both class weights and embedding features. Instead of using them to increase the number of instances of seen classes, they are treated as virtual classes to compute the loss function along with the actual classes, as illustrated in Figure 1. Moreover, we incorporate the idea of curriculum learning (CL) to gradually increase the learning difficulty by slowly adding virtual classes. The proposed MemVir has the following advantages: **1)** MemVir trains a model with augmented information, which includes increased number of classes ($C \rightarrow (N+1)C$) and instances ($B \rightarrow (N+1)B$) without additional feature extraction. **2)** CL-like gradually increasing the learning difficulty improves the optimization stability and final performance. **3)** Exploiting virtual classes help achieve more generalized embedding space by alleviating excessively strong focus on seen classes of training data. **4)** MemVir can be easily applied to many existing loss functions to obtain a significant performance boost without any modification of the loss function.

Contributions. **1)** We propose a novel training strategy for DML that exploits past embeddings and class weights as virtual classes to improve generalization. We further improve the training process and performance by incorporating the idea of CL. **2)** We exhaustively analyze our proposed method and demonstrate that employing virtual classes improves generalization by alleviating a strong focus on seen classes theoretically and empirically. **3)** MemVir achieves state-of-the-art performance on three popular benchmarks of DML in both conventional and *Metric Learning Reality Check (MLRC)* [33] evaluation protocol.

2. Related Work

Sample Generation and Memory-based Learning. In DML, the generation of hard samples has been investigated to perform training with more informative samples [8, 49, 12, 24]. DAML [8] and HDML [49] utilize generative networks to generate synthetic samples, while Symm [12] and EE [24] generate synthetic samples by geometric relations. Meanwhile, utilizing information from previous steps has been explored in many computer vision tasks [15, 5, 45, 22]. In supervised DML, XBM [45] is proposed to use memorized embeddings for extending negative samples in pair-based losses. In XBM, the state difference between past and current embeddings is disregarded based on "*slow drift*" phenomena. On the other hand, [22] argues that a large accumulated error caused by the state difference may degrade the training process. They present BroadFace method for softmax variant losses to control the error by compensating the state difference and gradient control. The above-mentioned methods focus on utilizing generated or memorized information with respect to increasing the number of instances for the seen classes. However, this may result in a model overly optimized to the seen classes while under-performing on the unseen classes in test data. Rather than disregarding or controlling the state difference, the proposed MemVir exploits the state difference by employing the memorized embeddings and class weights as *virtual classes*, which are treated as different classes from the actual (seen) classes. The exploitation of virtual classes helps achieve more generalized embedding space by alleviating a strong focus on seen classes. Additional comparison with XBM w.r.t "*slow drift*" phenomena is included in supplementary Section B.2.

Virtual Class. In image recognition task, Virtual softmax [3] has been presented to enhance the discriminative property of embeddings by injecting a virtual class into the softmax loss. However, it is not only limited by a single virtual class but also cannot be used with softmax variants using l_2-normalization. In comparison, MemVir exploits multiple virtual classes and can be used with any softmax variants and proxy-based losses.

Curriculum Learning. CL in machine learning is motivated by the idea of curriculum in human learning, where the models learn from easier samples first and more difficult samples later. Imposing CL for model training has been shown to accelerate and improve the training process in many machine learning tasks [1, 47, 13, 18]. When exploiting CL, two key factors have to be considered: (1) Scoring the difficulty of each sample; (2) scheduling the pace by which the sample is presented to the network. To define the difficulty, bootstrapping and transfer learning have been used to score the difficulty of each sample [47, 13]. For scheduling, the samples to be presented to the network can be determined in fixed or adaptive steps [47, 18]. The main difference between conventional CL and MemVir is the former schedules within the training data, whereas the latter (MemVir) increases the learning difficulty with virtual classes, which are augmented information.

3. Proposed Method

3.1. Preliminary

We define a deep neural network as $f : \boldsymbol{\mathcal{I}} \rightarrow \boldsymbol{\mathcal{X}}$, which is a mapping from an input data space $\boldsymbol{\mathcal{I}}$ to an embedding space $\boldsymbol{\mathcal{X}}$. Let $X = [x_1, x_2, \ldots, x_H]$ denote the D-dimensional embedding features, and each feature x_i has a corresponding label $y_i \in \{1, \ldots, C\}$. The generalized form of the objective function can be written as follows:

$$\mathcal{L}(X, W) = -\frac{1}{\mid X \mid} \sum_{i=1}^{\mid X \mid} l(x_i, y_i), \tag{1}$$

where W denotes the class weights, and $l(\cdot)$ can be any of the loss functions defined below.

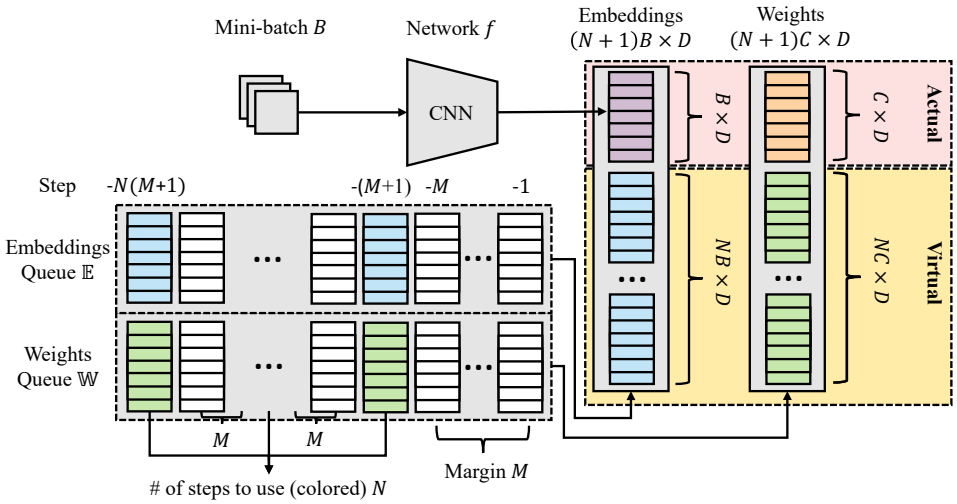


Figure 2. Overview of MemVir. Past embeddings and class weights queues are maintained. We select N steps of past embeddings and weights with margin M in between the selected steps, and use them as additional virtual classes along with actual classes for loss computation.

The most widely used classification loss function, softmax loss, has been revalued as a competitive objective function in metric learning [48, 2]. The softmax loss is used to optimize the network f and class weight W:

$$l_{softmax}(x_i, y_i) = \log \frac{e^{W_{y_i}^T x_i}}{\sum_{j=1}^{C} e^{W_j^T x_i}}, \tag{2}$$

where $W_j \in \mathbb{R}^D$ denotes the j-th column of weight $W \in \mathbb{R}^{D \times C}$. The bias b is set to 0 because it does not affect the performance [28, 7]. The weight W_j is the center of each class [7, 42] and serves as a representative.

For improved performance and better interpretation, [41, 40, 28] proposes to normalize weights and embedding features to lay them on a hypersphere with a fixed radius. We perform l_2-normalization to fix the size of the weights and embedding features to the following: $\| W_j \| = 1$ and feature $\| x_i \| = 1$. Subsequently, we can simplify the logits [35] by transforming $W_j^T x_i = \| W_j \| \| x_i \| \cos\theta_j = \cos\theta_j$, and define the Norm-softmax loss as follows:

$$l_{norm}(x_i, y_i) = \log \frac{e^{\gamma \cos\theta_{y_i}}}{e^{\gamma \cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{C} e^{\gamma \cos\theta_j}}, \tag{3}$$

where γ is a scale factor. The proposed method MemVir can be used with softmax variants as well as proxy-based losses because a proxy is a class representative feature much like class weights of softmax variants. Hence, we include the details of other loss functions (CosFace [42], ArcFace [7], CurricularFace [18], Proxy-NCA [32], and Proxy-Anchor [21]) in supplementary Section A.

3.2. Learning with Memory-based Virtual Classes

We propose a novel training strategy called MemVir, which trains a model with virtual classes from past steps to exploit augmented information and obtain better generalization. When conventional metric learning trains a model with given C classes and B embeddings from training data, MemVir gradually increases the number of classes ($C \rightarrow (N+1)C$) and embeddings ($B \rightarrow (N+1)B$) with the virtual classes. We use the naming convention of MemVir(N,M), which indicates the hyper-parameters of the proposed method, to be defined below.

Queuing Past Embeddings and Weights. To form a class in loss computation, a pair of the class representative feature (weight) and embedding features are required. Hence, in MemVir, we maintain two types of memory queues: embedding queue $\mathbb{E}$ and weight queue $\mathbb{W}$, where each entity of the queues is a collection of embeddings or class weights of each step as illustrated in Figure 2. For each step, the collection of embeddings X and weights W are enqueued to $\mathbb{E}$ and $\mathbb{W}$, respectively. The size of each queue is determined as $N(M+1)$, where N is the number of selected steps to use for the loss computation, and M is the margin between the selected steps. The shape and position of class clusters vary by each step because the network parameters change during training process. Such variance between steps is utilized in MemVir by exploiting weights and embeddings from previous steps as virtual classes. Here, the difference between the selected steps can be controlled by the margin M.

Scheduling Usage of Virtual Classes. In MemVir, virtual classes will be utilized to gradually increase learning difficulty as CL. The scheduling of virtual class usage includes two periods: *warm-up* and *step-pacing*. We turn on MemVir and begin managing queues after the warm-up step U (epoch U_e), because the embeddings of the initial phase are typically scattered without forming clusters,

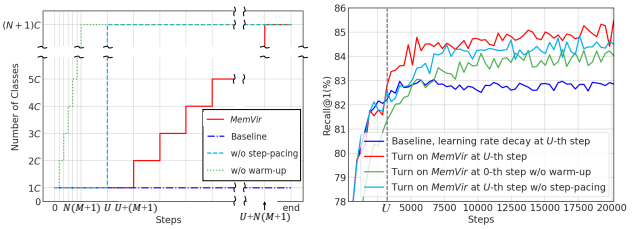
Algorithm 1: Pseudo-code of MemVir

```
// f: encoder network
// weight/embed_queue: weight and embedding memory queue
// U_e, N, M: warm-up epoch, number of steps, margin
for input, label in loader do
    embed = f.forward(input)
    weight = f.get_class_weight()
    // Turn on MemVir when it is in use and past warm-up epoch
    if MemVir is True and epoch ≥ U_e then
        cur_weight = weight.copy()
        cur_embed = embed.copy()
        cur_label = label.copy()
        // Prepare embeddings and weights by step-pacing
        // The order of each queue is from new to old
        if len(weight_queue) > M then
            for idx in range(M, len(weight_queue), M + 1) do
                pre_weight = weight_queue[idx]
                pre_embed, pre_label = embed_queue[idx]
                // Create new label indices for virtual classes
                new_label = create_new_label(pre_label)
                weight.concatenate(weight, pre_weight)
                embed.concatenate(embed, pre_embed)
                label.concatenate(label, new_label)
            end
        // Update memory queues
        enqueue(weight_queue, cur_weight)
        enqueue(embed_queue, (cur_embed, cur_label))
        if len(weight_queue) > N(M + 1) then
            dequeue(weight_queue)
            dequeue(embed_queue)
    // Compute loss and back-propagation
    loss = compute_loss(weight, embed, label)
    loss.backward()
    optimizer.step()
end
```

which can be a distraction for training. It is noteworthy that we use MemVir without learning rate decay because decaying the learning rate changes the difference between steps; thus, the learning rate decay can be used with a modification of hyper-parameter M of MemVir. After the warm-up, the step-pacing algorithm takes place by storing embeddings and weights of each step in their respective queues and reusing them for loss computations, as described in Algorithm 1. As the queue size grows, previously stored embeddings and weights from every $M+1$ steps are selected as virtual classes when computing the loss at each step. The number of selected steps for virtual classes would increase gradually from 0 to N determined by current queue size. This results in increasing the number of classes by a staircase function, and the function s of the number of classes can be written as:

$$s(i) = \begin{cases} C, & i < U, \\ C \times \left\{ min(\lfloor \frac{i-U}{M+1} \rfloor, N) + 1 \right\}, & i \geq U, \end{cases} \quad (4)$$

where i denotes the current step. The scheduling function of MemVir is illustrated by the red line in Figure 3a.



(a) Different ways of scheduling. (b) Performance by scheduling.

Figure 3. Impact of scheduling. (a) Different ways of scheduling for adding virtual classes. (b) Performance of each scheduling case with MemVir(5,100) and Norm-softmax as baseline on CARS196.

Learning with Multiple Virtual Classes. When we select N steps of past embeddings and weights from the queues, it indicates that we have NC virtual classes. We denote the set of selected past embeddings and weights as $\widetilde{X}$ and $\widetilde{W}$, respectively. Subsequently, we compute the objective function with virtual classes as follows:

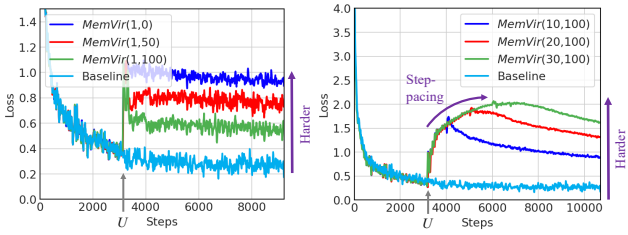
$$\mathcal{L}(X \cup \widetilde{X}, W \cup \widetilde{W}) = -\frac{1}{\mid X \cup \widetilde{X} \mid} \sum_{i=1}^{|X \cup \widetilde{X}|} l(x_i, y_i), \quad (5)$$

where $l(\cdot)$ can be any type of loss function. The implementation of MemVir is simple without any modification of the loss function, and it gives a significant performance improvement in DML without any additional computational cost in the inference phase.

3.3. Discussion and Analysis

3.3.1 Analysis of Scheduling

Figure 3 shows the different ways of scheduling and the performance of each case. In Figure 3a, when the MemVir is turned on at warm-up step U, it begins adding virtual classes after each $M+1$ step, gradually. Compared with MemVir, 'w/o warm-up' starts adding virtual classes from the initial steps, whereas 'w/o step-pacing' adds all virtual classes at once after warm-up step U. For the case of 'w/o warm-up', training starts with degraded performance, but finally, the performance is higher than the baseline. In fact, embeddings from virtual classes at the initial steps would be scattered without forming clusters; thus, it can be a distraction at the initial steps. Meanwhile, 'w/o step-pacing' exhibits a slight performance degradation immediately after warm-up step U. This is because placing NC number of virtual classes simultaneously can be too difficult for training the model. By considering both cases, MemVir is able to increase the training difficulty gradually for a more stable optimization.



(a) Difficulty by margin. (b) Difficulty by number of steps.

Figure 4. Impact of difficulty with Norm-softmax as baseline on CARS196. Difficulty is measured by loss value of each step. (a) Difficulty by varying margin parameter M with a fixed number of steps $N = 1$. (b) Difficulty by varying the number of steps N with a fixed margin of $M = 100$.

3.3.2 Analysis of Difficulty

MemVir controls learning difficulty via following hyper-parameters: number of steps N and margin M. To see the impact of learning difficulty by each hyper-parameter, we measure the difficulty with the loss value by following [27, 47]. As shown in Figure 4a, a smaller margin of M results in greater difficulty, which is obvious because the embeddings from the recent steps would be similar to the embeddings from the current steps. Furthermore, Figure 4b shows that adding more virtual classes increases the learning difficulty. It is noteworthy that the loss value increases slowly after warm-up step U by adding virtual classes gradually (step-pacing); subsequently, it starts decreasing after reaching a peak. The detailed performance by different hyper-parameters is presented in Section 4.4.

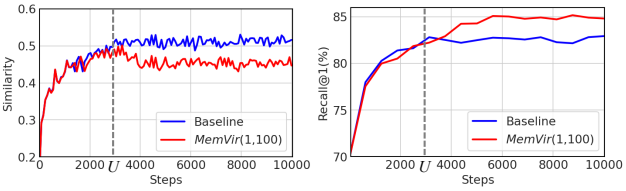
3.3.3 Gradient Analysis for Generalization

Considering the distribution shift in training and test data, strong focus on seen classes has to be alleviated in the generalization of transfer learning problems such as DML [37, 31, 30]. To demonstrate how MemVir works in generalizing models during training, we have analyzed the gradient of the softmax loss. For convenient analysis, the softmax loss in Equation 2 is re-written as follows:

$$l_{softmax}(x_i, y_i) = \log \frac{e^{\alpha(x_i,y_i)}}{\sum_{j=1}^{C} e^{\alpha(x_i,j)}}, \quad (6)$$

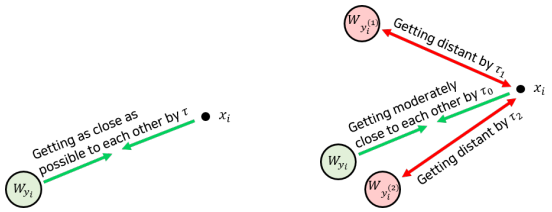
where $\alpha(x_i, j) = W_j^T x_i$. The gradient of the softmax loss over the embedding feature x_i can be inducted as follows:

$$\begin{aligned} \frac{\partial l_{softmax}(x_i, y_i)}{\partial x_i} &= W_{y_i} - \frac{\sum_{j=1}^{C} e^{\alpha(x_i,j)} W_j}{\sum_{j=1}^{C} e^{\alpha(x_i,j)}} \\ &\approx W_{y_i} - \frac{e^{\alpha(x_i,y_i)} W_{y_i}}{\sum_{j=1}^{C} e^{\alpha(x_i,j)}} \\ &= \tau W_{y_i}, \end{aligned} \quad (7)$$



(a) Cosine similarity (x_i, W_{y_i}). (b) Generalization performance.

Figure 5. Generalization analysis with Norm-softmax as baseline on CARS196. (a) Similarity between embeddings and corresponding class weights of seen classes in training data. (b) Performance on unseen classes in test data.



(a) Softmax loss. (b) MemVir + softmax loss.

Figure 6. Illustration of an embedding (x_i) and corresponding class weight (W_{y_i}) learning, where $W_{y_i^{(n)}}$ are virtual class weights originated from the class y_i.

$$\tau = 1 - \frac{e^{\alpha(x_i,y_i)}}{\sum_{j=1}^{C} e^{\alpha(x_i,j)}}. \quad (8)$$

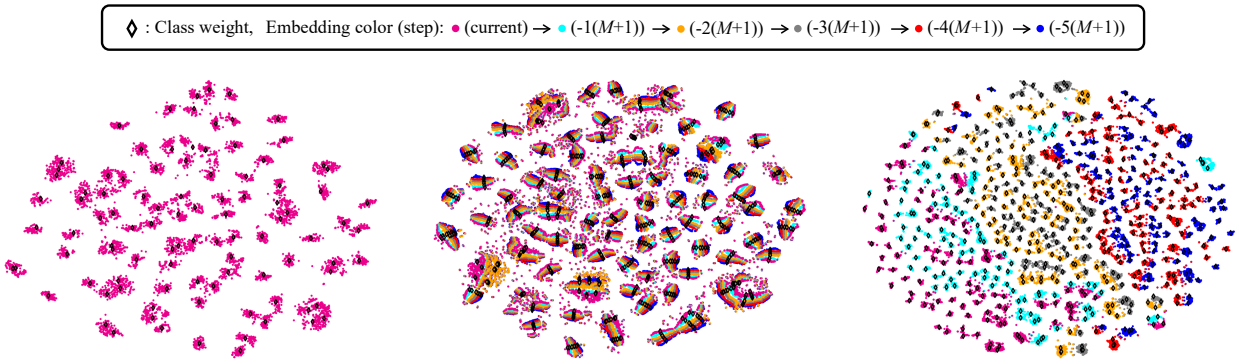
It is obvious that $\tau > 0$ and $\tau \to 0$ when $x_i \to W_{y_i}$, implying that x_i tries to get as close to W_{y_i} as possible, which is illustrated in Figure 6a. This can result in a strong focus on the target weight W_{y_i} and an over-fit to the seen classes of the training data.

In comparison, the gradient of MemVir + softmax loss over the embedding feature x_i can be inducted as follows:

$$\begin{aligned} \frac{\partial l_{MemVir}(x_i, y_i)}{\partial x_i} &= W_{y_i} - \frac{\sum_{j=1}^{(N+1)C} e^{\alpha(x_i,j)} W_j}{\sum_{j=1}^{(N+1)C} e^{\alpha(x_i,j)}} \\ &\approx W_{y_i} - \frac{\sum_{n=0}^{N} e^{\alpha(x_i,y_i^{(n)})} W_{y_i^{(n)}}}{\sum_{j=1}^{(N+1)C} e^{\alpha(x_i,j)}} \\ &= \tau_0 W_{y_i} + \sum_{n=1}^{N} \tau_n W_{y_i^{(n)}}, \end{aligned} \quad (9)$$

$$\tau_0 = 1 - \frac{e^{\alpha(x_i,y_i)}}{\sum_{j=1}^{(N+1)C} e^{\alpha(x_i,j)}}, \tau_n = -\frac{e^{\alpha(x_i,y_i^{(n)})}}{\sum_{j=1}^{(N+1)C} e^{\alpha(x_i,j)}} \quad (10)$$

where, $y_i^{(n)} (n > 0)$ are virtual classes and $y_i^{(0)} = y_i$. It is obvious that $\tau_0 > 0$. However, τ_0 would not be close to zero whether x_i is nearby W_{y_i} or not, because the denominator



(a) 50th epoch, $\#\ of\ classes = C$ (b) 60th epoch, $\#\ of\ classes = 6C$ (c) 200th epoch, $\#\ of\ classes = 6C$

Figure 7. t-SNE visualization of 512-dimensional embedding space. Embedding features are extracted by model trained with MemVir(5,100) on CARS196 training data. Each color indicates step for embedding features.

of τ_0 would be large as the virtual classes are close to W_{y_i}. As illustrated in Figure 6b, this makes it difficult for x_i to get highly close to W_{y_i} and thus, alleviates the phenomenon of the embedding feature becoming extremely close to the target W_{y_i}. In addition, because $\tau_n < 0$, x_i tries to get farther away from the virtual classes $W_{y_i^{(n)}}$. Thus, the alleviation would be more extensive and can effectively ease the intense focus of the softmax loss, leading to a more substantial generalization. This is empirically shown in Figure 5. The baseline gradually increases the similarity between the embeddings and corresponding class weights. By contrast, when MemVir is turned on at step U, the similarity is slightly degraded by alleviating the strong focus on the seen classes, and better generalization is achieved as shown in Figure 5b. The detailed induction is provided in the supplementary Section B.1.

4. Experiments

In this section, we conduct a series of experiments to analyze and validate the effectiveness of MemVir. Please refer to the supplementary material for additional experiments: analysis of memory and computational cost (Section D.1), impact of learning rate (Section D.2), impact of warm-up (Section D.3), robustness to input deformation (Section D.4), impact of embeddings and class weights in virtual class (Section D.6), and more.

4.1. Experimental Setting

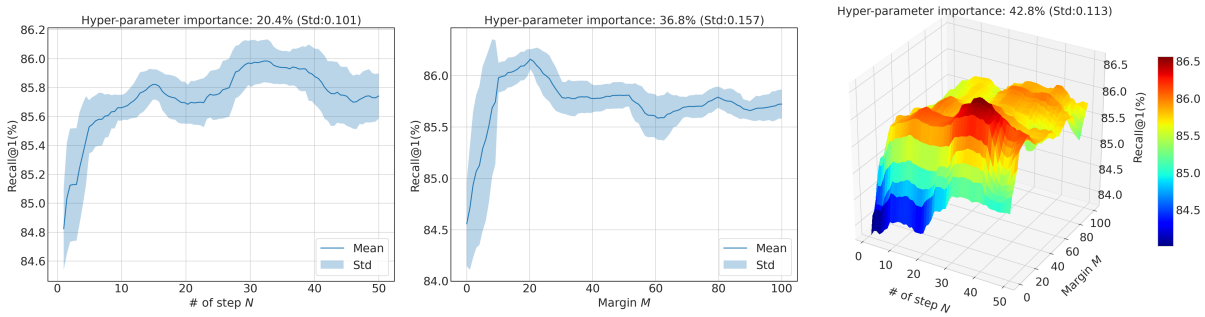
We use three popular datasets for evaluation in DML: CUB-200-2011 (CUB200) [39], CARS196 [25], and Stanford Online Products (SOP) [34]. We perform two types of evaluation procedures: *conventional evaluation* and *MLRC evaluation*. Conventional evaluation is based on the common training and evaluation procedure described in [34, 21]. All experiments are conducted on an Inception network with batch normalization [20] and a 512-dimensional embedding feature. A batch size of 128, the Adam optimizer [23] with a learning rate of 10^{-4}, and warm-up epoch $U_e = 50$ are adopted unless otherwise noted in the experiment. Considering recent works [33, 9] that have proposed improved evaluation procedures for fairness, we include the MLRC evaluation protocol [33]. In MLRC evaluation, the procedure includes hyper-parameter search with 4-fold cross-validation, ensemble evaluation, and the usage of fair metrics (P@1, RP, and MAP@R). Please refer to supplementary Section C for details regarding the datasets and implementation.

4.2. Embedding Space Visualization

In Figure 7, we visualize the embedding space of the training data via t-SNE [29] to present how MemVir learns the embedding space. At the 50th epoch in Figure 7a, the model has been trained with only actual classes and obtains sparse embedding space with concentration on the actual classes. When all virtual classes are added at the 60th epoch in Figure 7b, virtual classes tend to be close to the actual classes and the embedding space is still sparse as in Figure 7a. This demonstrates that the model is not fully utilizing the embedding space and is highly focused on the seen classes. After enough epochs of training, at the 200th epoch in Figure 7c, the model obtains dense embedding space with sufficient discriminative power over all actual and virtual classes. To sum up, MemVir offers better utilization of embedding space by alleviating strong focus on seen classes for generalization. We include extended visualization in supplementary Section D.8.

4.3. Impact of Batch Size and Number of Classes

One advantage of MemVir is that it can utilize augmented information, including an increased number of embedding features and classes without additional feature extraction. To see the impact of the number of embedding



(a) Impact of number of step N (b) Impact of margin M (c) Impact of pair-wise interaction

Figure 8. We use fANOVA [19] to estimate the impact of hyper-parameters. Reported performances are predicted values from random forest of fANOVA, which is trained with experimental results of MemVir on CARS196.

Batch size	8	16	32	64	128	256	512	1024
Norm-softmax	79.1	82.8	83.1	**83.5**	83.3	82.8	81.0	78.5
+ *MemVir*	80.4	83.6	85.0	**85.5**	85.0	85.0	84.8	84.6
Diff	+1.3	+0.8	+1.9	+2.0	+1.7	+2.2	+3.8	**+6.1**

(a) Impact of batch size.

Class ratio (%)	10	20	30	40	50	60	70	80	90	100
Norm-softmax	56.4	67.3	69.6	74.8	77.7	78.8	79.4	81.7	82.0	**83.3**
+ *MemVir*	58.5	70.1	72.8	77.2	80.0	81.3	82.6	83.8	84.1	**85.0**
Diff	+2.1	+2.8	**+3.2**	+2.4	+2.3	+2.5	**+3.2**	+2.1	+2.1	+1.7

(b) Impact of number of classes.

Table 1. Impact of batch size and number of classes on CARS196 dataset. We report Recall@1(%) performance and underline when MemVir(1,100) exceeds the best performance of the baseline Norm-softmax.

features and classes, we conduct experiments by varying the batch size and number of classes, where the training classes are randomly sampled by class ratio. As shown in Table 1a, the performance of the Norm-softmax baseline increases from the batch size of 8 to 64 and then decreases after, indicating that the increase in the batch size does not guarantee performance improvement [26, 11]. Applying MemVir to the baselines allows the models to learn with twice the number of embedding features by the virtual classes. MemVir yields performance improvement by 2.5% on average and exceeds the best performance of the baseline of batch size 64 with the batch size of only 16. Moreover, we observe that using MemVir is more robust to performance degradation due to the large batch size. As shown in Table 1b, decreasing the class ratio degrades the performance of the Norm-softmax baseline from 83.3% to 56.4%. With MemVir, which doubles the number of classes with virtual classes, we observe that the performance increases by an average of 2.4% and exceeds the best performance of the baseline with only 80% of the classes.

4.4. Impact of Hyper-parameters

For hyper-parameter analysis, we use the fANOVA framework [19], which can estimate the pattern and importance of each hyper-parameter and pair-wise interaction. We report the hyper-parameter analysis of CUB200 and SOP as well as the details of the fANOVA in the supplementary Section C.3 and D.5. As illustrated in Figure 8, the performance on CARS196 improves as the number of steps N increases. The performance improves until the margin $M = 20$, and then stabilizes after a slight degradation. However, the patterns of the impact of the hyper-parameters differ for each dataset because the characteristics of each dataset and the number of classes are diverse. We observe two common patterns among all datasets. First, a margin M larger than zero is typically better than $M = 0$; this is because classes from adjacent steps would be too similar to act as different classes and hence become distractions. Second, N exceeding one is typically better than $N = 1$. This is because by using more steps N, the effect of CL can be exploited more effectively by scheduling addition of virtual classes with a longer time.

4.5. Comparison with Related Methods

We compare MemVir with related methods from image recognition task, including the virtual class (Virtual softmax [3]), the memory-based (BroadFace [22]), and the CL (CurricularFace [18]) methods. Also, we include XBM [45] from DML to compare with BroadFace. For a fair comparison, we follow the experimental setting of [22, 18], which consists of a stochastic gradient descent (SGD) optimizer, a learning rate of 5×10^{-3}, a batch size of 512, and the ResNet50 backbone [16]. As presented in Table 3, Virtual softmax degrades the performance, whereas MemVir + softmax improves the performances of both datasets. When we combine XBM with ArcFace, we observe performance degradation when the memory size is large, as reported in BroadFace [22]. The performance can be further improved

Method	CUB200			CARS196			SOP		
	P@1	RP	MAP@R	P@1	RP	MAP@R	P@1	RP	MAP@R
Norm-softmax [41]	65.65 ± 0.30	35.99 ± 0.15	25.25 ± 0.13	83.16 ± 0.25	36.20 ± 0.26	26.00 ± 0.30	75.67 ± 0.17	50.01 ± 0.22	47.13 ± 0.22
MemVir + Norm-softmax	**69.22 ± 0.15**	**37.92 ± 0.16**	**27.10 ± 0.13**	**85.81 ± 0.18**	**38.78 ± 0.19**	**28.92 ± 0.17**	**75.77 ± 0.20**	**50.24 ± 0.22**	**47.45 ± 0.25**
CosFace [42]	67.32 ± 0.32	37.49 ± 0.21	26.70 ± 0.23	85.52 ± 0.24	37.32 ± 0.28	27.57 ± 0.30	75.79 ± 0.14	49.77 ± 0.19	46.92 ± 0.19
MemVir + CosFace	**69.79 ± 0.26**	**37.85 ± 0.23**	**27.08 ± 0.28**	**87.57 ± 0.13**	**39.10 ± 0.21**	**29.56 ± 0.26**	**75.88 ± 0.27**	**49.95 ± 0.37**	**47.18 ± 0.38**
ArcFace [7]	67.50 ± 0.25	37.31 ± 0.21	26.45 ± 0.20	85.44 ± 0.28	37.02 ± 0.29	27.22 ± 0.30	**76.20 ± 0.27**	50.27 ± 0.38	47.41 ± 0.40
MemVir + ArcFace	**69.33 ± 0.41**	**37.82 ± 0.28**	**26.96 ± 0.25**	**88.02 ± 0.18**	**39.12 ± 0.15**	**29.63 ± 0.15**	76.05 ± 0.30	**50.56 ± 0.33**	**47.75 ± 0.32**
Proxy-NCA [32]	65.69 ± 0.43	35.14 ± 0.26	24.21 ± 0.27	83.56 ± 0.27	35.62 ± 0.28	25.38 ± 0.31	75.89 ± 0.17	50.10 ± 0.22	47.22 ± 0.21
MemVir + Proxy-NCA	**69.25 ± 0.32**	**37.31 ± 0.12**	**26.43 ± 0.17**	**87.02 ± 0.15**	**38.51 ± 0.15**	**28.76 ± 0.16**	**76.97 ± 0.31**	**50.81 ± 0.26**	**48.02 ± 0.27**
Proxy-anchor [21]	69.73 ± 0.31	38.23 ± 0.37	27.44 ± 0.35	86.20 ± 0.21	39.08 ± 0.31	29.37 ± 0.29	75.37 ± 0.15	50.19 ± 0.14	47.25 ± 0.15
MemVir + Proxy-anchor	**69.81 ± 0.28**	**38.57 ± 0.14**	**27.83 ± 0.16**	**86.40 ± 0.18**	**40.27 ± 0.20**	**30.58 ± 0.20**	**77.80 ± 0.17**	**53.21 ± 0.12**	**50.35 ± 0.13**

Table 2. **[MLRC evaluation]** Performance (%) on three famous datasets in image retrieval task. We report the performance of concatenated 512-dim over 10 training runs. Bold numbers indicate the best score within the same loss and dataset.

Method	CARS196			SOP		
	R@1	R@2	R@4	R@1	R@10	R@100
SoftMax	78.3	86.4	91.9	76.6	89.4	95.8
Virtual SoftMax	75.1	84.1	90.1	74.5	87.9	94.8
MemVir + SoftMax	**79.2**	**87.0**	**92.1**	**78.9**	**90.6**	**96.2**
ArcFace	78.8	86.4	91.7	76.9	89.1	95.0
XBM + ArcFace	78.9	86.2	91.9	78.1	89.7	95.8
BroadFace + ArcFace	79.5	87.3	92.0	80.2	91.0	95.9
MemVir + ArcFace	**80.7**	**88.1**	**92.7**	**80.8**	**91.3**	**96.5**
CurricularFace	79.9	87.3	92.0	79.8	90.7	95.6
MemVir + CurricularFace	**81.0**	**87.9**	**92.9**	**81.3**	**91.7**	**98.8**

Table 3. Performance (%) comparison with related methods on CARS196 and SOP dataset.

by adding compensation technique and gradient control presented in BroadFace. However, exploiting the memorized features as virtual classes in MemVir shows a higher performance boost than just utilizing them for the increased number of instances in XBM and BroadFace. Considering that CurricularFace has already included the idea of CL, MemVir can improve the performance even further by providing virtual classes as harder cases. Moreover, it is noteworthy that the experimental results show the flexibility of MemVir for different types of backbones and optimizers. Extended experiments with different experimental settings are presented in the supplementary Section D.7.

4.6. Comparison with State-of-the-art

Finally, we compare the proposed method with state-of-the-art methods in DML. In the conventional evaluation shown in Table 4, every softmax variant and proxy-based loss combined with MemVir show significantly improved performance in every dataset. The average performance improvements are 2.3%, 3.4%, and 1.1% for CUB200, CARS196 and SOP, respectively. In comparison with the memory-based (XBM), sample generation (Symm, EE), and other recent methods (MS, SoftTriple and ProxyGML), MemVir shows competitive performance in all datasets. Even in the MLRC evaluation shown in Table 2, which is specifically designed in terms of fairness, MemVir improves performance in every dataset and metric substantially. These results demonstrate the flexibility and effectiveness of MemVir in DML. Please refer to the supplementary Section D.9 for extended results of the metrics and comparisons with existing methods in conventional evaluation, as well as additional performance report of separated 128-dim in MLRC evaluation.

Method	CUB200	CARS196	SOP
Multi-similarity (MS)† [44]	64.5	82.1	76.3
SoftTriple [36]	65.4	84.5	78.3
ProxyGML [50]	66.6	85.5	78.0
Symm [12] + MS [38]	64.9	82.4	76.9
EE [24] + MS [44]	65.1	82.9	77.0
XBM [45] + Contrastive [14]	65.8	82.0	79.5
Softmax	64.2	81.5	76.3
MemVir + Softmax	66.8 (+2.6)	86.5 (+5.0)	77.8 (+1.5)
Norm-softmax [41]	64.9	83.3	78.6
MemVir + Norm-softmax	67.3 (+2.4)	**86.8 (+3.5)**	79.6 (+1.0)
CosFace [42]	65.7	83.6	78.6
MemVir + CosFace	67.7 (+2.0)	86.6 (+3.0)	79.7 (+1.1)
ArcFace [7]	66.1	83.7	78.8
MemVir + ArcFace	67.4 (+1.3)	86.5 (+2.8)	**80.0 (+1.2)**
Proxy-NCA [32]	64.3	82.0	78.1
MemVir + Proxy-NCA	68.3 (+4.0)	86.5 (+4.5)	79.2 (+1.1)
Proxy-anchor† [21]	67.7	84.9	78.9
MemVir + Proxy-anchor	**69.0 (+1.3)**	86.7 (+1.8)	79.7 (+0.8)

Table 4. **[Conventional evaluation]** Recall@1 (%) on three famous datasets in image retrieval task. † denotes evaluation in a fair setting described in supplementary Section C.2.1.

5. Conclusion

In this paper, we have presented a novel training strategy that exploits memory-based virtual classes and incorporates the idea of CL. Theoretical and empirical analysis demonstrates that employing virtual classes as augmented information help achieve better generalization by alleviating a strong focus on seen classes. Furthermore, we show that gradually increasing the learning difficulty by slowly adding virtual classes improves the training process and final performance. Considering that MemVir is easily applicable to existing loss functions for better generalization, it is hence a competitive training strategy in DML.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[2] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. *arXiv preprint arXiv:2003.08983*, 2020.

[3] Binghui Chen, Weihong Deng, and Haifeng Shen. Virtual class enhanced discriminative embedding learning. In *Advances in Neural Information Processing Systems*, pages 1942–1952, 2018.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

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

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[8] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018.

[9] Istvan Fehervari, Avinash Ravichandran, and Srikar Appalaraju. Unbiased evaluation of deep metric learning algorithms. *arXiv preprint arXiv:1911.12528*, 2019.

[10] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016.

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[12] Geonmo Gu and Byungsoo Ko. Symmetrical synthesis for deep metric learning. *arXiv preprint arXiv:2001.11658*, 2020.

[13] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. *arXiv preprint arXiv:1904.03626*, 2019.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

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

[17] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, 2016.

[18] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: adaptive curriculum learning loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5901–5910, 2020.

[19] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR, 2014.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. *arXiv preprint arXiv:2003.13911*, 2020.

[22] Yonghyun Kim, Wonpyo Park, and Jongju Shin. Broadface: Looking at tens of thousands of people at once for face recognition. *arXiv preprint arXiv:2008.06674*, 2020.

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

[24] Byungsoo Ko and Geonmo Gu. Embedding expansion: Augmentation in embedding space for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7255–7264, 2020.

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

[26] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[27] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in neural information processing systems*, pages 1189–1197, 2010.

[28] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[29] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[30] Timo Milbich, Karsten Roth, Homanga Bharadhwaj, Samarth Sinha, Yoshua Bengio, Björn Ommer, and Joseph Paul Cohen. Diva: Diverse visual feature aggregation fordeep metric learning. *arXiv preprint arXiv:2004.13458*, 2020.

[31] Timo Milbich, Karsten Roth, Biagio Brattoli, and Björn Ommer. Sharing matters for generalization in deep metric learning. *arXiv preprint arXiv:2004.05582*, 2020.

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

[33] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. *arXiv preprint arXiv:2003.08505*, 2020.

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

[35] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

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

[37] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjoern Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. *arXiv preprint arXiv:2002.08473*, 2020.

[38] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016.

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

[40] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018.

[41] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017.

[42] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[43] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

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

[45] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6388–6397, 2020.

[46] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[47] Daphna Weinshall, Gad Cohen, and Dan Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. *arXiv preprint arXiv:1802.03796*, 2018.

[48] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018.

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

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