Informative Sample-Aware Proxy for Deep Metric Learning

Aoyu Li
aoyuli@rio.gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Ikuro Sato
isato@c.titech.ac.jp
Tokyo Institute of Technology
Denso IT Laboratory
Tokyo, Japan

Kohta Ishikawa
ishikawa.kohta@core.d-itlab.co.jp
Denso IT Laboratory
Tokyo, Japan

Rei Kawakami
reikawa@sc.e.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Rio Yokota
rioyokota@gsic.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

ABSTRACT

Among various supervised deep metric learning methods proxy-based approaches have achieved high retrieval accuracies. Proxies, which are class-representative points in an embedding space, receive updates based on proxy-sample similarities in a similar manner to sample representations. In existing methods, a relatively small number of samples can produce large gradient magnitudes (*i.e.*, hard samples), and a relatively large number of samples can produce small gradient magnitudes (*i.e.*, easy samples); these can play a major part in updates. Assuming that acquiring too much sensitivity to such extreme sets of samples would deteriorate the generalizability of a method, we propose a novel proxy-based method called Informative Sample-Aware Proxy (Proxy-ISA), which directly modifies a gradient weighting factor for each sample using a scheduled threshold function, so that the model is more sensitive to the informative samples. Extensive experiments on the CUB-200-2011, Cars-196, Stanford Online Products and In-shop Clothes Retrieval datasets demonstrate the superiority of Proxy-ISA compared with the state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Image representations**.

KEYWORDS

metric learning, adaptive sampling, robust estimation

ACM Reference Format:
Aoyu Li, Ikuro Sato, Kohta Ishikawa, Rei Kawakami, and Rio Yokota. 2022. Informative Sample-Aware Proxy for Deep Metric Learning. In *ACM Multimedia Asia (MMAsia '22), December 13–16, 2022, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3551626.3564942

MMAsia '22, December 13–16, 2022, Tokyo, Japan

ACM ISBN 978-1-4503-9478-9/22/12…$15.00
https://doi.org/10.1145/3551626.3564942

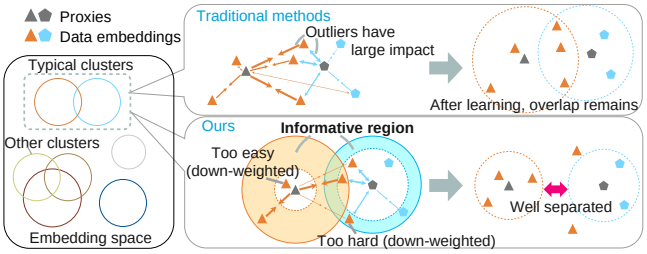


Figure 1: Overview of typical proxy-based DML and the proposed approach in this paper. Traditional methods can result in a few hard samples pulling the proxy, exacerbating the overlapping problem. The Proxy-ISA proposed in this paper aims to address this limitation by identifying such samples and reducing their contribution to the proxy.

1 INTRODUCTION

Learning semantic similarities between a pair of visual data is an important objective in computer vision tasks. Metric learning aims to learn an embedding space such that samples from the same category are close to each other and otherwise far apart. In deep metric learning (DML), learnable encoders are utilized to produce representations in an embedding space in which semantic distances between samples can be measured. DML has broad computer-vision applications such as image retrieval [30, 32], face recognition [40], person re-identification [42] and few-shot learning [4, 16, 43].

Throughout the past studies, DML methods can be categorized into two families: *pair-based* and *proxy-based*. Pair-based methods consider the distance between a pair of data embeddings, while proxy-based methods consider the distance between a data embedding and a class-representative point in the embedding space.

A typical example of pair-based losses is contrastive loss [6], which aims to pull close or push apart a pair of data embeddings according to the identicality of the class labels. Other pair-based losses, such as triplet loss [13] and N-pair loss [32], extend this idea to more than two samples. Due to its combinatorial nature, a naive implementation of a pair-based method suffers from polynomial growth of complexity with respect to the number of training samples. Additionally, training with pairs from a subset (mini-batch) lacks global information of the embedding space. In contrast, proxy-based methods [8, 26, 29, 36] assign one or more trainable reference

points called proxies to each class, and reduce the training complexity by considering only the distance from the data embeddings to proxies. During training, proxies attract data embeddings of the same class (*i.e.*, positives) and push away those of different classes (*i.e.*, negatives). Meanwhile, proxies are kept in memory and thus serve as a source of global information.

In existing methods, a small number of *hard* samples, such as positives that are far apart and negatives that are close, produce large gradients, and thus will have more impact on learning. A large number of *easy* samples can also dominate the gradient because of their large population size. An intuitive strategy is to select only the *hard* samples [12], but this has been observed to produce noisy gradients and converge to bad local optima [41], and its impact on proxies remains to be discussed. In pair-based methods, mining strategies, such as semi-hard negative mining [31], have been proposed to select and learn more important negatives that are closer to the anchor but still farther away than the corresponding positives. However, the *hardness* is usually manually defined by a numerical threshold, is fixed during learning, and is shared among different classes. As illustrated in Fig. 1, increasing the gradient weights of hard proxy-data pairs may also disrupt the ideal distribution of the clusters.

To address the limitations of existing proxy-based methods and to effectively incorporate the idea of sample importance into proxy-based DML, we propose the Informative Sample-Aware Proxy (Proxy-ISA) that directly controls gradient weights non-uniformly according to the temporal learning states, which are defined by how the samples with different hardness distribute in the embedding space. Our idea is motivated by empirical findings in active learning [5], where informative samples dynamically change along learning. In Proxy-ISA, as outlined in Fig. 1, each proxy is assigned a temporal similarity range, determined by the space already learned (too easy), informative region, and the space contains outliers (too hard); weights for positive and negative pairs are assigned separately. We apply a memory queue to estimate each class hardness. Unlike memory-based DML [20, 39], where past information is directly used to update weights, our use of memory allows us to estimate the space occupied by easy samples around the proxies and to judge whether the sample of interest is informative.

Our main contributions are summarized as follows:

- We propose a novel method for proxy-based DML, which performs class-dependent dynamic weighting for each sample based on the learned intra- and inter-class relations.
- We apply the concept of a "*volume*" of the class-related region [7] to DML and demonstrate empirically that focusing on informative samples based on the estimation of the class hardness improves generalization.
- Proxy-ISA achieves state-of-the-art performance on four public benchmarks of DML in both standard and *Metric Learning Reality Check* (MLRC) [27] evaluation settings.

2 RELATED WORK

Pair Mining / Pair Weighting in DML Non-uniform sampling [1, 2, 15, 17] and weighting [5, 22, 23] methods have been shown to improve the performance of deep neural networks and have been widely applied to various tasks. In pair-based DML, many sampling strategies [9, 41] and a general pair weighting (GPW) framework [37] have been developed.

Hard sample mining is the most widely discussed sampling strategy for pair-based DML. For example, Wu *et al.*[41] showed that pair-wise distance distribution in the embedding space is biased, and proposed *distance weighted sampling* to sample negative pairs uniformly according to the pair-wise distance within the mini-batch. Additionally, the deep sampler network (DSN) [9] was proposed to learn an adaptive sampling distribution based on the prior relations between training samples in the feature embedding network. However, the architecture of DSN relies on the design of the loss function, and online sampling increases the training cost. Wang *et al.*[37] proposed the general pair weighting (GPW) framework, which casts the sampling problem of deep metric learning into a unified view of pair weighting through gradient analysis. They proposed the Multi-Similarity (MS) loss to generate non-uniform pair weights based on various pair relations.

Since hard mining is easily influenced by outliers that lead the learned model to a bad local optima [31], methods such as Density Aware DML [10] and Class-Aware Attention [38] were proposed to identify outliers and reduce their impact on learning. In Hardness-Aware DML [44], the hardness threshold was scheduled by a global loss to perform adaptive mining. Although hardness threshold and weighting score have been applied in existing methods, the proposed thresholds and weights were not adaptive to class hardness, which can vary during training. Additionally, these methods provide pair-wise information by exploiting data-to-data relations within mini-batches, which results in the deficiency of global information; thus, the corresponding learning signal is sub-optimal.

Proxy-Based Deep Metric Learning The use of proxy is raised by Proxy-NCA [26], which combines the NCA loss [11] with a proxy. In its standard setting, one proxy is assigned to each class, and the raw data point is encouraged to be close to a positive proxy and far from all negative proxies. Alternatively, Proxy-Anchor [18] regards each proxy as an anchor, and intra-class relations are treated similarly to the MS loss [37]. More recent studies [29, 45] also consider assigning more proxies to each class to capture intra-class variance. For example, Qian *et al.*[29] extended the SoftMax loss to DML with multiple class centers, and proposed SoftTriple loss, which reflects the local geometry for each class. ProxyGML [45] applies the proxy in a graph structure manner, and multiple proxies are selected to construct a subgraph to help raw data points learn the neighbor structure. Although memory-based learning [20] was introduced to proxy-based methods for better generalization, existing methods do not consider the learning stability of the proxy itself, and still do not simultaneously address both intra- and inter-class relations in the loss function.

Rather than regarding the proxies as class representatives or graph nodes, the proposed Proxy-ISA treats each proxy as the center of a class-related region (subspace), and the definition of all the regions are estimated separately according to the proxies. As the learning difficulty varies from class to class, the definition of regions also varies even within the same iteration. In the proposed method, the proxy-data pairs are treated non-uniformly and adaptively based on the learned intra- and inter-class relations, resulting in better handling of global information.

3 PROPOSED APPROACH

In this section, we first revisit the GPW framework [37] to provide a unified view of DML loss functions, and qualitatively discuss the impact and problems of hard samples on the learning of the proxies. We then propose our gradient weighting method for proxy-based loss by introducing an estimation logic for hardness that adaptively changes along learning to ameliorate the problems.

3.1 The Gradient Weighting Mechanism

Let $\boldsymbol{x}_i \in \mathbb{R}^D$ be a real-value sample vector, and $y_i \in \{1, 2, \ldots, C\}$ be the corresponding label. Then the input matrix and the label vector for m training samples can be denoted as $\boldsymbol{X} \in \mathbb{R}^{m \times D}$ and $\boldsymbol{y} \in \mathbb{R}^m$, respectively. Let $\boldsymbol{p}_c \in \mathbb{R}^d$ be the proxy for class c ($c = 1, 2, \ldots, C$). We denote a deep neural network by $f(\cdot; \boldsymbol{\theta}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$, where $\boldsymbol{\theta}$ denotes the corresponding parameters and d is the embedding dimension. Then the cosine similarity between a data embedding vector and a proxy can be defined as $\mathcal{S}_{i,c} := \langle f(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{p}_c \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the dot product. Similarly, the similarity matrix can be denoted as $\boldsymbol{S} \in \mathbb{R}^{m \times C}$.

Given a proxy-based loss $\mathcal{L}(\boldsymbol{S}, \boldsymbol{y})$, the derivative w.r.t. $\boldsymbol{\theta}$ at the t-th iteration can be written as

$$\begin{aligned} \left.\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{y})}{\partial \boldsymbol{\theta}}\right|_t &= \sum_{i=1}^{m} \sum_{c=1}^{C} \left.\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{y})}{\partial \mathcal{S}_{i,c}}\right|_t \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{\theta}}\right|_t \\ &= \sum_{i=1}^{m} \left(\sum_{c \neq y_i} \left.\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{i,c}}\right|_t \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{\theta}}\right|_t + \sum_{c=y_i} \left.\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{i,c}}\right|_t \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{\theta}}\right|_t \right). \end{aligned} \tag{1}$$

$\left.\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{i,c}}\right|_t$ in Eq. (1) is regarded as a constant scalar in the gradient w.r.t. $\boldsymbol{\theta}$. Since positive pairs are encouraged to be close and negative pairs need to be pushed away from each other, we assume $\left.\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{i,c}}\right|_t \geq 0$ for negative pairs, and $\left.\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{i,c}}\right|_t \leq 0$ for positive pairs. Thus, Eq. (1) can be transformed into a form of weighted sum:

$$\left.\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{y})}{\partial \boldsymbol{\theta}}\right|_t = \sum_{i=1}^{m} \left(\sum_{c \neq y_i} w_{i,c} \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{\theta}}\right|_t - \sum_{c=y_i} w_{i,c} \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{\theta}}\right|_t \right), \tag{2}$$

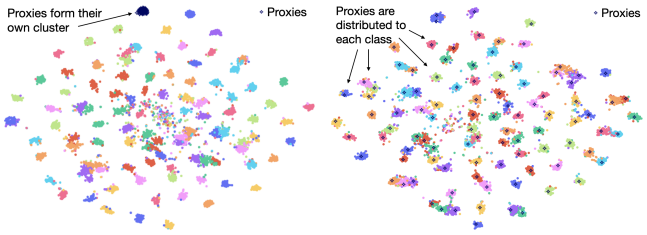
where $w_{i,c} = \left|\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{y})}{\partial \mathcal{S}_{i,c}}\right|_t$. Eq. (2) suggests that the gradient signal is actually controlled by $w_{i,c}$, namely, how the similarity metric is defined in the DML loss function.

3.2 Impact of Gradient Weights on Proxies

Let $\boldsymbol{P} \in \mathbb{R}^{C \times d}$ be the proxy set. The gradient w.r.t. $\boldsymbol{P}$ can be obtained by replacing $\boldsymbol{\theta}$ in Eq. (2):

$$\left.\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{y})}{\partial \boldsymbol{P}}\right|_t = \sum_{i=1}^{m} \left(\sum_{c \neq y_i} w_{i,c} \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{p}_c}\right|_t - \sum_{c=y_i} w_{i,c} \left.\frac{\partial \mathcal{S}_{i,c}}{\partial \boldsymbol{p}_c}\right|_t \right). \tag{3}$$

Eq. (3) suggests that the learning of a proxy is also affected by $w_{i,c}$. In pair-based methods, the effect of $w_{i,c}$ is symmetrical since we can treat either side of the pair as an anchor. However, in proxy-based methods, larger gradient weights may result in an undesirable proxy distribution, which is detrimental to the representation of global information.



(a) Proxy-Anchor@epoch30 **(b) Norm.SoftMax@epoch30**

Figure 2: t-SNE [34] visualization of 512-dimensional embedding space for the Cars-196 dataset. (a) and (b) present the embedding space learned by Proxy-Anchor and normalized SoftMax, respectively. The proxies of Proxy-Anchor, plotted with navy blue stars, form a cluster at an early stage due to the strong repulsive forces.

The first proxy-based loss combined with gradient weighting was the Proxy-Anchor loss [18], which is formulated as follows:

$$\begin{aligned} \mathcal{L}_{PA} := &\frac{1}{|C^+|} \sum_{c \in C^+} \log\left(1 + \sum_{i:y_i=c} e^{\alpha(\delta - \mathcal{S}_{i,c})}\right) \\ &+ \frac{1}{C} \sum_{c=1}^{C} \log\left(1 + \sum_{i:y_i \neq c} e^{\alpha(\delta + \mathcal{S}_{i,c})}\right), \end{aligned} \tag{4}$$

where $C^+ = \{c | c \in \boldsymbol{y}\}$ denotes a set of class labels in a mini-batch, and α, δ are fixed hyper-parameters. By taking the derivative of $\mathcal{L}_{PA}$ w.r.t. $\mathcal{S}_{i,c}$, the gradient weights are given by

$$w_{i,c} = \begin{cases} \frac{1}{|C^+|} \frac{\alpha e^{\alpha(\delta - S_{i,c})}}{1 + \sum_{j:y_j=c} e^{\alpha(\delta - S_{j,c})}}, & y_i = c, \\ \frac{1}{C} \frac{\alpha e^{\alpha(\delta + S_{i,c})}}{1 + \sum_{j:y_j \neq c} e^{\alpha(\delta + S_{j,c})}}, & y_i \neq c. \end{cases} \tag{5}$$

From Eq. (5), it is clear that for a positive pair, $w_{i,c}$ will be large if the data embedding is far from its proxy (*i.e.* $\mathcal{S}_{i,c}$ is small), and will be larger if it is farther than other positive pairs related to the same proxy (*i.e.* $\mathcal{S}_{i,c} < \forall \mathcal{S}_{j,c}$). A similar approach is adopted for the negative pairs, as illustrated in Fig. 1, where the thickness of the arrows indicates the magnitude of the gradient weights. If a positive pair contains an outlier, the proxy will be pulled heavily in that direction.

We now analyze the optimization problem according to each term in the Proxy-Anchor loss.

Proposition 1

$$\begin{aligned} \log\left(1 + \sum_{i:y_i=c} e^{\alpha(\delta - \mathcal{S}_{i,c})}\right) &= \max_{\mathcal{P}_c^+} \alpha \sum_{i:y_i=c} \mathcal{P}_c^+(i)(\delta - \mathcal{S}_{i,c}) + H(\mathcal{P}_c^+), \\ \log\left(1 + \sum_{i:y_i \neq c} e^{\alpha(\delta + \mathcal{S}_{i,c})}\right) &= \max_{\mathcal{P}_c^-} \alpha \sum_{i:y_i \neq c} \mathcal{P}_c^-(i)(\mathcal{S}_{i,c} + \delta) + H(\mathcal{P}_c^-), \end{aligned} \tag{6}$$

where $H(\cdot)$ denotes entropy for regularization, $\mathcal{P}_c^+$ and $\mathcal{P}_c^-$ are the probability distributions over the positives and negatives related to a proxy $\boldsymbol{p}_c$, respectively, and $\mathcal{P}_c^+(i) \in \{\mathcal{P}_c^+ | \mathcal{P}_c^+(\delta) + \sum_{i:y_i=c} \mathcal{P}_c^+(i) =$

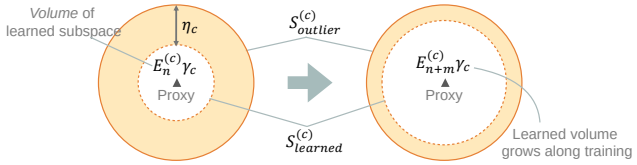


Figure 3: Two thresholds related to a proxy. $\mathcal{S}_{outlier}$: similarity threshold to detect outliers, *i.e.* data embeddings with $\mathcal{S}_{i,c} < \mathcal{S}_{outlier}^{(c)}$; $\mathcal{S}_{learned}$: similarity threshold to detect overlapping between the learned subspace and the newly sampled data. Samples that satisfy $\mathcal{S}_{i,c} > \mathcal{S}_{learned}^{(c)}$ are considered too easy for the model. When $E_n^{(c)}$ increases, η_c should decrease.

$1, \forall i, \mathcal{P}_c^+(i) \geq 0\}, \mathcal{P}_c^-(i) \in \{\mathcal{P}_c^- | \mathcal{P}_c^-(-\delta) + \sum_{i:y_i \neq c} \mathcal{P}_c^-(i) = 1, \forall i, \mathcal{P}_c^-(i) \geq 0\}$, *where we define*

$$\mathcal{P}_c^+(\delta) = \frac{1}{1 + \sum\limits_{j:y_j=c} e^{\alpha(\delta - \mathcal{S}_{j,c})}}, \quad \mathcal{P}_c^-(-\delta) = \frac{1}{1 + \sum\limits_{j:y_j \neq c} e^{\alpha(\mathcal{S}_{j,c} - (-\delta))}} \tag{7}$$

(see appendix for proof).

Proposition 1 suggests that Proxy-Anchor loss actually optimizes two data distributions related to each proxy, and that these distributions are independent to each other. Thus, the global proxy distribution is not relevant during training. This is in contrast to the normalized SoftMax loss, which considers an optimal global proxy distribution related to each data embedding [29], resulting in updates of proxies that are completely dominated by data embeddings. Since more samples are regarded as hard in the early stages of learning, the repulsive force generated by negatives cause the proxies to be far from the ideal distribution, as illustrated in Fig. 2. Additionally, solutions for the positive terms are sub-optimal when the positives contain outliers.

3.3 Class Hardness

Following the idea of active learning, we can reduce the weights of outliers to stabilize the learning of the proxy if most of the data embeddings are close enough to the proxies (*i.e.* easy classes). The problem is then how to obtain a boundary to detect outliers for each class. We define such an outlier boundary as a class-(proxy-) dependent similarity threshold $\mathcal{S}_{outlier}^{(c)}$, and define η_c to be the range for informative samples. In other words, we increase the gradient weights for positive pairs such that $\mathcal{S}_{i,c} \in [\mathcal{S}_{outlier}^{(c)}, \mathcal{S}_{outlier}^{(c)} + \eta_c]$ and decrease them otherwise, where $\mathcal{S}_{outlier}^{(c)} + \eta_c$ should be another threshold related to the current learned subspace for class c.

Since we do not have any *a priori* knowledge about the difficulty of each class, all samples should be treated equally in the early stages of training. Such distinctions should be made independently after the subspace of the class has reached a certain learning stage.

Volume of the learned subspace To simulate the learning of the subspace, we assume each data embedding owns one *volumetric unit* of the subspace belonging to its class, and apply the theory of *effective number* (discussed in [7]).

Let γ_c be the volumetric unit for class c, we assume it is affected by the class hardness and should be larger for a difficult class. A sample from a harder class contains more information than that from an easier class. This is consistent with the fact that most data embeddings of easier classes are closer to each other than that of harder classes. According to the original definition in [7], the total volume of the feature space for each class has an upper bound, and random sampling can be considered as randomly covering this volume. This means that overlapping between the samples randomly happens. Let $V\gamma_c$ be the upper bound of the total volume and $\beta = \frac{V-1}{V}$. Then the effective number (denoted as E_n) is defined as

$$E_n^{(c)}\gamma_c = \frac{1-\beta^n}{1-\beta}\gamma_c, \tag{8}$$

where

$$\lim_{n\to\infty} E_n^{(c)} = \frac{1}{1-\beta} = V \tag{9}$$

holds. Eqs. (8) and (9) suggest that E_n is independent from the class hardness. Therefore, in the proposed approach, we regard the effective number as training *progress bar* and introduce an effective number for each class. The upper bound V is set as a hyper-parameter. Furthermore, we treat the boundary of the feature space as the outlier boundary, and define $\mathcal{S}_{outlier}^{(c)} + \eta_c$ as the threshold of the learned subspace related to class c, denoted as $\mathcal{S}_{learned}^{(c)}$. Hence, η_c decreases as the learning progresses, as shown in Fig. 3.

Estimating the Class Hardness Based on the above definition, for a proxy-based method, the average cosine similarity of the learned positive proxy-data pairs (excluding the outliers) reflects how well the model learned about a class, and can be used to estimate the class hardness after a certain stage.

Estimating the class hardness using only the current mini-batch may not be enough, since there are usually only a few samples from the same class in the mini-batch, and the number of samples for some classes may be zero. Therefore, we apply a memory queue to store the clean data embeddings which were sampled during the past few iterations. Formally, let M be the memory queue with max size T, and T_c be the total number of embeddings from class c in M. Then $\mathcal{S}_{learned}^{(c)}$ is calculated by

$$\mathcal{S}_{learned}^{(c)} = h \cdot \mathcal{S}_{avg}^{(c)} = \frac{h}{T_c} \sum_{i:(\boldsymbol{x}_i, y_i) \in M, y_i = c} \mathcal{S}_{i,c}, \tag{10}$$

where h is the hyper-parameter used to scale the hardness. After processing a mini-batch, we filter out outliers and enqueue the rest to M, as illustrated in Fig. 4. The memory queue starts after a certain number of steps is reached, because the embeddings in the initial stages are typically scattered and the resulting similarities are not representative.

3.4 Informative Sample-Aware Proxy

Our proposed Proxy-ISA utilizes the aforementioned properties to generate dynamic pair-weights. Let $\omega_{i,c}$ be the weighting score for the pair which consists of a data embedding $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ and a proxy $\boldsymbol{p}_c$. This weighting score depends on the semantic state of $\boldsymbol{x}_i$ and learning progress $E_n^{(c)}$. Let $\omega_{i,c}^+$ and $\omega_{i,c}^-$ denote the weighting score for positive pairs and negative pairs, respectively.

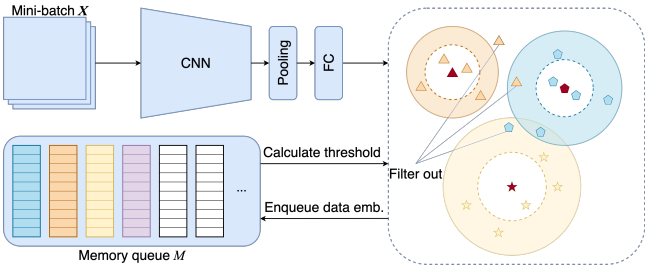


Figure 4: Training flow of Proxy-ISA. Shapes denote embeddings from different classes, whose thresholds are denoted in different colors. When the size of M reaches T, it dequeues the old embeddings to keep the memory up-to-date. Embeddings from the past few iterations are available for reference due to the *slow drift* phenomenon discussed in [39]. The algorithm is described in appendix.

To reduce the harmful factors from extremely hard samples, we first reform the optimization problems as

$$\begin{aligned}\max_{Q_c^+} \alpha \sum_{i:y_i=c} \omega_{i,c}^+ Q_c^+(i)(\delta - \mathcal{S}_{i,c}) + H(Q_c^+)\\ \max_{Q_c^-} \alpha \sum_{i:y_i\neq c} \omega_{i,c}^- Q_c^-(i)(\mathcal{S}_{i,c} + \delta) + H(Q_c^-)\end{aligned} \tag{11}$$

where Q_c^+ and Q_c^- have the same form as $\mathcal{P}_c^+$ and $\mathcal{P}_c^-$, respectively. According to Proposition 1, the optimal of the formulas in Eq. (11) are

$$\log\left(1 + \sum_{i:y_i=c} e^{\omega_{i,c}^+ \cdot \alpha(\delta - \mathcal{S}_{i,c})}\right), \quad \log\left(1 + \sum_{i:y_i\neq c} e^{\omega_{i,c}^- \cdot \alpha(\delta + \mathcal{S}_{i,c})}\right), \tag{12}$$

respectively. Thus, the objective function of Proxy-ISA can be formulated as

$$\begin{aligned}\mathcal{L}_{ISA} := &\frac{1}{\sum_{c\in C^+} \bar{\omega}_c^+} \sum_{c\in C^+} \log\left(1 + \sum_{i:y_i=c} e^{\omega_{i,c}^+ \cdot \alpha(\delta - \mathcal{S}_{i,c})}\right)\\ &+ \frac{1}{\sum_c^C \bar{\omega}_c^-} \sum_{c=1}^{C} \log\left(1 + \sum_{i:y_i\neq c} e^{\omega_{i,c}^- \cdot \alpha(\delta + \mathcal{S}_{i,c})}\right),\end{aligned} \tag{13}$$

where $\bar{\omega}_c^+$ and $\bar{\omega}_c^-$ are the average of all $\omega_{i,c}^+$ and $\omega_{i,c}^-$, respectively, for all samples i that belong to class c. The summation reflects the average scores of all classes (the global learning status) that relate to the positive or negative term.

Adaptive Pair Weighting Scores Since the model has higher confidence in the intra-class hardness when the learning progresses to a later stage, the penalty from $\omega_{i,c}$ is greater for a larger $E_n^{(c)}$. A naive implementation is to set $\omega_{i,c} = \frac{1}{E_n^{(c)}}$ for both positive and negative pairs when penalty is needed. However, this causes the learning signal to be too weak for positive pairs and thus an imbalance between the positive and negative terms when $E_n^{(c)}$ is large, since the penalties for outliers are only made for positive pairs. To prevent the diminishing of $\omega_{i,c}^+$, we set a lower bound $v_n^{(c)}$

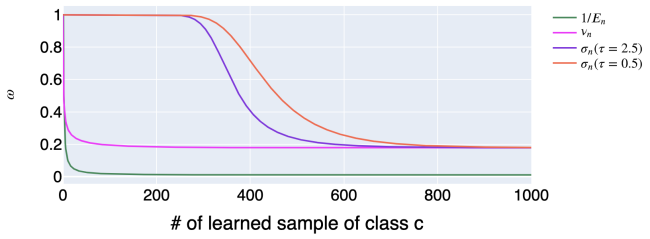


Figure 5: Curve of the decay functions.

for all positive pairs, defined as

$$v_n^{(c)} = \frac{1}{1 + \log(1 + E_n^{(c)})}, \tag{14}$$

Based on the analysis in Sec. 3.3, we define η_c as follows:

$$\eta_c = (1 + k(1 - h \cdot \mathcal{S}_{avg}^{(c)}))v_n^{(c)} + \lambda, \tag{15}$$

where k (> 0) is the sensitivity factor and $\lambda \in [0, 1]$ is the margin. Thus, the searching length η_c is controlled by class hardness, where a harder class has a wider searching space, and an easier class is restricted to help stabilize the learning.

Finally, the proposed dynamic weights are defined as follows:

$$\begin{aligned}\omega_{i,c}^+ &= \begin{cases} 1 + \sigma_n^{(c)}, & h \cdot \mathcal{S}_{avg}^{(c)} - \eta_c \le \mathcal{S}_{i,c} \le h \cdot \mathcal{S}_{avg}^{(c)},\\ \sigma_n^{(c)}, & \text{otherwise,}\end{cases}\\ \omega_{i,c}^- &= \begin{cases} \frac{1}{\max(1, E_n^{(c)})}, & \mathcal{S}_{i,c} < h \cdot \mathcal{S}_{avg}^{(c)} - \eta_c,\\ 1, & \text{otherwise,}\end{cases}\end{aligned} \tag{16}$$

where $\sigma_n^{(c)}$ is a decay function for $\omega_{i,c}^+$, defined as

$$\sigma_n^{(c)} = 1 + \frac{(1 + e^{-\tau})(v_n^{(c)} - 1)}{1 + e^{V - E_n^{(c)} - \tau}}, \tag{17}$$

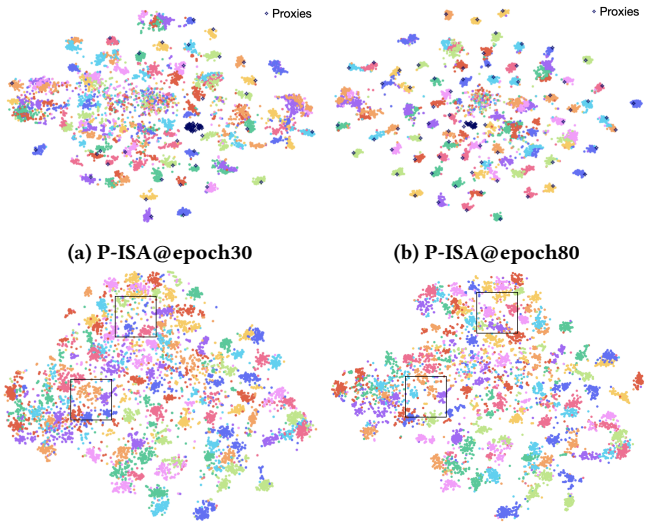
and τ (> 0) is a hyper-parameter that controls the timing of decay. Since the embeddings are more likely to change in the early stages and gradually stabilize later, the dynamic weights need to be controlled such that they change less in the early stages (see appendix for derivation of $\sigma_n^{(c)}$). Using Eq. (9) and Eq. (14), $\lim_{n\to\infty} \sigma_n^{(c)} = \lim_{n\to\infty} v_n^{(c)} = \frac{1}{1+\log(1+V)}$. This ensures that $\omega_{i,c}^+$ does not fall below $v_n^{(c)}$, as shown in Fig. 5.

4 EXPERIMENTS

We conduct experiments on four widely used benchmarks to evaluate and analyze the effectiveness of Proxy-ISA.

4.1 Experimental Setting

Datasets Experiments were conducted on the CUB-200-2011 [35], Cars-196 [21], Stanford Online Products (SOP) [33], and In-Shop Clothes Retrieval (In-Shop) [24] datasets. We follow the standard protocol applied in [33] to split them into training and testing parts. CUB-200-2011 contains 200 species of birds with 11,788 images; we used the first 100 classes (5,864 images) for training and the rest for testing. Cars-196 consists of 196 model categories of cars with 16,185 images; the first 98 classes (8,054 images) were used



(a) P-ISA@epoch30 (b) P-ISA@epoch80

(c) Test set, PA (d) Test set, P-ISA

Figure 6: t-SNE visualization of the 512-dimensional embedding space for the Cars-196 dataset. (a) and (b) present the changes of the embedding space learned by Proxy-ISA during training at epoch 30 and 80, respectively, and (c) and (d) show the results of PA and P-ISA on the test set, respectively.

for training and the rest are used for testing. SOP contains 22,634 classes with 120,053 product images; we used the first 11,318 classes (59,551 images) for training and the rest for testing. The first 3,997 classes (25,882 images) of In-Shop were used for training, while the remaining 3,985 classes were used for the test set, which was partitioned into a query set and a gallery set containing 14,218 and 12,612 images, respectively.

Evaluation Metrics The evaluation procedure included two types of metrics. We first performed a standard evaluation following [33], calculating the Recall@K for image retrieval tasks. We also adopted MAP@R, known as *Mean Average Precision*, from the MLRC [27] evaluation settings. MAP@R considers the correctness of each result associated with a single query, and is considered to be more suitable for evaluating the entire embedding space.

Implementation Details Our method was implemented in PyTorch [28].We used the Inception network with batch normalization [14] as the embedding network, the embedding dimension was set to 512. All input images were cropped to 224×224, and random cropping and horizontal flipping were applied for augmentation during training; only center-cropping was used during testing. The memory queue was turned on at the second epoch and the filter was enabled one epoch after that. The hyper-parameters induced by Proxy-Anchor were set to $\alpha = 32$, $\delta = 10^{-1}$ following [18] for all experiments. Unless otherwise mentioned, we adopted a batch size of 128, and used the Adam optimizer [19] with a learning rate of 10^{-4} and hyper-parameters $V = 100$, $h = 1.5 \times 10^{-1}$, $k = 9 \times 10^{-1}$, $\lambda = 10^{-1}$, $\tau = 1.5$ as our default setting.

Table 1: Comparison of Recall@1 of proxy-based methods on four famous datasets for the task of image retrieval. † denotes evaluation in a fair setting

Method	CUB	Cars	SOP	In-Shop
Proxy-NCA [26]	65.0	83.2	75.9	86.9
SoftTriplet [29]	65.4	84.5	78.3	–
Proxy-Anchor† [18]	66.6	84.0	78.3	91.3
ProxyGML [45]	66.6	85.5	78.0	–
Proxy-ISA (ours)	**68.1**	**86.4**	**78.9**	**92.3**

Table 2: Evaluation in MLRC setting. † denotes evaluation in a fair setting

Method	CUB-200-2011		Cars-196	
	R@1	MAP@R	R@1	MAP@R
Margin [41]	63.6 ± 0.5	23.1 ± 0.3	81.2 ± 0.5	24.2 ± 0.3
Proxy-NCA [26]	65.0 ± 0.4	23.9 ± 0.3	83.6 ± 0.3	25.4 ± 0.3
CosFace [36]	67.3 ± 0.3	26.7 ± 0.2	85.5 ± 0.2	27.6 ± 0.3
ArcFace [8]	67.5 ± 0.3	26.5 ± 0.2	85.4 ± 0.3	27.2 ± 0.3
MS [37]	65.0 ± 0.3	24.7 ± 0.1	85.1 ± 0.3	28.1 ± 0.2
MS+Miner [37]	67.7 ± 0.2	25.2 ± 0.2	83.7 ± 0.3	27.0 ± 0.4
SoftTriplet [29]	66.2 ± 0.4	25.6 ± 0.2	84.5 ± 0.3	27.1 ± 0.2
Proxy-Anchor† [18]	66.3 ± 0.3	25.7 ± 0.3	83.6 ± 0.4	27.1 ± 0.3
Proxy-ISA (ours)	**68.1 ± 0.3**	**26.8 ± 0.2**	**86.3 ± 0.2**	**29.3 ± 0.2**

4.2 Embedding Space Visualization

To outline how Proxy-ISA controls the learning of proxies, we visualize the embedding space via t-SNE [34]. As illustrated in Fig. 6a, Proxy-ISA gradually widens the distance between proxies so that each proxy is distributed in its own subspace, and easier classes form clusters earlier than harder classes. This demonstrates that the class hardness differs, and the definition of informative samples for each class also differs. The relationships of class hardness learned by Proxy-ISA reserves a wider range for a harder class even if it is an unseen category, as illustrated in Fig. 6d. This demonstrates that Proxy-ISA can better utilize the embedding space by learning dynamic margins according to the class hardness. As a result, the model trained with Proxy-ISA acquires a more detailed subspace discriminative capability and thus a better generalization.

4.3 Comparison with State-of-the-Art

We now compare Proxy-ISA with state-of-the-art DML methods by performing image retrieval tasks. The learning rates were set to 6×10^{-4} for SOP and In-Shop. We only adopted the warm-up epoch and the AdamW optimizer [25] in the exceptional settings[1] introduced by [18] for these two datasets. For a fair comparison, all such settings were also removed for Proxy-Anchor in experiments on CUB-200-2011 and Cars-196. As shown in Table 1, the proposed Proxy-ISA achieves the best performance compared with the existing proxy-based methods.

[1] See appendix for details.

The evaluation results in the MLRC setting are presented in Table 2. As the MAP@R metric evaluates the whole embedding space, the results on the two widely used datasets demonstrate the effectiveness of Proxy-ISA in improving the quality of the embedding space.

5 CONCLUSION

Treating the proxy-data pairs equally to data pairs can cause undesirable proxy distribution in the embedding space. To address this problem, we proposed the Informative Sample-Aware Proxy (Proxy-ISA), which controls gradient weights non-uniformly according to different semantic states. Based on the adaptive semantic states for different classes, Proxy-ISA helps each proxy discover the most informative data by exploiting learned information. Furthermore, we showed that the definition of "informative" is class hardness dependent by applying the concept of "*volume*" of the learned subspace to DML. The empirical results demonstrate the superiority of Proxy-ISA over state-of-the-arts.

ACKNOWLEDGMENTS

This work is supported by DENSO IT LAB Recognition and Learning Algorithm Collaborative Research Chair (Tokyo Tech.).

REFERENCES

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron C. Courville, and Yoshua Bengio. 2015. Variance Reduction in SGD by Distributed Importance Sampling. *CoRR* (2015). arXiv:1511.06481
[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.
[3] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[4] Chris Careaga, Brian Hutchinson, Nathan Oken Hodas, and Lawrence Phillips. 2019. Metric-Based Few-Shot Learning for Video Action Recognition. *CoRR* (2019). arXiv:1909.09602
[5] Haw-Shiuan Chang, Erik G. Learned-Miller, and Andrew McCallum. 2017. Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples. In *NeurIPS*.
[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*.
[7] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-Balanced Loss Based on Effective Number of Samples. In *CVPR*.
[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *CVPR*.
[9] Yueqi Duan, Lei Chen, Jiwen Lu, and Jie Zhou. 2019. Deep Embedding Learning With Discriminative Sampling Policy. In *CVPR*.
[10] Soumyadeep Ghosh, Richa Singh, and Mayank Vatsa. 2019. On Learning Density Aware Embeddings. In *CVPR*.
[11] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. 2005. Neighbourhood Components Analysis. In *NeurIPS*.
[12] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In Defense of the Triplet Loss for Person Re-Identification. *CoRR* (2017). arXiv:1703.07737
[13] Elad Hoffer and Nir Ailon. 2015. Deep Metric Learning Using Triplet Network. In *Similarity-Based Pattern Recognition*.
[14] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*.
[15] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *ICML*.
[16] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogério Schmidt Feris, Raja Giryes, and Alexander M. Bronstein. 2019. RepMet: Representative-Based Metric Learning for Classification and Few-Shot Object Detection. In *CVPR*.
[17] Angelos Katharopoulos and Francois Fleuret. 2018. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *ICML*.
[18] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. 2020. Proxy Anchor Loss for Deep Metric Learning. In *CVPR*.
[19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[20] Byungsoo Ko, Geonmo Gu, and Han-Gyu Kim. 2021. Learning with Memory-based Virtual Classes for Deep Metric Learning. In *ICCV*.
[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D Object Representations for Fine-Grained Categorization. In *ICCV*.
[22] M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *NeurIPS*.
[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. In *ICCV*.
[24] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval With Rich Annotations. In *CVPR*.
[25] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR*.
[26] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. 2017. No Fuss Distance Metric Learning Using Proxies. In *ICCV*.
[27] Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim. 2020. A Metric Learning Reality Check. *CoRR* (2020). arXiv:2003.08505
[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.
[29] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. 2019. SoftTriple Loss: Deep Metric Learning Without Triplet Sampling. In *ICCV*.
[30] Jérôme Revaud, John Almazán, Rafael S. Rezende, and César Roberto de Souza. 2019. Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In *ICCV*.
[31] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*.
[32] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *NeurIPS*.
[33] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep Metric Learning via Lifted Structured Feature Embedding. In *CVPR*.
[34] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* (2008).
[35] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J Belongie. 2011. *The Caltech-UCSD Birds-200-2011 Dataset*. Technical Report CNS-TR-2011-001. California Institute of Technology.
[36] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. CosFace: Large Margin Cosine Loss for Deep Face Recognition. In *CVPR*.
[37] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. 2019. Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning. In *CVPR*.
[38] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, and Neil M. Robertson. 2019. Deep Metric Learning by Online Soft Mining and Class-Aware Attention. In *AAAI*. Article 657.
[39] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. 2020. Cross-Batch Memory for Embedding Learning. In *CVPR*.
[40] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A Discriminative Feature Learning Approach for Deep Face Recognition. In *ECCV*.
[41] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2017. Sampling Matters in Deep Embedding Learning. In *ICCV*.
[42] Wanyin Wu, Dapeng Tao, Hao Li, Zhao Yang, and Jun Cheng. 2021. Deep features for person re-identification on metric learning. *Pattern Recognition* (2021).
[43] Yang Zhao, Chunyuan Li, Ping Yu, and Changyou Chen. 2021. ReMP: Rectified Metric Propagation for Few-Shot Learning. In *CVPR*.
[44] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. 2019. Hardness-Aware Deep Metric Learning. In *CVPR*.
[45] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. 2020. Fewer is More: A Deep Graph Metric Learning Perspective Using Fewer Proxies. In *NeurIPS*.

A ALGORITHM AND TRAINING COMPLEXITY

We describe the training flow of our proposed method in Algorithm 1.

Training Complexity Although our proposed Proxy-ISA takes complexity of $O(NC)$, it will increase the training time to some extent on large datasets that contain a larger number of categories (*i.e.*, $C \gg N$), since each dynamic weight is treated independently and calculation of each η_c takes extra complexity.

Algorithm 1 A training iteration of Proxy-ISA

Input: data embeddings $\boldsymbol{X}$, class label $\boldsymbol{y}$, proxy set $\boldsymbol{P}$, memory queue M.
$S \leftarrow$ cosine similarity between $\boldsymbol{P}$ and $\boldsymbol{X}$
$C^+ \leftarrow$ class labels appeared in $\boldsymbol{y}$
$filter \leftarrow \emptyset$
$\eta_1, \eta_2, \ldots, \eta_C \leftarrow$ calculate search length for each class
for $c \in C^+$ **do**
  for $i : y_i = c$ **do**
    if $filter$ is turned on **then**
      Calculate $\omega^+_{i,c}$ according to $\mathcal{S}_{i,c}$
      Add $\boldsymbol{x}_i$ to $filter$ if $\mathcal{S}_{i,c} < \mathcal{S}^{(c)}_{learned} - \eta_c$
    else
      $\omega^+_{i,c} \leftarrow 1$
    end if
  end for
end for
for $c = 1, 2, \ldots, C$ **do**
  for $i : y_i \neq c$ **do**
    Calculate ω^-_{ic} according to $\mathcal{S}_{i,c}$
  end for
end for
Calculate loss
if M is turned on **then**
  Enqueue $\boldsymbol{x}_i \in \{\boldsymbol{X} - filter\}$ to M
  Dequeue old embeddings if M out of size
  for $c \in C^+$ **do**
    Update $\mathcal{S}^{(c)}_{learned}$ through M
  end for
end if

B PROOF OF PROPOSITION 1

Proof. According to the K.K.T. conditions [3], $\mathcal{P}^+_c$ in Eq. (6) has the closed-form solution

$$\mathcal{P}^+_c(i) = \frac{e^{\alpha(\delta - \mathcal{S}_{i,c})}}{1 + \sum\limits_{j:y_j=c} e^{\alpha(\delta - \mathcal{S}_{j,c})}}. \tag{18}$$

Therefore, we have

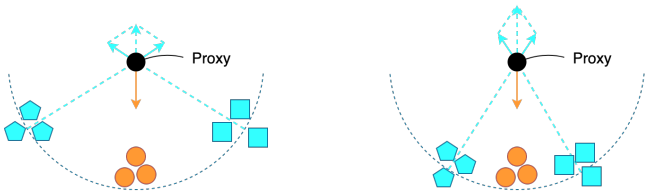
$$\alpha \sum_{i:y_i=c} \mathcal{P}^+_c(i)(\delta - \mathcal{S}_{i,c}) + H(\mathcal{P}^+_c) = \log\left(1 + \sum_{i:y_i=c} e^{\alpha(\delta - \mathcal{S}_{i,c})}\right). \tag{19}$$

The same analysis is applicable to $\mathcal{P}^-_c$.

C DETAILS OF THE DECAY FUNCTION

Since the penalty from $\omega^+_{i,c}$ is decided by the estimated class hardness, it should not be too large when the model cannot determine the class hardness. In other words, all $\omega^+_{i,c}$ should be close to 1, with less variation in the early learning phase. Given an original sigmoid function that varies with $E^{(c)}_n$:

$$\frac{1}{1 + e^{-E^{(c)}_n}}, \tag{20}$$



(a) A loose space **(b) A compact space**

Figure 7: Impact of compactness of the embedding space on the repulsive force. A compact space (or clusters of the harder classes) generates greater joint repulsive force, even if the cosine similarity to the proxy (*i.e.* the gradient weight of the proxy-data pairs) is kept unchanged.

we then adjust it to our desired curve. Firstly, to satisfy $\lim_{n\to 0+} \sigma^{(c)}_n \approx 1$, we transform the function to

$$1 - \frac{1}{1 + e^{-(E^{(c)}_n - V) - \tau}}, \tag{21}$$

where we introduce τ to control the timing of decay. Secondly, to satisfy $\lim_{n\to\infty} \sigma^{(c)}_n = \nu^{(c)}_n$, we formulate an equation with unknown variable ξ as follows:

$$\lim_{n\to\infty} \sigma^{(c)}_n = 1 - \frac{\xi}{1 + e^{-\tau}} = \nu^{(c)}_n. \tag{22}$$

The solution is

$$\xi = (1 + e^{-\tau})(1 - \nu^{(c)}_n), \tag{23}$$

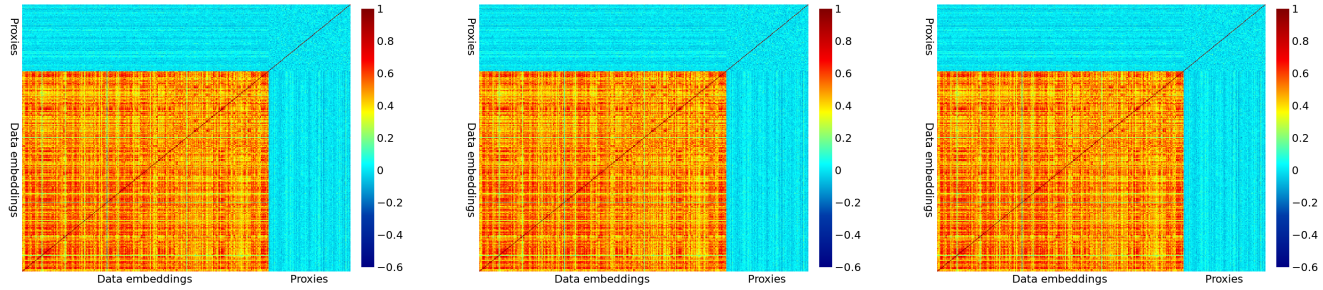
thus, the decay function is formulated as:

$$\sigma^{(c)}_n = 1 + \frac{(1 + e^{-\tau})(\nu^{(c)}_n - 1)}{1 + e^{V - E^{(c)}_n - \tau}}. \tag{24}$$

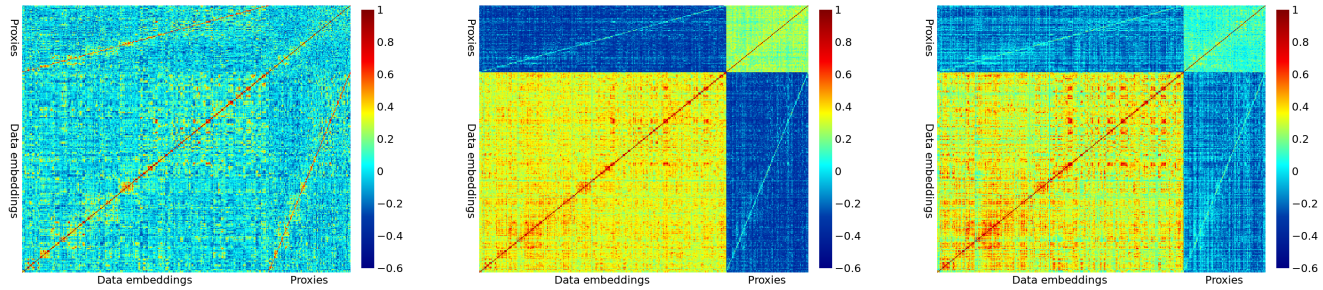
The penalty from $\omega^-_{i,c}$ is set to $1/E^{(c)}_n$ $(< \nu^{(c)}_n)$ for preventing the repulsive force on the proxy from being too large under the impact of a large population of easy negatives.

D IMPACT OF PROXY-ISA ON THE EMBEDDING SPACE

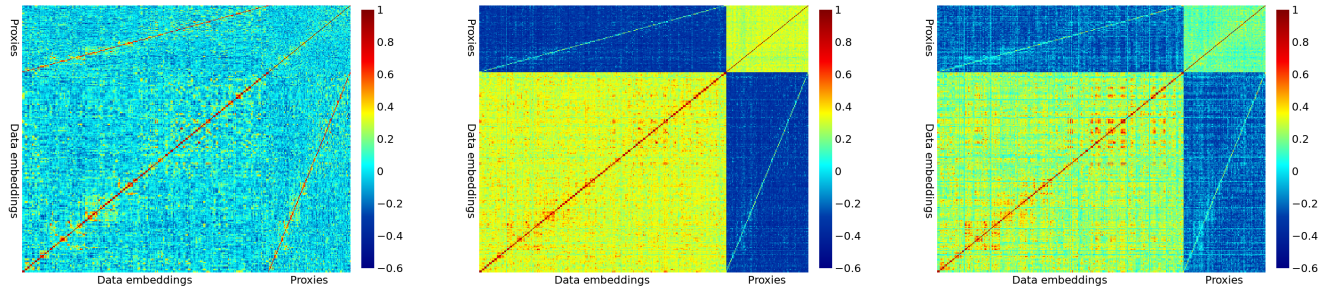
To quantitatively describe the learned embedding space, we show its compactness via heat map of pair-wise cosine similarity. As illustrated in Fig. 8, the distribution of the data embeddings is relatively compact at the initial stage (results of the pre-trained backbone and a randomly initialized embedding layer), and is dispersed to varying degrees after applying different objective functions. Fig. 8e, Fig. 8h and Fig. 8k show that the embedding space learned by Proxy-Anchor loss [18] stops dispersing after it reaches a certain stage, compared to the case of normalized SoftMax loss in Fig. 8d, Fig. 8g and Fig. 8j. This is resulted from the key difference between Proxy-Anchor and normalized SoftMax, the consideration of the optimization problem (*i.e.*, Proxy-Anchor only considers the optimal distribution of data embeddings related to each proxy, while normalized SoftMax only considers the optimal distribution of proxies related to each data embedding), since Proxy-Anchor generates stronger attractive and repulsive force between a proxy and a relatively hard sample, the proxy distribution is easily disrupted.



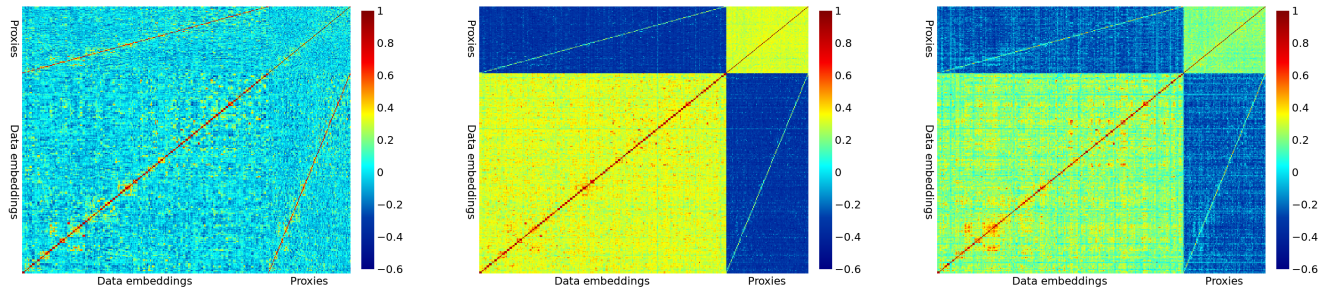
(a) N.SoftMax@Initial (b) Proxy-Anchor@Initial (c) Proxy-ISA@Initial



(d) N.SoftMax@eph10 (e) Proxy-Anchor@eph10 (f) Proxy-ISA@eph10



(g) N.SoftMax@eph30 (h) Proxy-Anchor@eph30 (i) Proxy-ISA@eph30



(j) N.SoftMax@eph60 (k) Proxy-Anchor@eph60 (l) Proxy-ISA@eph60

Figure 8: Heat map of cosine similarity between data embeddings of training set of the Cars-196 dataset and proxies. Three data samples are randomly selected for each class.

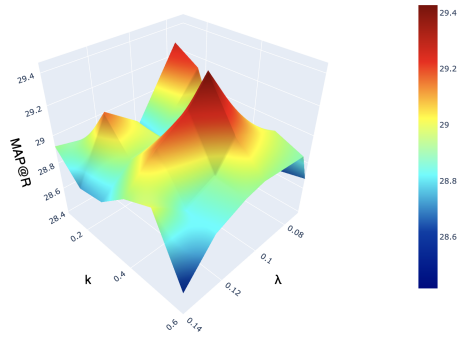


Figure 9: Impact of the hyper-parameters k and λ on the MAP@R metric.

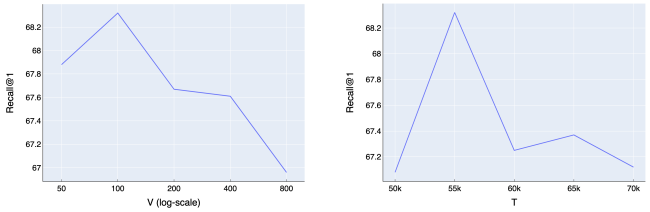
Since most of the data embeddings are scattered in a relatively compact space at the initial stage, a larger fraction of this is regarded as hard samples (mostly negative) by Proxy-Anchor, which means the proxies are more likely to move in undesired directions, thus forming a cluster far from the data embeddings, as shown in the paper. After the proxies reach a balance (*i.e.*, learned a shortcut by discriminating the data embeddings while getting far from them), they still keep the ability to force the positives to be close to each other and to be far from the negatives, but failed to disperse the whole embedding space further. In the meanwhile, a too compact embedding space generates stronger repulsive force, which blocks the proxy from getting closer to its ideal cluster, as described in Fig. 7.

Unlike Proxy-Anchor, our proposed Proxy-ISA captures informative samples that correspond to different learning stages, thus continues to disperse the embedding space, even in the late learning phase, as shown in Fig. 8i and Fig. 8l. Although the dispersion is not as great as normalized SoftMax, Proxy-ISA allows the model to consider relative sample hardness for each class, which is ignored by normalized SoftMax (also ignored by other SoftMax based methods [8, 36] and Proxy-NCA [26]). In other words, Proxy-ISA considers both sample-wise hardness and class-wise hardness, this helps the proxy discover its own informative samples adaptively according to different classes, and thus generates better learning signals for the model.

E ABLATION STUDY

E.1 Impact of Hyper-parameters

The most important hyper-parameters are k and λ, which control the search length (η_c) of the informative samples, and k also controls the sensitivity to class hardness. As illustrated in Fig. 9, for Cars-196 [21] dataset, when k reaches an appropriate range, the adaptive search length helps each proxy focus on its informative samples, and thus improves the model performance for a wider range of the margin λ. In the meanwhile, a search length with too large a k, *i.e.*, too sensitive to the class hardness, will be more likely to introduce outliers for harder classes, thus can degrade the generalization. This is consistent with that hard samples do not always mean informative. For different datasets, the appropriate k should be different because the inter-class relation differs.



(a) Recall@1 results *vs.* V **(b) Recall@1 results *vs.* T**

Figure 10: Impact of the hyper-parameters V (upper bound of E_n) and T (maximum size of the memory queue) on the Recall@1 metric.

Table 3: Comparison of different combination of penalty from $\omega_{i,c}$ on Cars-196 dataset

$\omega^+_{i,c}$:	1 (no penalty)	1 (no penalty)	$1/E_n^{(c)}$	$v_n^{(c)}$	$\sigma_n^{(c)}$
$\omega^-_{i,c}$:	1 (no penalty)	$1/E_n^{(c)}$	$1/E_n^{(c)}$	$1/E_n^{(c)}$	$1/E_n^{(c)}$
Recall@1	83.6	85.1	71.4	85.8	**86.3**
MAP@R	27.1	28.3	16.3	28.7	**29.3**

We show the impact of V and T on the test set of CUB-200-2011 [35] (100 classes). As illustrated in Fig. 10a, too large V will degrade the model performance, since total volume of the finite samples in the feature space of one class has an appropriate limit, which should not be set too large. In the meanwhile, V controls the decay of the dynamic weights, which needs to be at an appropriate timing to help the model focus on the informative samples.

The memory size T affects the estimation quality of $\mathcal{S}^{(c)}_{learned}$, which is the threshold of a learned subspace. Too small T will result in an insufficient number of samples used for estimation, while too large T will result in some of the samples in M being outdated, thus the memory size has a best range, as shown in Fig. 10b.

E.2 Effect of the Decay Function

To show the effect of the decay function $\sigma_n^{(c)}$, we compare the performance by setting different penalties of $\omega^+_{i,c}$. As shown in Table 3, setting $\omega^+_{i,c} = 1/E_n^{(c)}$, which is the same to $\omega^-_{i,c}$, will degrade the performance since the imbalanced weighting between positives and negatives, as discussed in the paper, bounding it with $v_n^{(c)}$ alleviates this problem. After applied the decay function $\sigma_n^{(c)}$ for $\omega^+_{i,c}$, our method provides weighting factors dynamically along learning, and the penalty increases significantly only when the learning of the class reached an appropriate phase, this also helps the model treat informative samples non-uniformly along learning.

In the meanwhile, it prevents the gradient from being dominated by a relatively large number of easy proxy-data pairs by directly reducing their gradient weights, which also keeps the class-related regions from being over-compressed, thus prevents over-fitting, as shown in Fig. 11a. A similar effect can be observed when we apply the adaptive weighting scores directly on Proxy-NCA [26] (Fig. 11b).

(a) (b)

Figure 11: Performance on unseen classes of Cars-196. Although the baselines converge faster, over-fitting happens as learning progresses.

Table 4: Recall@1 compared to Proxy-Anchor with exceptional settings applied

Method	CUB	Cars	SOP	In-Shop
Proxy-Anchor [18]	68.4	86.1	79.1	91.5
Proxy-ISA	**69.4**	**86.8**	**79.3**	**92.5**

E.3 Extra Settings for Fair Comparison

As mentioned in other work [20, 45], Proxy-Anchor loss is actually implemented with three additional tricks: 1) the AdamW [25] optimizer instead of Adam [19], 2) a parameter warm-up strategy for the last FC layer, 3) a combination of average and max pooling following the backbone network. Comparison to Proxy-Anchor with all the exceptional settings enabled is shown in Table 4.