ASAP DML: DEEP METRIC LEARNING WITH ALTERNATING SETS OF ALTERNATING PROXIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep metric learning (DML) aims to minimize empirical expected loss of the pairwise intra-/inter- class proximity violations in the embedding image. We relate DML to feasibility problem of finite chance constraints. We show that minimizer of proxy-based DML satisfies certain chance constraints, and that the worst case generalization performance of the proxy-based methods can be characterized by the radius of the smallest ball around a class proxy to cover the entire domain of the corresponding class samples, suggesting multiple proxies per class helps performance. To provide a scalable algorithm as well as exploiting more proxies, we consider the chance constraints implied by the minimizers of proxy-based DML instances and reformulate DML as finding a feasible point in intersection of such constraints, resulting in a problem to be approximately solved by alternating projections. Simply put, we repeatedly train a regularized proxy-based loss and re-initialize the proxies with the embeddings of the deliberately selected new samples. We apply our method with the well-accepted losses and evaluate on four popular benchmark datasets for image retrieval. Outperforming state-of-the-art, the proposed approach consistently improves the performance of the applied losses.

1 INRODUCTION

Deep metric learning (DML) poses distance metric problem as learning the parameters of an embedding function so that the semantically similar samples are embedded to the small vicinity in the representation space as the dissimilar ones are placed relatively apart in the Euclidean sense. The typical embedding function is implemented as convolutional neural networks (CNN) for visual tasks and the parameters are learned through minimizing the empirical expected loss with possibly deliberately selected mini-batch gradient updates (Roth et al., 2020; Musgrave et al., 2020). The loss terms in the empirical loss penalize violations of the desired intra- and inter-class proximity constrains. Large-scale problems (in terms of $\#classes$) suffer from the noisy estimation of the expected loss with mini-batches (Schroff et al., 2015; Wang et al., 2020; Musgrave et al., 2020). Recently, augmenting the mini-batches with virtual embeddings called *proxies* is shown to better approximate empirical loss in large-scale problems (Wang et al., 2020; Musgrave et al., 2020) owing to pseudo-global consideration of the dataset during loss computation. These advances raise a critical question: *"How does increasing proxies help?"* which is partially addressed empirically with the methods exploiting multiple proxies per class (Qian et al., 2019; Wang et al., 2020; Zhu et al., 2020).

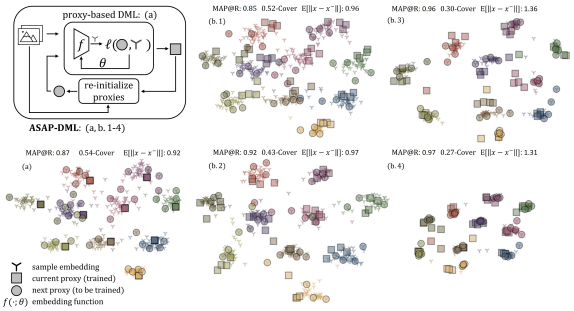


Figure 1: Illustration of our method and the geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-4), where boxes are the converged proxies and the circles are the next proxies as the result of k-Center. In proxy-based DML, proxies are coalesced into one whereas with ASAP, we have diverse proxies, resulting reduced covering radius.

Characterizing generalization performance of proxy-based DML can be a decisive step towards theoretically addressing that question. To this end, we approach DML differently by posing it as a

feasibility problem. In particular, we consider a chance constraint for desired embedding function and relate it to the typical expected loss of DML. Using such a relation, we provide an upper bound to the generalization error of proxy-based DML. Aligned with the literature, the form of the bound suggests possible room for the improvement on the generalization performance if more and diverse proxies are considered per class. However, straightforward increase of the proxies may not help; since, i) proxies of the same class tend to merge (Qian et al., 2019) and ii) memory is prohibitive.

To alleviate these limitations, we relate minimizer of the proxy-based DML to a feasible point of some chance constraints, and reformulate DML as finding a feasible point at the intersection of the sets that the proxies imply. We provide a scalable algorithm using alternating projections to the individual sets to solve the problem. Each projection problem corresponds to a regularized proxy-based DML. Hence, we inherently increase the number of diverse proxies included in the problem. We empirically study the implications of our formulation. Evaluation of our method on image retrieval shows state-of-the-art (SOTA) performance in improving the baselines.

2 Related Work

We discuss the works which are most related to ours. Briefly, our contributions include that i) we reformulate DML as a feasibility problem (*i.e.*, set intersection) to be solved by alternating projections, ii) we expand on the discussions of the works on the generalization bounds to characterize generalization of proxy-based DML, iii) we implicitly utilize arbitrary number of proxies per class.

DML. Most of the existing efforts follow tailoring the loss terms (Roth et al., 2020; Musgrave et al., 2020) to impose the desired intra- and inter-class proximity constraints in the representation space. Mini-batch construction for empirical estimation of the expected loss is tackled with either i) mining of samples in global (Harwood et al., 2017; Ge, 2018; Suh et al., 2019) and local (Schroff et al., 2015; Sohn, 2016; Yuan et al., 2017; Wu et al., 2017; Wang et al., 2019a;b) manner or ii) generating informative samples (Duan et al., 2018; Zhao et al., 2018; Lin et al., 2018; Zheng et al., 2019; Ko & Gu, 2020; Liu et al., 2021). Regularization terms to improve gradient updates towards better generalization are introduced (Zhang et al., 2020; Kim & Park, 2021). More sophisticated tracks to enhance the diversity of the semantic information embedded to the vector representations involve ensemble (Opitz et al., 2017; Kim et al., 2018; Xuan et al., 2018; Sanakoyeu et al., 2019) and discriminative feature learning (Lin et al., 2018; Roth et al., 2019; Do et al., 2019; Jacob et al., 2019; Milbich et al., 2020; Zhao et al., 2021) approaches.

Ranking losses in DML. Typical DML objective enforces distance ranking constraints among the samples in the embedding space via hinge losses penalizing ranking violations. The contrastive (Hadsell et al., 2006), triplet (Weinberger et al., 2006; Schroff et al., 2015), and generalized contrastive with margin (Wu et al., 2017) losses are the simplest forms of the pairwise distance ranking based losses. Proceeding approaches utilize smoothed versions of these losses by replacing hinge loss with log-sum-exp (Wang et al., 2019b) or soft-max (Sohn, 2016; Yu & Tao, 2019) expressions. Similarly, mining for tuples (Schroff et al., 2015; Sohn, 2016; Wu et al., 2017) and ranking among more samples via soft-batch-mining (Oh Song et al., 2016; Sohn, 2016; Wang et al., 2019a; Yu & Tao, 2019; Kim et al., 2020) are addressed to exploit in-batch tuples with non-trivial settings. In our work, we do not consider crafting a loss term. We rather relate minimizing the ranking losses to feasibility of some chance constraints on the probability of observing pairs violating desired intra-/inter-class distances.

Proxy-based DML. Proxy-based methods consider augmenting the mini-batch with more samples for less noisy estimate of the expected loss and circumvent the costly embedding computation to include more samples in the mini-batches. Proxies typically are vectors representing embeddings of the class centers (Movshovitz-Attias et al., 2017; Chen et al., 2018; Teh et al., 2020; Kim et al., 2020) and are trained along with the embedding function parameters. Non-trainable proxies are also exploited in (Wang et al., 2020) to gradually augment mini-batch with previously computed embeddings. In proxy-based DML, the pairwise distances are computed between the proxies and the mini-batch samples. Thus, pseudo-global information of the dataset is considered during loss computation. To better represent global geometry, multiple proxies per class are considered in (Rippel et al., 2016; Qian et al., 2019; Zhu et al., 2020) where the former two build on improving P@1[1] by fine-grained clustering of class samples to overlook intra-class variances. In our analysis, we also

[1] P@1 (immediate neighbourhood) and MAP@R (global geometry) are explained in Appendix-A.2.

align with increasing the proxies. Our work differs in that i) we build on reducing the probability of proximity violations (*i.e.*, improving MAP@R [1]) and ii) we progressively increase the proxies by relating the proxy-based DML instances.

Fairness in evaluation. Independent works (Roth et al., 2020; Musgrave et al., 2020; Fehervari et al., 2019) reveal that conventional training and evaluation procedures in DML may fail to properly assess the true order of performance that the methods bring. The consensus for unbiased comparability is evaluation of the methods with their best version under the same experimental settings unless the compared methods demand any particular architecture or experimental setup. Our experimental setting is completely aligned with the literature's claims for unbiased evaluation of our method.

Characterizing generalization bounds. Notion of robustness in learning algorihms is studied in (Xu & Mannor, 2012) and generalization error bounds of several techniques are derived accordingly. This study is extended to metric learning setting in (Bellet & Habrard, 2015). These works study the deviation between the expected loss and the empricial loss over the whole dataset. Differently in (Sener & Savarese, 2018), deviation between two empirical losses, *core-set loss*, is studied and generalization error is characterized by the core-set loss to formulate an optimization objective for their active learning method. Core-set loss typically measures how much is lost when a subset of the training data is exploited. Generalization bound for metric learning is further studied in (Dong et al., 2020) to analyze and suggest training strategies. Our work expands on the theories in the aforementioned works to characterize and improve generalization bound for proxy-based DML.

3 Notation and Problem Definition

In typical DML, we consider the set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ with elements $z = (x \in \mathcal{X}, y \in \mathcal{Y})$ where $\mathcal{X}$ is a compact space and $\mathcal{Y} = \{1, \dots, C\}$ is a finite label set. We will use $x(y)$ to denote $x(y)$ component of z. We have $p_{\mathcal{Z}}$, an unknown probability distribution over $\mathcal{Z}$. Indicator of the two samples, z and z', belonging to the same class is denoted as $\iota_{y,y'} \in \{\text{-}1, 1\}$ where $\iota_{y,y'} = 1$ if $y = y'$.

We are interested in finding the parameters, θ, of an embedding function, $f(\cdot;\theta) : \mathcal{X} \to \mathbb{R}^D$, so that the parametric distance, $d_f(x, x';\theta) \coloneqq \|f(x;\theta) - f(x';\theta)\|_2$, between two samples, $(x, x') \sim p_{\mathcal{X}}$, reflects their semantic dissimilarity. For any pair, $(z, z') \sim p_{\mathcal{Z}}$ and embedding function, $f(\cdot;\theta)$, we associate a loss, $\ell(z, z';\theta)$, penalizing proximity violations in the embedding image. We omit f dependency in the ℓ notation for simplicity. We are to consider minimization of the expected loss:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{z,z' \sim p_{\mathcal{Z}}}[\ell(z, z';\theta)] \tag{3.1}$$

In practice, we are given a dataset of n instances sampled *i.i.d.* from $\mathcal{Z}$ as $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$ where $[n] = \{1, \dots, n\}$, and an algorithm, $\mathcal{A}_{s_1 \times s_2}$, which outputs parameters, θ, minimizing empirical expected loss with a training error, $e(\mathcal{A}_{s_1 \times s_2})$, for a given set of pairs, $\{(z_i, z_j)\}_{i,j \in s_1 \times s_2}$, from the dataset, where $s_k = \{s_k(l) \in [n]\}_{l \in [n_k]} \subseteq [n]$ is a pool of indexes chosen from the dataset, $[n]$. *i.e.*,

$$\mathcal{A}_{s_1 \times s_2} \coloneqq \arg\min_{\theta} \tfrac{1}{n_1\, n_2} \sum_{i \in [n_1]} \sum_{j \in [n_2]} \ell(z_{s_1(i)}, z_{s_2(j)};\theta)\,, \tag{3.2}$$

and we formally define DML as $\mathcal{A}_{[n]\times[n]}$, *i.e.*, minimizing empirical expected loss with all possible pairs. We will consider improving the generalization error of $\mathcal{A}_{s_1 \times s_2}$ which is:

$$\mathcal{L}(\mathcal{A}_{s_1 \times s_2}) = \mathbb{E}_{z,z' \sim p_{\mathcal{Z}}}[\ell(z, z';\mathcal{A}_{s_1 \times s_2})]. \tag{3.3}$$

4 Method

We will iteratively solve multiple proxy-based DML problems. At each problem, we re-initialize the class proxies by samples from the dataset. We relate the problems by regularizing the learned parameters to be in the close vicinity of the previous ones. In the following sections, we provide theoretical foundation behind the motivation of our method. We defer all the proofs to Appendix-B.

We start with reformulating DML with a chance constraint. We will introduce two propositions that allow us to decompose the chance constraint into finite chance constraints. We also show minimizer of proxy-based DML satisfies some chance constraints. Hence, we link DML to finding a point in the

intersection of finite sets, which we solve using alternating projections that correspond to regularized proxy-based DML problem instances.

In the formulations throughout the paper, we rely on Lipschitz continuity of the loss function for which we refer to Lemma 4.1. Our methods builds on improving the generalization performance on training domain. Granted, in DML models, the embedding vector is typically obtained by global average pooling of local CNN features. In that manner, local features can be considered as visual words. Thus, if we consider better generalization in training domain, the semantics captured by the visual words can be transferred to represent the samples from an unseen domain. Besides, our empirical studies support that the implications suggested by the formulations are quite effective.

4.1 Reformulation of Deep Metric Learning with Chance Constraints

We consider the solution of the following feasibility problem:

$$\min_{\theta} 0^{\intercal}\theta \quad \text{subject to} \quad p_{z,z'\sim p_{\mathcal{Z}}}(\iota_{y,y'}(d_f(x,x';\theta)-\beta)\geqslant 0)\leqslant \varepsilon\,, \tag{4.1}$$

with some small ε. Namely, we want the probability of observing two samples of the same (different) class being apart (close) more than β in the embedding space being low. We write that probability as expected violation, $\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\mathbb{1}(\iota_{y,y'}(d_f(x,x';\theta)-\beta)\geqslant 0)]$, and bound it for $\beta\geqslant\alpha>0$ as:

$$p_{z,z'\sim p_{\mathcal{Z}}}(\iota_{y,y'}(d_f(x,x';\theta)-\beta)\geqslant 0)\leqslant {}^{1}\!/_{\alpha}\,\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[(\iota_{y,y'}(d_f(x,x';\theta)-\beta)+\alpha)_+]\,, \tag{4.2}$$

using Markov's inequality where $(u)_+=\max\{0,u\}$. Note that to each value of the expectation, $e(\theta)$, there corresponds an $\varepsilon={}^{e(\theta)}\!/_{\alpha}$ which the chance constraint satisfies. Hence, we use the expectation as the surrogate of the penalty term for the chance constraint and can redefine the aforementioned feasibility problem as in problem (3.1) with $\ell(z_1,z_2;\theta)=(\iota_{y,y'}(d_f(x,x';\theta)-\beta)+\alpha)_+$. In particular, we end up with minimization of the expected *generalized contrastive loss* (Wu et al., 2017).

We now consider the relaxed feasibility problem in which we consider m chance constraints conditioned on given m samples $\mathcal{S}=\{z_i\}_{i\in[m]}\sim p_{\mathcal{Z}}$. In other words, we want to find $\theta\in\mathcal{C}_{\mathcal{S}}$ where:

$$\mathcal{C}_{\mathcal{S}}=\{\theta\mid p_{z\sim p_{\mathcal{Z}}}(\iota_{y_i,y}(d_f(x_i,x;\theta)-\beta)\geqslant 0)\leqslant\varepsilon,\ \forall i\in[m]\} \tag{4.3}$$

Using expectation bounds as in Eqn. (4.2), the unconstrained problem becomes:

$$\theta^*=\arg\min_{\theta}\tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(z_i,z;\theta)]\,. \tag{4.4}$$

The following Proposition 4.1 gives an upper bound for the expected loss when we solve (4.4) instead.

Proposition 4.1. *Given $\mathcal{S}=\{z_i\}_{i\in[m]}\overset{i.i.d.}{\sim}p_{\mathcal{Z}}$ such that $\forall k\in\mathcal{Y}$ $\{x_i|y_i=k\}$ is $\delta_{\mathcal{S}}$-cover[2] of $\mathcal{X}$, $\ell(z,z';\theta)$ is ζ^{ℓ}-Lipschitz in x,x' for all y, y' and θ, and bounded by L; then with probability at least $1-\gamma$,*

$$\left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\theta)]-\tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(z_i,z;\theta)]\right|\leqslant\mathcal{O}(\zeta^{\ell}\,\delta_{\mathcal{S}})+\mathcal{O}(L\sqrt{\log\tfrac{1}{\gamma}/m}).$$

Proposition 4.1 gives an upper bound which is controlled by the diversity of the samples defining the relaxed problem. Theoretically, such a controlled bound allows DML to be formulated as a feasibility problem of finite sets for some accepted error tolerance. In practice, the best we can do is using all the samples in the dataset when defining $\mathcal{C}_{\mathcal{S}}$ in (4.3). Granted that the minimization of the empirical loss boils down to the classical DML (3.1), such a formulation using finite feasible sets is a key to characterize generalization of proxy based methods as well as developing our method.

4.2 Reducing Chance Constraints

The problems in (3.1) and (4.4) have the same overall empirical loss but different stochastic optimization procedure. The formulation based on the relaxed problem suggests sampling batch of instances rather than pairs, which yields less noisy gradient estimates with the same batch budget[3].

[2] $\mathcal{S}\subset\mathcal{S}'$ is $\delta_{\mathcal{S}}$-cover of $\mathcal{S}'$ if $\forall z'\in\mathcal{S}'$, $\exists z\in\mathcal{S}$ such that $\|z-z'\|_2\leqslant\delta_{\mathcal{S}}$.

[3] Besides intiution, implied by Thm. 3 in (Xu & Mannor, 2012) and Thm. 1 in (Bellet & Habrard, 2015).

This intuitively explains the superior performance of the methods using batches augmented by class proxies, past embeddings or more samples (*i.e.*, larger batches), especially in large-scale problems.

However, the loss terms conditioned on fixed samples in the formulation of (4.4) are computationally prohibitive in large-scale problems. Thus, we are interested in reducing the chance constraints inducing the loss terms as random variables. To this end, proxy-based methods are quite related.

Proxy-based methods use parametric vectors, $\{\rho_i\}_{i\in[C]}$, to represent embedding of the class centers and minimize the pair losses with respect to those centers. Formally, given a dataset $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$, proxy-based methods consider the following problem:

$$\min_{\theta,\rho} \tfrac{1}{n\,C} \sum_{i\in[C]} \sum_{j\in[n]} \hat{\ell}(\rho_i, z_j; \theta) \tag{4.5}$$

where $\hat{\ell}(\rho_i, z_j; \theta)$ is a loss term in which the pairwise distance is computed as $\|\rho_i - f(x_j;\theta)\|_2$. We can associate an algorithm, $\mathcal{A}_{s\times[n]}$ defined in (3.2), to the minimizer of (4.5) with $e(\mathcal{A}_{s\times[n]})$ training error where $s = \{s(i) \in [n] \mid f(x_{s(i)}; \mathcal{A}_{s\times[n]}) = \rho_i\}_{i\in[C]}$. In other words, to each proxy, we associate a sample whose embedding matches that proxy, assuming such sample exists. Hence, the minimizer of the proxy-based methods can be reformulated as the following feasibility problem:

$$\min_{\theta} \theta^{\intercal} 0 \quad \text{s. to} \quad \theta \in \{\theta \mid p_{z\sim p_{\mathcal{Z}}}(\iota_{i,y}(d_f(x_{s(i)}, x; \theta) - \beta) \geqslant 0) \leqslant \varepsilon,\ \forall i \in [C]\} \tag{4.6}$$

where $\varepsilon = \frac{1}{\alpha}\mathcal{L}(\mathcal{A}_{s\times[n]})$ from (4.2) and $\mathcal{L}(\mathcal{A}_{s\times[n]})$ defined in (3.3) is bounded (Bellet & Habrard, 2015). Reformulation of proxy-based DML defines the feasibility problem with one sample per class.

We now consider more general case where we use m samples per class from the dataset, $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$, to define the feasibility problem. We have m-many disjoint 1-per-class sets, $s = \cup_{k\in[m]} s_k$, where $s_k = \{s_k(i) \in [n] \mid y_{s_k(i)} = i\}_{i\in[C]}$ with $\cap_{k\in[m]} s_k = \emptyset$. We define the problem as:

$$\min_{\theta\in\cap_k \mathcal{C}_{s_k}} 0^{\intercal}\theta \quad \text{where} \quad \mathcal{C}_{s_k} = \{\theta \mid p_{z\sim p_{\mathcal{Z}}}(\iota_{i,y}(d_f(x_{s_k(i)}, x; \theta) - \beta) \geqslant 0) \leqslant \varepsilon,\ \forall i \in [C]\} \tag{4.7}$$

Solving the problem by minimizing the empirical expectation bounds in (4.4), we end up with an algorithm $\mathcal{A}_{s\times[n]}$ in which we are minimizing expected loss over a subset of all possible pairs. We want to characterize the generalization performance of the algorithm $\mathcal{A}_{s\times[n]}$. We consider the following bound proposed in (Sener & Savarese, 2018) for the generalization error:

$$\begin{aligned}\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\mathcal{A}_{s\times[n]})] \leqslant & \left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\mathcal{A}_{s\times[n]})] - \tfrac{1}{n^2}\sum_{i,j\in[n]\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_1)} \\ + & \left|\tfrac{1}{n^2}\sum_{i,j\in[n]\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]}) - \tfrac{1}{\lfloor s\rfloor\, n}\sum_{i,j\in s\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_2)} + \left|\tfrac{1}{\lfloor s\rfloor\, n}\sum_{i,j\in s\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_3)}\end{aligned} \tag{4.8}$$

where the bound is controlled by ($\mathcal{L}_1$) the deviation between expected loss and empirical loss over all possible pairs, ($\mathcal{L}_2$) the deviation between empirical loss over all possible pairs and empirical loss over the subset of pairs defining the algorithm, $\mathcal{A}_{s\times[n]}$, and ($\mathcal{L}_3$) training loss (*i.e.*, $e(\mathcal{A}_{s\times[n]})$). It is widely observed that high capacity CNNs can reach very small training error. Moreover, $\mathcal{L}_1$ is proved to be bounded in (Bellet & Habrard, 2015) and is independent of $\mathcal{A}$. Thus, $\mathcal{L}_2$ characterizes the generalization performance of using the subset of pairs over exploiting all possible pairs.

Proposition 4.2. *Given $\{z_i\}_{i\in[n]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ and a set $s \subset [n]$. If $s = \cup_k s'_k$ with s'_k is the δ_s-cover of $\{i \in [n] \mid y_i = k\}$ (i.e., the samples in class k), $\ell(z, z'; \theta)$ is ζ^{ℓ}-Lipschitz in x, x' for all y, y' and θ, and bounded by L, $e(\mathcal{A}_{s\times[n]})$ training error; then with probability at least $1-\gamma$ we have:*

$$\left|\tfrac{1}{n^2}\sum_{i,j\in[n]\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]}) - \tfrac{1}{\lfloor s\rfloor\, n}\sum_{i,j\in s\times[n]} \ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right| \leqslant \mathcal{O}(\zeta^{\ell}\,\delta_s) + \mathcal{O}(e(\mathcal{A}_{s\times[n]})) + \mathcal{O}(L\sqrt{\tfrac{\log 1/\gamma}{n}})$$

Corollary 4.2.1. *Generalization performance of the proxy-based methods can be limited by the maximum of distances between the proxies and the corresponding class samples in the dataset.*

Proposition 4.2 implies that increasing the number of chance constraints with more samples in the feasible point problem formulation of DML improves the generalization error bound as long as the included samples improve the covering radius of the dataset. In other words, including more samples do not improve the bound unless the covering radius is decreased. Similarly, Corollary

4.2.1 informally suggests possible improvement on the generalization error bound of the proxy-based methods if we manage to introduce more proxies which are spread over the dataset once trained. In practice introducing more proxies generally does not help the performance; since, they eventually coalesce into a single point. Besides, the computation power limits the number of proxies to be included in the formulation. In the next section, we develop an approach to alleviate these problems.

4.3 Solving Feasibility Problem with Alternating Projections

We now introduce our method, outlined in Algorithm 1, exploiting proxy-based training together with satisfying arbitrarily increased chance constraints. In short, we repeatedly solve a proxy-DML and improve the solution by re-initializing the proxies with the new samples reducing the covering radius.

Algorithm 1 ASAP DML

initialize θ^* randomly, given $\{z_i\}_{i\in[n]}\sim p_{\mathcal{Z}}$
initialize ρ^* with random samples, set budget b
repeat
  $\rho \leftarrow GreedyKCenterProxy(\rho^*, b, f(\cdot;\theta^*))$
  repeat
    sample $\{j(i)\in[n]\}_{i\in[m]}\sim[n]$ a batch
    $g_\theta \leftarrow \lambda(\theta^*-\theta)+\nabla_\theta \frac{1}{m|\rho|}\sum_{\rho\times[m]}\ell(\rho_i, z_j;\theta)$
    $g_\rho \leftarrow \nabla_\rho \frac{1}{m|\rho|}\sum_{\rho\times[m]}\ell(\rho_i, z_j;\theta)$
    $(\theta,\rho)\leftarrow ApplyGradient(\theta,\rho,g_\theta,g_\rho)$
  until convergence
  $\theta^*\leftarrow\theta$, $\rho^*\leftarrow\rho$
until convergence

We consider the problem in (4.7) as finding a point in the intersection of the sets. In particular, given dataset $\{z_i\}_{i\in[n]}\sim p_{\mathcal{Z}}$, we have m many 1-per-class sets, $s_k=\{s_k(i)\in[n]\mid y_{s_k(i)}=i\}_{i\in[C]}$, to define the constraint set as $\mathcal{C}_s=\cap_{k\in[m]}\mathcal{C}_{s_k}$. If the sets were closed and convex, the problem would be solvable by alternating projection methods (Bregman, 1967; Bauschke & Lewis, 2000). Nevertheless, it is not uncommon to perform alternating projection methods to non-convex set intersection problems (Pang, 2015; Solomon et al., 2015). Hence, we propose to solve the problem approximately by performing alternating projections onto the feasible sets, $\mathcal{C}_{s_k}$, defined by s_k. At each iteration, k, we solve the following projection problem given $\theta^{(k-1)}$:

$$\theta^{(k)}=\arg\min_{\theta\in\mathcal{C}_{s_k}}\tfrac{1}{2}\|\theta^{(k-1)}-\theta\|_2^2 \quad \text{where } C_{s_k} \text{ is defined in (4.7)} \tag{4.9}$$

Using expectation bounds as the surrogate of the penalty terms for the chance constraints, we have:

$$\theta^{(k)}=\arg\min_{\theta}\tfrac{\lambda}{2}\|\theta^{(k-1)}-\theta\|_2^2+\tfrac{1}{C}\sum_{i\in[C]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(z_{s_k(i)},z;\theta)] \tag{4.10}$$

where λ is a hyperparameter for the projection regularization. We can minimize the resultant loss by using batch stochastic gradient approaches. However, the batch should be augmented by C many samples to compute the loss, which becomes prohibitive for large-scale problems. To alleviate costly embedding computation of C many samples, we propose to use proxies, ρ_i, in place of the embedding of the samples, $z_{s_k(i)}$. Namely, at each iteration k, we initialize $\rho_i=f(z_{s_k(i)};\theta^{(k-1)})$ and solve:

$$\theta^{(k)},\rho^*=\arg\min_{\theta,\rho}\tfrac{\lambda}{2}\|\theta^{(k-1)}-\theta\|_2^2+\tfrac{1}{C}\sum_{i\in[C]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(\rho_i,z;\theta)] \tag{4.11}$$

where the resultant problem we solve at each iteration corresponds to a proxy-based DML.

Proxy selection. Theoretically, we should cycle through the sets until convergence to solve $\theta\in\cap_{k\in[m]}\mathcal{C}_{s_k}$. On the other hand, Proposition 4.2 implies that generalization is improved as long as we end up with converged proxies reducing the covering radius. Therefore, instead of explicitly defining the sets we will alternate on, we greedily select (*i.e.*, initialize) the next proxies on the fly as outlined in Algorithm 2. Given a budget, b, we sample b many instances per class and compute their embeddings to form a pool. We then select the samples that reduce the covering radius most once added to proxy set. This selection is equivalent to k-Center problem as formulated in (Sener & Savarese, 2018). Such a selection of proxies helps

Algorithm 2 Greedy K-Center Proxy

input: proxy set ρ, sampling budget b and $f(\cdot;\theta)$
repeat for each class c
  $s_c\leftarrow\{x_i\mid y_i=c\}_{i\in[b]}$, b-sample-per-class
  initialize $r_c\leftarrow\{\}$, $p\leftarrow f(s_c;\theta)$
  repeat
    $q\leftarrow\arg\max_{u\in p\setminus r_c}\min_{v\in\rho_c\cup r_c}\|u-v\|_2$
    $r_c\leftarrow\{q\}\cup r_c$
  until $|r_c|=|\rho_c|$
return $\cup_c r_c$

converged proxies to be diverse as well as fitting better to training with random data augmentations. $b = 1$ reduces to random sampling which also works in practice. Intuitively, we eventually observe informative samples through the iterations.

We set up the formulation using single proxy per class, it is straightforward to extend to more general case including multiple proxies. The updates of the proxies are not guaranteed to mimic the actual updates of the corresponding samples. With that being said, we will still have a solution, as (4.6) suggests, to feasibility of some chance constraints as long as the converged proxies, ρ^*, are diverse. We empirically observe that the proxies initialized with diverse samples converge to embedding of distinct samples (Fig. 1). Hence, we have solutions to alternating sets with alternating proxies.

Relation to cross-batch-memory (XBM)(Wang et al., 2020). XBM stores past embeddings in a queue based memory which dequeues the oldest ones at each iteration to enqueue the latest batch. If the memory size is relatively larger than the batch size and *slow drift* (Wang et al., 2020) is assumed, the loss terms are conditioned to particular proxies until they are updated. In this manner, XBM can be considered as solving alternating problems of proxy-based DML with *fuzzy* boundaries and $\lambda = 0$.

4.4 Implementation Details

Embedding function. For the embedding function, $f(\cdot;\theta)$, we use CNNs with ReLU activation, max- and average-pooling. In particular, we use ResNetV2-20 (He et al., 2016) for MNIST (LeCun & Cortes, 2010) experiments, and ImageNet (Russakovsky et al., 2015) pretrained BN-Inception (Ioffe & Szegedy, 2015) for the rest. We exploit architectures until the output of the global average pooling layer. We add a fully connected layer to the output of the global average pooling layer to obtain the embedding vectors of size 2 (ResNetV2-20) and 128 (BN-Inception). We state the following lemma to prove our loss is Lipschitz continuous:

Lemma 4.1. *Generalized contrastive loss defined as $\ell(z, z'; \theta) \coloneqq (\iota_{y,y'}(d_f(x, x'; \theta) - \beta) + \alpha)_+$ is $\sqrt{2}\omega^L$-Lipschitz in x and x' for all y, y', θ for the embedding function $f(\cdot;\theta)$ being L-layer CNN (with ReLU, max-pool, average-pool) with a fully connected layer at the end, where ω is the maximum sum of the input weights per neuron.*

ω can be made arbitrarily small by using weight regularization, which is commonly used. SOTA methods widely use L2 normalization on the embeddings. For L2 normalization, we apply $\hat{v} = v/\|v\|_2$ if $\|v\|_2 \geqslant 1$ or no normalization otherwise (*i.e.*, $\hat{v} = v$ if $\|v\|_2 \leqslant 1$). Unlike L2 normalization, such a transform is Lipschitz continuous, hence so are our loss.

Solving projections. Performing a projection defined in (4.11) involves a minimization problem. We monitor MAP@R validation accuracy and use early stopping patience of 3 to pass the next projection.

5 Experimental Work

We examine the effectiveness of the proposed proxy-based DML framework for the image retrieval task. We further perform ablation studies for the implications of our formulation as well as the effects of the hyperparameters. We use our own framework implemented in Tensorflow (Abadi et al., 2016) library in the experiments. Throughout the section, we use ASAP to refer our framework.

5.1 Experimental Setup

We mostly follow the procedures proposed in (Musgrave et al., 2020) to provide fair and unbiased evaluation of our method as well as comparisons with the other invented methods. We provide full detail of our experimental setup in Appendix-A.2 for complete transparency and reproducibility.

Datasets. CUB-200-2011 (CUB) (Wah et al., 2011), Cars196 (Krause & Golovin, 2014), InShop (Liu et al., 2016), Stanford Online Products (SOP) (Oh Song et al., 2016) with Musgrave et al. (2020)'s data augmentation.

Evaluation metrics. Precision at 1 (P@1), precision (P@R) and mean average precision (MAP@R) at R where R is defined for each query and is the total number of true references of the query.

Training. Default Adam (Kingma & Ba, 2014) optimizer with 10^{-5} learning rate, 10^{-4} weight decay, mini-batch size of 32 (4 samples per class), 4-fold: 4 models (1 for each $3/4$ partition of the train set).

Evaluation. Average performance (128-D) and Ensemble (concatenated) performance (512-D).

Losses with ASAP. *C1-ASAP*: Contrastive loss (Hadsell et al., 2006), *C2-ASAP*: Contrastive loss with positive margin (Wu et al., 2017), *MS-ASAP*: Multi-similarity (MS) loss (Wang et al., 2019b), *Triplet-ASAP*: Triplet loss (Schroff et al., 2015).

ASAP hyperparameters. λ=2 10^{-4}, 8(4) proxies per class with pool size of 12(7) (*i.e.*, b=12(7) in greedy k-Center method) for CUB and Cars (Inshop and SOP).

Compared methods and fairness. We compare our method against proxy-based SoftTriple (Qian et al., 2019), ProxyAnchor (Kim et al., 2020) and ProxyNCA++ (Teh et al., 2020) methods as well as XBM (Wang et al., 2020). We note that like the compared methods, our method's improvement claims do not demand any particular architecture or experimental setup. Therefore, to evaluate the improvements purely coming from the proposed ideas, we implemented the best version of the compared methods in our framework and evaluate on the same architecture and experimental settings.

5.2 Ablation Study

Proof of the concept. We evaluate our method on MNIST dataset with 2-D embeddings to show the implications of our formulation. In Fig. 1, we provide the distribution of the samples in the embedding space. We use 4 proxies per class and pool size b=16. We observe that when single proxy-based method is converged (Fig. 1-(a)), the class proxies collapse to a single point. Once we continue training with proposed approach (Fig. 1-(b)), the covering radius decreases, leading to performance improvement. It can also be observed that diverse samples results in diverse proxies. We also experiment the case where we use samples instead of proxies. Though it is not practically applicable to large-scale problems, it is important to see whether our intuitions about alternating proxies in place of samples hold. We obtain 98.06% MAP@R performance with sample-based training against 97.21% MAP@R performance of proxy-based training. This empirical result supports our motivation on using the proxies in place of samples.

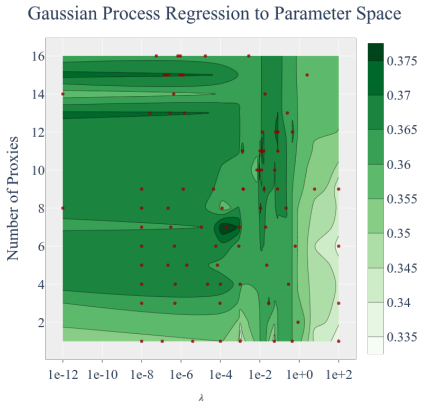


Figure 2: Bayesian search on λ-$\#proxy$

Effect of proxy per class and projection regularization. We perform Bayesian search on the λ-$\#proxy$ space to see the effect of two in CUB with C2-ASAP. We provide the results in Fig. 2. We observe that absence of λ degrades the performance. Similarly, large values of λ causes over-regularization. We obtain interval of $[10^{-1}, 10^{-5}]$ that works well for λ. For the number of proxies, we observe increasing the proxy per class improves performance. On the other hand, the increase saturates as it can also be observed from Fig. 3. As the result of Bayesian parameter search, we take λ=2 10^{-4} and $\#proxy$=8 with pool size b=12 in our evaluations against other methods.

Effect of proxy selection. We perform ablation study with C2-ASAP to see the relation between the number of proxies and the pool size used for the proxy selection. We give the corresponding results in Fig. 3. We observe that both increasing the number of proxies and the pool size for proxy selection helps performance. We interestingly see that for single proxy case, increasing the pool size gives no better results than random selection. Owing to our greedy proxy selection, we do consider the past geometry no earlier than single step. Thus, in the single proxy case, we are prune to oscillate between similar samples for proxy selection. On the other hand, selecting the samples that reduce the covering radius most brings better generalization over random selection.

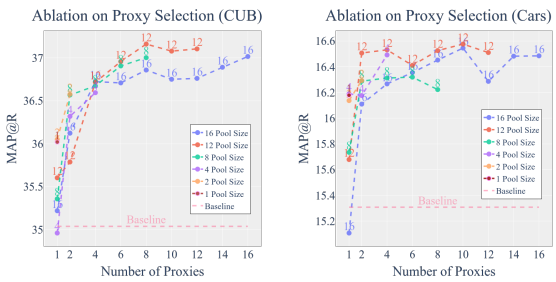
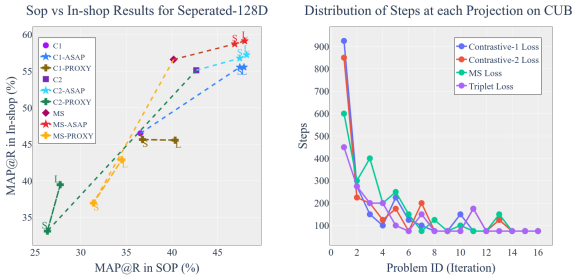


Figure 3: Analysis of the relation between the number of proxies and the pool size used for the proxy selection on CUB (left) and Cars (right) dataset with C2-ASAP.

Effect of alternating problems. We provide results on MNIST in Fig. 1 to show the effect of solving alternating problems instead of single proxy-based DML. We additionally evaluate the baseline losses through solving only a single proxy-DML (Loss-Proxy) to show (Fig. 4-(a)) that our performance increase is not solely coming from augmentation of proxies in the problem. We clearly observe that alternating proxies helps performance as our formulation suggests. Moreover, we also provide a typical distribution of the steps per proxy-based projection problem in Fig. 4-(b) to show that we are not greedy on alternating the proxies just to provide more examples.



(a) (b)

Figure 4: Impact of alternating proxies (a) and typical distribution of the steps per projection problem (b).

5.3 Quantitative Results

We provide quantitative results in Table 1. We summarize the results in Fig. 5 through average 128-D MAP@R performance of 4-fold models on InShop and SOP. We use Method-S/L naming convention to denote memory size in XBM and the proxy per class in SoftTriple, ASAP where S denotes 1, and L denotes 4(10) for SoftTriple and 4(8) for ASAP in InShop, SOP (CUB, Cars196). For a fair comparison, we match XBM memory size and the number of proxies in ASAP. We observe that ASAP consistently outperforms the associated baseline methods on each dataset. Contrastive loss' great performance with ASAP is important to support the implications of our formulation. Furthermore, performance improvements on the loss functions which does not directly fit in our formulation show that the broader applicability of our method to the pairwise distance based loss functions. Additionally, the proposed ASAP framework outperforms not only the related proxy-based methods but also every single benchmarked approaches in (Musgrave et al., 2020). When compared with SoftTriple and XBM (*i.e.*, multiple proxy methods), our method outperforms by large margin especially in the cases where less number of proxies are used.

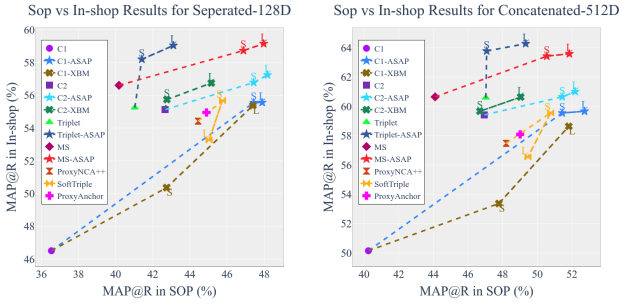


Figure 5: Summary of relative improvements

Table 1: Comparison with the existing methods for the retrieval task on SOP, InShop, CUB, Cars

	SOP				InShop				CUB				Cars196			
	512D		128D		512D		128D		512D		128D		512D		128D	
Method	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**
C1	68.84	40.25	64.96	36.54	80.12	50.15	76.08	46.51	63.67	23.08	56.21	19.06	77.75	23.50	64.17	16.13
C1-XBM-L	78.68	51.82	75.37	47.39	88.39	58.64	**85.75**	55.37	65.40	24.87	57.57	19.84	83.68	27.93	**72.13**	18.83
C1-ASAP-L	**79.53**	**52.73**	**76.24**	**48.07**	**88.52**	**59.67**	85.50	**55.56**	**68.11**	**27.11**	**59.56**	**21.27**	**83.76**	**28.32**	72.05	**18.96**
C2	74.87	46.94	71.15	42.66	86.32	59.42	83.04	55.13	67.49	26.47	59.73	21.01	81.04	24.73	69.17	17.22
C2-XBM-L	76.66	49.04	73.47	45.15	87.66	60.64	84.58	56.75	68.62	26.83	60.18	21.41	82.40	25.99	70.01	18.01
C2-ASAP-L	**78.95**	**52.19**	**75.92**	**48.18**	**88.52**	**61.07**	**86.11**	**57.24**	**69.73**	**28.02**	**62.39**	**22.67**	**82.89**	**26.27**	**72.16**	**18.52**
MS	72.74	44.10	68.96	40.18	88.37	60.65	85.39	56.61	64.65	24.15	57.24	19.64	80.88	26.23	69.27	18.25
MS-ASAP-L	**78.96**	**51.85**	**75.80**	**47.97**	**90.24**	**63.59**	**87.10**	**59.15**	**68.84**	**27.44**	**61.10**	**22.40**	**86.26**	**29.14**	**74.97**	**19.85**
Triplet	75.40	47.03	70.41	41.03	86.71	60.60	82.58	55.25	64.01	23.43	55.51	18.51	78.44	23.11	64.57	15.68
Triplet-ASAP-L	**77.09**	**49.33**	**72.21**	**43.11**	**89.44**	**64.28**	**86.00**	**59.04**	**65.36**	**24.53**	**56.65**	**19.31**	**81.84**	**25.21**	**68.75**	**17.43**
ProxyAnchor	77.10	49.01	73.86	44.89	88.08	58.09	85.87	54.95	68.43	26.53	60.61	21.48	85.29	27.73	**75.79**	19.56
ProxyNCA++	76.07	48.20	72.89	44.44	87.33	57.48	84.79	54.42	65.48	24.85	58.49	20.96	82.87	26.34	72.45	19.32
SoftTriple-S/L	78.48	50.77	74.66	45.75	88.37	59.56	85.71	55.68	66.10	24.06	56.97	18.82	84.90	27.80	73.16	19.18

6 Conclusion

We bring a difference perspective to DML formulation in terms of chance constraints. As a result, we develop a proxy-based method that implicitly considers arbitrary number of proxies for better generalization. Extensive evaluations on the benchmark datasets show the efficiency of our method.

REPRODUCIBILITY

We provide full detail of our experimental setup and recapitulate thr implementation details in Appendix-A.2 for the sake of complete transparency and reproducibility. Code is available at: ASAP-DML Framework.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Heinz H Bauschke and Adrian S Lewis. Dykstras algorithm with bregman projections: A convergence proof. *Optimization*, 48(4):409–427, 2000.

Aurélien Bellet and Amaury Habrard. Robustness and generalization for metric learning. *Neurocomputing*, 151: 259–267, 2015.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7 (3):200–217, 1967.

Binghui Chen, Weihong Deng, and Haifeng Shen. Virtual class enhanced discriminative embedding learning. *Advances in Neural Information Processing Systems*, 31:1942–1952, 2018.

Thanh-Toan Do, Toan Tran, Ian Reid, Vijay Kumar, Tuan Hoang, and Gustavo Carneiro. A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

M Dong, X Yang, R Zhu, Y Wang, and J Xue. Generalization bound of gradient descent for non-convex metric learning. Neural Information Processing Systems Foundation, 2020.

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*, pp. 2780–2789, 2018.

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

Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 269–285, 2018.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pp. 1735–1742. IEEE, 2006.

Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2829, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3238–3247, 2020.

Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 736–751, 2018.

Yonghyun Kim and Wonpyo Park. Multi-level distance regularization for deep metric learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1827–1835, 2021.

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

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*, pp. 7255–7264, 2020.

Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 689–704, 2018.

Chang Liu, Han Yu, Boyang Li, Zhiqi Shen, Zhanning Gao, Peiran Ren, Xuansong Xie, Lizhen Cui, and Chunyan Miao. Noise-resistant deep metric learning with ranking-based instance selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6811–6820, 2021.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1096–1104, 2016.

Timo Milbich, Karsten Roth, Homanga Bharadhwaj, Samarth Sinha, Yoshua Bengio, Björn Ommer, and Joseph Paul Cohen. Diva: Diverse visual feature aggregation for deep metric learning. In *European Conference on Computer Vision*, pp. 590–607. Springer, 2020.

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*, pp. 360–368, 2017.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pp. 681–699. Springer, 2020.

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*, pp. 4004–4012, 2016.

Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5189–5198, 2017.

CH Pang. Nonconvex set intersection problems: From projection methods to the newton method for super-regular sets. *arXiv preprint arXiv:1506.08246*, 2015.

Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *International Conference on Learning Representations*, 2016.

Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pp. 8242–8252. PMLR, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66, 2015.

Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

Aad W Van Der Vaart, Aad van der Vaart, Adrianus Willem van der Vaart, and Jon Wellner. *Weak convergence and empirical processes: with applications to statistics*. Springer Science & Business Media, 2013.

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

Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M. Robertson. Ranked list loss for deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019a.

Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019b.

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*, pp. 6388–6397, 2020.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pp. 1473–1480, 2006.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840–2848, 2017.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 723–734, 2018.

Baosheng Yu and Dacheng Tao. Deep metric learning with tuplet margin loss. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE international conference on computer vision*, pp. 814–823, 2017.

Dingyi Zhang, Yingming Li, and Zhongfei Zhang. Deep metric learning with spherical embedding. *Advances in Neural Information Processing Systems*, 33, 2020.

Wenliang Zhao, Yongming Rao, Ziyi Wang, Jiwen Lu, and Jie Zhou. Towards interpretable deep metric learning with structural matching. *arXiv preprint arXiv:2108.05889*, 2021.

Yiru Zhao, Zhongming Jin, Guo-jun Qi, Hongtao Lu, and Xian-sheng Hua. An adversarial approach to hard triplet generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 501–517, 2018.

Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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.

A SUPPLEMENTARY MATERIAL FOR EXPERIMENTAL WORK

A.1 REPRODUCIBILITY

Code is available at: ASAP-DML Framework.

A.2 EXPERIMENTAL SETUP

Datasets. We perform our experiments on four widely-used benchmark datasets: Stanford Online Products (SOP) (Oh Song et al., 2016), InShop (Liu et al., 2016), Cars196 (Krause & Golovin, 2014) and, CUB-200-2011 (CUB) (Wah et al., 2011). SOP Oh Song et al. (2016) has 22,634 classes with 120,053 product images. The first 11,318 classes (59,551 images) are split for training and the other 11,316 (60,502 images) classes are used for testing. InShop has 7,986 classes with 72,712 images. We use 3,997 classes with 25,882 images as the training set. For the evaluation, we use 14,218 images of 3,985 classes as the query set and 12,612 images of 3,985 classes as the gallery set. Cars196 contains 196 classes with 16,185 images. The first 98 classes (8,054 images) are used for training and remaining 98 classes (8,131 images) are reserved for testing. CUB-200-2011 dataset consists of 200 classes with 11,788 images. The first 100 classes (5,864 images) are split for training, the rest of 100 classes (5,924 images) are used for testing.

Training Splits. We split datasets into disjoint training, validation and test sets according to (Musgrave et al., 2020). In particular, we partition $^{50\%}/_{50\%}$ for training and test, and further split training data to 4 partitions where 4 models are to be trained by exploiting $^1/_4$ as validation while training on $^3/_4$. For the ablation studies, we split training set into 3 splits instead of 1 and train a single model on the $^2/_3$ of the set while using $^1/_3$ for the validation.

Data augmentation follows (Musgrave et al., 2020). During training, we resize each image so that its shorter side has length 256, then make a random crop between 40 and 256, and aspect ratio between $^3/_4$ and $^4/_3$. We resize the resultant image to 227x227 and apply random horizontal flip with 50% probability. During evaluation, images are resized to 256 and then center cropped to 227x227.

Evaluation metrics. We consider precision at 1 (P@1), precision (P@R) and mean average precision (MAP@R) at R where R is defined for each query[4] and is the total number of true references as the query. Among those, MAP@R performance metric is shown to better reflect the geometry of the embedding space and to be less noisy as the evaluation metric (Musgrave et al., 2020). Thus, we use MAP@R to monitor training.

P@1: Find the nearest reference to the query. The score for that query is 1 if the reference is of the same class, 0 otherwise. Average over all queries gives P@1 metric.

P@R: For a query, i, find R_i nearest references to the query and let r_i be the number of true references in those R_i-neighbourhood. The score for that query is $\text{P@R}_i = {}^{r_i}/_{R_i}$. Average over all queries gives P@R metric, *i.e.*, $\text{P@R} = \frac{1}{n} \sum_{i \in [n]} \text{P@R}_i$, where n is the number of queries.

MAP@R: For a query, i, we define $\text{MAP@R}_i \coloneqq \frac{1}{R_i} \sum_{i \in [R_i]} P(i)$, where $P(i) = \text{P@i}_i$ if i^{th} retrieval is correct or 0 otherwise. Average over all queries gives MAP@R metric, *i.e.*, $\text{MAP@R} = \frac{1}{n} \sum_{i \in [n]} \text{MAP@R}_i$, where n is the number of queries.

Training procedure. For the optimization procedure, we use Adam (Kingma & Ba, 2014) optimizer for mini-batch gradient descent with a mini-batch size of 32 (4 samples per class), 10^{-5} learning rate, 10^{-4} weight decay, default moment parameters, β_1=.9 and β_2=.99. We evaluate validation MAP@R for every 25 steps of training in CUB and Cars196, for 250 steps in SOP and InShop. We stop training if no improvement is observed for 60 steps and recover the parameters with the best validation performance. Following (Musgrave et al., 2020), we train 4 models for each $^3/_4$ partition of the train set. For the ablation studies, we train a single model on the $^2/_3$ partition.

[4] A query is an image for which similar images are to be retrieved, and the references are the images in the searchable database.

Embedding vectors. Embedding dimension is fixed to 128. During training and evaluation, the embedding vectors are L2 normalized using the transformation proposed in Section 4.4. We follow the evaluation method proposed in (Musgrave et al., 2020) and produce two results: i) Average performance (128 dimensional) of 4-fold models and ii) Ensemble performance (concatenated 512 dimensional) of 4-fold models where the embedding vector is obtained by concatenated 128D vectors of the individual models before retrieval.

Losses with ASAP. We evaluate our method with *C1-ASAP*: Contrastive loss (Hadsell et al., 2006), *C2-ASAP*: Contrastive loss with positive margin (Wu et al., 2017), *MS-ASAP*: Multi-similarity (MS) loss (Wang et al., 2019b), *Triplet-ASAP*: Triplet loss (Schroff et al., 2015).

Compared methods. We compare our method against proxy-based SoftTriple (Qian et al., 2019), ProxyAnchor (Kim et al., 2020) and ProxyNCA++ (Teh et al., 2020) methods as well as XBM (Wang et al., 2020).

Fairness. We note that like the compared methods (*i.e.*, loss functions, proxy-based methods), our method's improvement claims do not demand any particular architecture or experimental setup. Therefore, to evaluate the improvements purely coming from the proposed ideas, we implemented the best version of the compared methods in our framework and evaluate on the same architecture and experimental settings. In this manner, we stick to BN-Inception with global average pooling architecture to directly compare our method with the benchmarked losses in (Musgrave et al., 2020). To eliminate any framework related performance differences, we re-implemented the methods within our framework and produce the consistent results with (Musgrave et al., 2020).

Our experimental setting is fair and unbiased; since,

- The compared methods are either invented loss functions or proxy-based approaches, which do not demand a particular setting to show the effectiveness of the proposed ideas.
- We use the same experimental setting for each method (*e.g.* image size, architecture, embedding size, batch size, data augmentation).
- We implement and re-evaluate all the compared methods on our framework (*i.e.*, train and evaluate).
- We reproduce consistent results reported in (Musgrave et al., 2020) to eliminate any framework related performance bias.
- We use the same train and test split as the conventional methods, but we do not exploit test data during training. We use $1/4$ split of train data for the validation set.

Hyperparameters. For the hyperparameter selection, we exploit the recent work (Musgrave et al., 2020) that has performed parameter search via Bayesian optimization on variety of losses. We further experiment the suggested parameters from the original papers and official implementations. We pick the best performing parameters. We perform no further parameter tuning for the loss parameters when applied to our method to purely examine the effectiveness of our method.

C1: We adopted XBM's official implementation for fair comparison. We use 0.5 margin for all datasets.

C2: C2 has two parameters, (m^+, m^-): positive margin, m^+, and negative margin. We set (m^+, m^-) to $(0, 0.3841)$, $(0.2652, 0.5409)$, $(0.2858, 0.5130)$, $(0.2858, 0.5130)$ for CUB, Cars196, InShop and SOP, respectively.

Triplet: We set its margin to 0.0961, 0.1190, 0.0451, 0.0451 for CUB, Cars196, InShop and SOP, respectively.

MS: MS has three parameters (α, β, λ). We set (α, β, λ) to $(2, 40, 0.5)$, $(14.35, 75.83, 0.66)$, $(8.49, 57.38, 0.41)$, $(2, 40, 0.5)$ for CUB, Cars196, InShop and SOP, respectively.

ProxyAnchor: We set its two paremeters (δ, α) to $(0.1, 32)$ for all datasets. We use 1 sample per class in batch setting (*i.e.*, 32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

ProxyNCA++: We set its temperature parameter to 0.1 for all datasets. We use 1 sample per class in batch setting (*i.e.*, 32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

SoftTriple: SoftTriple has 4 parameters $(\lambda, \gamma, \tau, \delta)$. We set $(\lambda, \gamma, \tau, \delta)$ to $(20, 0.1, 0.2, 0.01)$, $(17.69, 19.18, 0.0669, 0.3588)$, $(20, 0.1, 0.2, 0.01)$, $(100, 47.9, 0.2, 0.3145)$ for CUB, Cars196, InShop and SOP, respectively. We use 1 sample per class in batch setting (*i.e.*, 32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

XBM: We evaluate XBM with C1 and C2; since, in the original paper, contrastive loss is reported to be the best performing baseline with XBM. We set the memory size of XBM to the total number of proxies (*i.e.*, $proxy\_per\_class \times \#classes$) to compare the methodology by disentangling the effect of proxy number. With that being said, we also evaluate XBM with the memory sizes suggested in the original paper. In this manner we use two memory sizes for XBM for each dataset: (S, L) where S and L denote the number of batches in the memory. For CUB and Cars196, ASAP uses 1(8) proxies per class for $S(L)$. Thus, we set (S, L) to $(3, 25)$ for CUB and Cars196. For InShop and SOP, ASAP uses 1(4) proxies per class for $S(L)$. Thus, we set (S, L) to $(100, 400), (400, 1400)$ for InShop and SOP, respectively. We perform 1K steps of training with the baseline loss prior to integrate XBM loss in order to ensure *slow drift* assumption.

ASAP: For the hyperparameters of our method, we use 8 proxies per class and λ=2 10^{-4} for CUB and Cars datasets, as the result of the parameter search; and use pool size, b=12, for greedy k-Center method. We select pool size based on our empirical studies on the effect pool size and number of proxies. Due to computation limitations, we use 4 proxy per class, λ=2 10^{-4} and b= 7 for SOP and InShop dataset. We perform no warm-up or do not use learning rate multiplier for the proxies.

A.3 Further Experimental Results

A.3.1 Proof of the Concept Tests

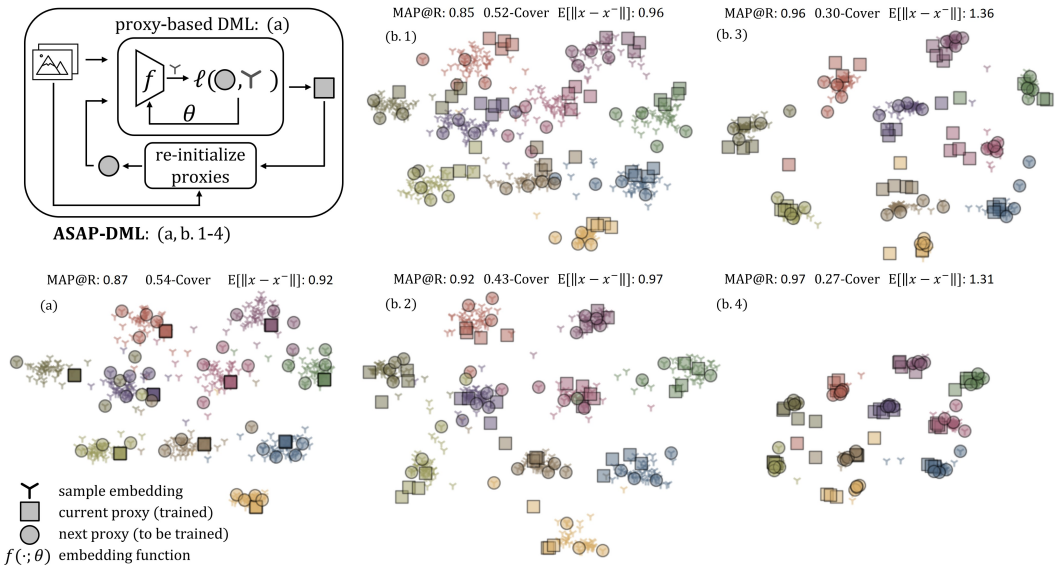


Figure 6: Illustration of our method and the geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-4), where boxes are the converged proxies and the circles are the next proxies as the result of k-Center. In proxy-based DML, proxies are coalesced into one whereas with ASAP, we have diverse proxies, resulting reduced covering radius.

Our proof of the concept visualization in Fig. 6 is to show that the proposed method helps to reduce the covering radius. Thus, we visualize the training data to illustrate proxy convergence, proxy selection and embedding space geometry. With that being said, it is important to show how such efforts in the training domain are reflected in the test domain.

We further provide the visualization of the validation data in CUB dataset in Fig. 7. We use 2-D TSNE embeddings of the validation data in the visualization. We compute covering radii for 1 to n sample case in k-Center. Namely, we take k samples with minimum cover for $k \in [n]$ where n is

the number of samples per class. We then take the average of these radii to compute a representative metric for the covering radius.

We observe that solving single proxy-based DML results in relatively poor generalization in the test domain. On the contrary, solving the problem as the set intersection problem with alternating projections improves the embedding geometry (reduced radius with increased inter-class pairwise distances).

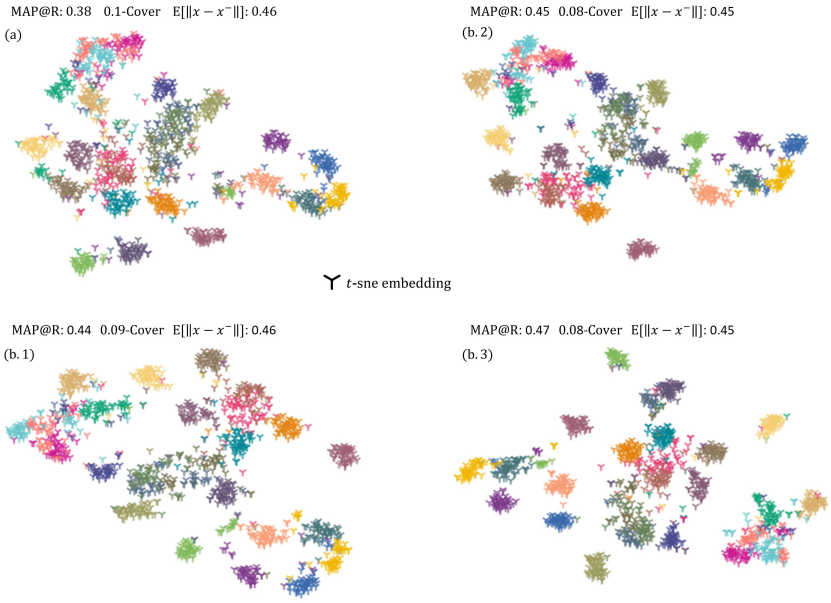


Figure 7: The geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-3), relating how the generalization efforts in training domain transfer to the geometry of test domain on CUB dataset with C2-ASAP. We use 2-D TSNE embeddings of the validation data in the visualization. We report MAP@R, average covering radius and average inter-class pairwise distances in the visualization.

A.3.2 Results with P@R Performances

Table 2: Comparison with the existing methods for the retrieval task on SOP and InShop. Red: the overall best. Bold: the loss term specific best.

	SOP						InShop					
	512D			128D			512D			128D		
Method	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**
C1	68.84	43.28	40.25	64.96	39.68	36.54	80.12	53.11	50.15	76.08	49.61	46.51
C1-XBM-L	78.68	54.66	51.82	75.37	49.95	47.39	88.39	61.33	58.64	**85.75**	58.13	55.37
C1-ASAP-L	**79.53**	**55.11**	**52.73**	**76.24**	**50.07**	**48.07**	**88.52**	**62.54**	**59.67**	85.50	**58.51**	**55.56**
C2	74.87	49.88	46.94	71.15	45.77	42.66	86.32	62.36	59.42	83.04	58.27	55.13
C2-XBM-L	76.66	51.91	49.04	73.47	48.18	45.15	87.66	63.50	60.64	84.58	59.78	56.75
C2-ASAP-L	**78.95**	**55.01**	**52.19**	**75.92**	**51.14**	**48.18**	**88.52**	**63.94**	**61.07**	**86.11**	**60.16**	**57.24**
MS	72.74	47.07	44.10	68.96	43.25	40.18	88.37	63.53	60.65	85.39	59.65	56.61
MS-ASAP-L	**78.96**	**54.71**	**51.85**	**75.80**	**50.48**	**47.97**	**90.24**	**66.31**	**63.59**	**87.10**	**61.74**	**59.15**
Triplet	75.40	50.13	47.03	70.41	44.32	41.03	86.71	63.81	60.60	82.58	58.74	55.25
Triplet-ASAP-L	**77.09**	**52.42**	**49.33**	**72.21**	**46.38**	**43.11**	**89.44**	**67.23**	**64.28**	**86.00**	**62.24**	**59.04**
ProxyAnchor	77.10	51.95	49.01	73.86	47.94	44.89	88.08	60.91	58.09	85.87	57.80	54.95
ProxyNCA++	76.07	51.17	48.20	72.89	47.47	44.44	87.33	60.33	57.48	84.79	57.33	54.42
SoftTriple-S	78.48	53.68	50.77	74.66	48.79	45.75	88.37	62.56	59.56	85.71	58.74	55.68

Table 3: Comparison with the existing methods for the retrieval task on CUB and Cars196. Red: the overall best. Bold: the loss term specific best.

	CUB						Cars196					
	512D			128D			512D			128D		
Method	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**	**P@1**	**P@R**	**MAP@R**
C1	63.67	33.77	23.08	56.21	29.65	19.06	77.75	33.69	23.50	64.17	26.50	16.13
C1-XBM-L	65.40	35.57	24.87	57.57	30.42	19.84	83.68	37.74	27.93	**72.13**	28.55	18.83
C1-ASAP-L	**68.11**	**37.85**	**27.11**	**59.56**	**32.06**	**21.27**	**83.76**	**37.78**	**28.32**	72.05	**28.74**	**18.96**
C2	67.49	37.18	26.47	59.73	31.86	21.01	81.04	34.97	24.73	69.17	27.70	17.22
C2-XBM-L	68.62	37.53	26.83	60.18	32.25	21.41	82.40	36.07	25.99	70.01	28.49	18.01
C2-ASAP-L	**69.73**	**38.69**	**28.02**	**62.39**	**33.49**	**22.67**	**82.89**	**36.27**	**26.27**	**72.16**	**28.98**	**18.52**
MS	64.65	34.84	24.15	57.24	30.29	19.64	80.88	36.45	26.23	69.27	28.93	18.25
MS-ASAP-L	**68.84**	**38.19**	**27.44**	**61.10**	**33.23**	**22.40**	**86.26**	**38.97**	**29.14**	**74.97**	**30.44**	**19.85**
Triplet	64.01	34.55	23.43	55.51	29.38	18.51	78.44	33.83	23.11	64.57	26.52	15.68
Triplet-ASAP-L	**65.36**	**35.42**	**24.53**	**56.65**	**30.17**	**19.31**	**81.84**	**35.61**	**25.21**	**68.75**	**28.21**	**17.43**
ProxyAnchor	68.43	37.36	26.53	60.61	32.36	21.48	85.29	37.53	27.73	**75.79**	29.91	19.56
ProxyNCA++	65.48	35.60	24.85	58.49	31.73	20.96	82.87	36.56	26.34	72.45	29.91	19.32
SoftTriple-L	66.10	34.99	24.06	56.97	29.63	18.82	84.90	37.69	27.80	73.16	29.60	19.18

A.3.3 Computational Analysis

Batch size. We analyze the dependency of the performance on the batch size; since, batch size plays important role in DML methods to perform well. Therefore, it is worth to analyze the robustness to the batch size especially for the cases where increasing the batch size is prohibitive. We trained baseline contrastive loss and ASAP contrastive loss for the batch sizes of 16, 32, 64 and 128. The training setup is the same we do in the state-of-the-art comparison. In each batch we use 4 samples per class. We provide the results in Fig. 8. We observe that baseline contrastive loss has increasing performance as the batch size increases whereas our method's performance with small batch size is on par with the large batch size. Thus, our method has reduced batch size complexity.

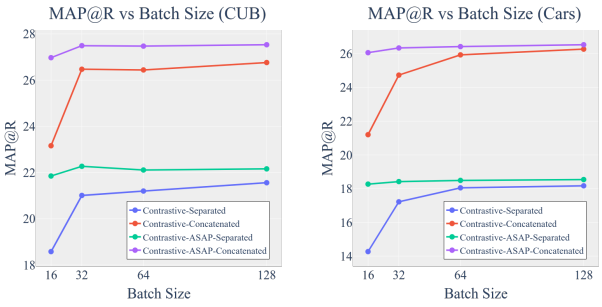


Figure 8: Analysis of batch size dependence of the performance on CUB (left) and Cars (right) dataset with C2-ASAP.

Complexity. Proposed method outlined in Algorithm-1 puts little computation and memory overhead on top of the traditional approaches.

For the computation, we have proxy initialization and weight update steps at the beginning of the each problem instance. In overall, in our system with RTX 2080 Ti GPU and i7 CPU, that additional computation adds on the average 5-10 ms per step (batch update). In particular, for batch size of 32, we typically have rate of 105 ms/batch with Contrastive-ASAP whereas vanilla has 97 ms/batch rate. In InShop and SOP dataset, we have the same rates however for ASAP, we have 200 to 400 ms computation overhead due to sampling for proxy initialization. We do not have such overhead in Cars and CUB owing to the much less number of classes. Granted, we have such 400 ms overhead in InShop and SOP only at the beginning of new problem instance, which has no significant effect in long run. Due to alternating problems, our method takes more steps to converge than their baseline counterparts.

For the memory, we store the weighs of the previously converged model in the memory as well as the variables for proxies. For the model, approximately 40-45 mb additional GPU memory is used and for the proxies 16.6 mb and 5.9 mb memory is used in SOP and InShop dataset (75 kb in CUB and Cars).

B PROOFS

B.1 PROOF FOR LEMMA 4.1

Lemma 4.1. *Generalized contrastive loss defined as* $\ell(z, z'; \theta) \coloneqq (\iota_{y,y'}(d_f(x, x'; \theta) - \beta) + \alpha)_+$ *is* $\sqrt{2}\omega^L$*-Lipschitz in* x *and* x' *for all* y, y', θ *for the embedding function* $f(\cdot; \theta)$ *being* L*-layer CNN (with ReLU, max-pool, average-pool) with a fully connected layer at the end, where* ω *is the maximum sum of the input weights per neuron.*

Proof. We first show that $f(x; \theta)$ is Lipschitz continuous.

We consider $x \in \mathbb{R}^d$ as an input to a layer and $\hat{x} \in \mathbb{R}^{d'}$ as the corresponding output. We express i^{th} component of $\hat{x}$ as $\hat{x}_i = \sum_j w_{i,j} x_{s_i(j)}$ where $s_i = \{s_i(j) \in [d]\}$ is the set of components contributing to $\hat{x}_i$ and $w_{i,j} \in \theta$ is the layer weights. For instance, for a fully connected layer $s_i(j) = j$; for a 3×3 convolutional layer, s_i corresponds to 3×3 window of depth $\#channels$ centered at i. We now consider two inputs x, x' and their outputs $\hat{x}, \hat{x}'$. We write:

$$\frac{\|\hat{x} - \hat{x}'\|_2^2}{\|x - x'\|_2^2} = \frac{\sum_{i \in [d']} |\hat{x}_i - \hat{x}'_i|^2}{\|x - x'\|_2^2} = \frac{\sum_{i \in [d']} |\sum_j w_{i,j} x_{s_i(j)} - \sum_j w_{i,j} x'_{s_i(j)}|^2}{\|x - x'\|_2^2}$$
$$\leqslant \frac{\sum_{i \in [d']} \sum_j |w_{i,j}|^2 |x_{s_i(j)} - x'_{s_i(j)}|^2}{\|x - x'\|_2^2}$$

Rearranging terms, we express:

$$\sum_{i \in [d']} \sum_j |w_{i,j}|^2 |x_{s_i(j)} - x'_{s_i(j)}|^2 = \sum_{k \in [d]} \sum_{i,j : s_i(j) = k} |w_{i,j}|^2 |x_k - x'_k|^2$$

If $\sum_{i,j : s_i(j) = k} |w_{i,j}| \leqslant \omega$ for all k and for all layers, *i.e.*, the absolute sum of the input weights per neuron is bounded by ω, we can write $\sum_{k \in [d]} \sum_{i,j : s_i(j) = k} |w_{i,j}|^2 |x_k - x'_k|^2 \leqslant \omega^2 \sum_{k \in [d]} |x_k - x'_k|^2 \leqslant \omega^2 \|x - x'\|_2^2$, hence,

$$\frac{\|\hat{x} - \hat{x}'\|_2}{\|x - x'\|_2} \leqslant \omega.$$

For max-pooling and average-pooling layers, the inequality holds with $\omega = 1$; since, we can express max-pooling as a convolution where only one weight is 1 and the rest is 0; and similarly, we can express average-pooling as a convolution where the weights sum up to 1.

For ReLU activation, we consider the fact that $|\max\{0, u\} - \max\{0, v\}| \leqslant |u - v|$ to write:

$$\frac{\|ReLU(x) - ReLU(x')\|_2}{\|x - x'\|_2} \leqslant 1.$$

Therefore, L-layer CNN $f(x; \theta)$ is ω^L-Lipschitz.

We now consider $\ell(z, z'; \theta) = \max\{0, \iota_{y,y'}(\|f(x; \theta) - f(x'; \theta)\|_2 - \beta) + \alpha\}$ as $g(h(f(x; \theta), f(x'; \theta)))$ where $g(h) = \max\{0, \iota_{y,y'}(h - \beta) + \alpha\}$ is 1-Lipschitz, and $h(f, f') = \|f - f'\|_2$ is $\sqrt{2}$-Lipschitz and 1-Lipschitz in f for fixed f'. Thus, for y, y', θ fixed, $\ell(z, z'; \theta) \coloneqq (\iota_{y,y'}(d_f(x, x'; \theta) - \beta) + \alpha)_+$ is ω^L-Lipschitz in x and in x'; and $\sqrt{2}\omega^L$-Lipschitz in both, for all y, y', θ. □

Note that it is easy to show that the normalization proposed in Section 4.4:

$$\hat{v} = \begin{cases} v & \text{for } \|v\|_2 \leqslant 1 \\ v/\|v\|_2 & \text{for } \|v\|_2 \geqslant 1 \end{cases}$$

is 2-Lipschitz. Therefore, our loss is still Lipschitz continuous with normalized embeddings in our framework.

B.2 PROOF FOR PROPOSITION 4.1

Proposition 4.1. *Given $\mathcal{S}=\{z_i\}_{i\in[m]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ such that $\forall k\in\mathcal{Y}$ $\{x_i|y_i=k\}$ is $\delta_{\mathcal{S}}$-cover[5] of $\mathcal{X}$, $\ell(z,z';\theta)$ is ζ^{ℓ}-Lipschitz in x, x' for all y, y' and θ, and bounded by L; then with probability at least $1-\gamma$,*

$$\left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\theta)] - \tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(z_i,z;\theta)]\right| \leqslant \mathcal{O}(\zeta^{\ell}\,\delta_{\mathcal{S}}) + \mathcal{O}(L\sqrt{\log\tfrac{1}{\gamma}/m}).$$

Proof. We start with defining $\hat{\mathcal{L}}(z;\theta) \coloneqq \mathbb{E}_{z'\sim p_{\mathcal{Z}}}[\ell(z,z';\theta)]$. Note that

$$\begin{aligned}\|\hat{\mathcal{L}}(z_1;\theta) - \hat{\mathcal{L}}(z_2;\theta)\|_2 &= |\,\mathbb{E}_{z'\sim p_{\mathcal{Z}}}[\ell(z_1,z';\theta)] - \mathbb{E}_{z'\sim p_{\mathcal{Z}}}[\ell(z_2,z';\theta)]| \\ &\leqslant \mathbb{E}_{z'\sim p_{\mathcal{Z}}}[|\ell(z_1,z';\theta) - \ell(z_2,z';\theta)|].\end{aligned}$$

Therefore, $\ell(z,z';\theta)$ being ζ^{ℓ}-Lipschitz in x for fixed x', y, y' and θ, and bounded by L implies $\hat{\mathcal{L}}(z;\theta)$ is also ζ^{ℓ}-Lipschitz in x for all y, θ and bounded by L. Hence, we have

$$|\hat{\mathcal{L}}(z_i;\theta) - \hat{\mathcal{L}}(z;\theta)| \leqslant \zeta^{\ell}\,\delta_{\mathcal{S}} \quad \forall z_i, z : z_i \in \mathcal{S}, z\in\mathcal{Z}, \|z_i - z\|_2 \leqslant \delta_{\mathcal{S}}$$

From Theorem 14 of Xu & Mannor (2012), we can partition $\mathcal{Z}$ into $K = \min_t\{|t| : t \text{ is } \frac{\delta_{\mathcal{S}}}{2}\text{-cover of } \mathcal{Z}\}$ disjoint sets, denoted as $\{\mathcal{R}_i\}_{i\in[K]}$, such that $\forall i : z_i \in \delta_{\mathcal{S}}$; both z_i, z being $\in \mathcal{R}_i$ implies $|\hat{\mathcal{L}}(z_i;\theta) - \hat{\mathcal{L}}(z;\theta)| \leqslant \zeta^{\ell}\,\delta_{\mathcal{S}}$. Hence, from Theorem 3 of Xu & Mannor (2012), with probability at least $1-\gamma$, we have:

$$\begin{aligned}\left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\theta)] - \tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell(z_i,z;\theta)]\right| &= \left|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\hat{\mathcal{L}}(z;\theta)] - \tfrac{1}{m}\sum_{i\in[m]}\hat{\mathcal{L}}(z_i;\theta)\right| \\ &\leqslant \zeta^{\ell}\,\delta_{\mathcal{S}} + L\sqrt{\frac{2\,K\,\log 2 + 2\,\log 1/\gamma}{m}}\end{aligned}$$

Note that K is dependent on δ_s and satisfies $\lim_{m\to\infty}\frac{K}{m}\to 0$ ensuring that the right hand side goes to zero as more samples are exploited and the covering radius is improved. Thus, asymptotically the following holds:

$$\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}[\ell(z,z';\theta^*)] \leqslant \mathcal{O}(\delta_s) + \mathcal{O}(\sqrt{\log\tfrac{1}{\gamma}/m}) \text{ with probability at least } 1-\gamma\,.$$

□

B.3 PROOF FOR PROPOSITION 4.2

Proposition 4.2. *Given $\{z_i\}_{i\in[n]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ and a set $s\subset[n]$. If $s = \cup_k s'_k$ with s'_k is the δ_s-cover of $\{i\in[n] \mid y_i = k\}$ (i.e., the samples in class k), $\ell(z,z';\theta)$ is ζ^{ℓ}-Lipschitz in x, x' for all y, y' and θ, and bounded by L, $e(\mathcal{A}_{s\times[n]})$ training error; then with probability at least $1-\gamma$ we have:*

$$\left|\tfrac{1}{n^2}\sum_{i,j\in[n]\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]}) - \tfrac{1}{|s|\,n}\sum_{i,j\in s\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right| \leqslant \mathcal{O}(\zeta^{\ell}\,\delta_s) + \mathcal{O}(e(\mathcal{A}_{s\times[n]})) + \mathcal{O}(L\sqrt{\log\tfrac{1}{\gamma}/n})$$

Proof. We are given a condition on s that we can partition $\mathcal{Z}$ into $m = |s|$ disjoint sets such that any sample from the dataset $(x_i, c), i\in[n]$, has a corresponding sample from s, $(x'_j, c), j\in s$ within δ_s ball. Thus, we start with partitioning $\mathcal{Z}$ into s disjoint sets as $\mathcal{Z} = \cup_i\mathcal{S}_i$ with $\mathcal{S}_i\cap\mathcal{S}_j = \emptyset$, $\forall i\neq j$.

We define $\ell_{[n]}(z) = \frac{1}{n}\sum_{i\in[n]}\ell(z,z_i,\mathcal{A}_{s\times[n]})$ and $\ell_s(z) = \frac{1}{m}\sum_{i\in s}\ell(z,z_i,\mathcal{A}_{s\times[n]})$ for the sake of clarity. Hence, we are interested in bounding $|\frac{1}{n}\sum_{[n]}\ell_{[n]}(z_i) - \frac{1}{m}\sum_s\ell_{[n]}(z_i)|$. We proceed with using triangle inequality to write:

$$\begin{aligned}&\left|\tfrac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i) - \tfrac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right| \\ &\qquad\leqslant \underbrace{\left|\tfrac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i) - \sum_{i\in s}\tfrac{n_i}{n}\ell_{[n]}(z_i)\right|}_{(T1)} + \underbrace{\left|\sum_{i\in s}\tfrac{n_i}{n}\ell_{[n]}(z_i) - \tfrac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right|}_{(T2)}\end{aligned}$$

[5] $\mathcal{S}\subset\mathcal{S}'$ is $\delta_{\mathcal{S}}$-cover of $\mathcal{S}'$ if $\forall z'\in\mathcal{S}'$, $\exists z\in\mathcal{S}$ such that $\|z - z'\|_2 \leqslant \delta_{\mathcal{S}}$.

For term $(T1)$ we write:

$$(T1) \leqslant \frac{1}{n} \sum_{i\in[m]} \sum_{z_j\in\mathcal{S}_i} |\ell_{[n]}(z_{s(i)}) - \ell_{[n]}(z_j)| \overset{(1)}{\leqslant} \zeta^\ell \, \delta_s$$

where in (1), we use ζ^ℓ-Lipschitz of the loss function and the condition $|z_{s(i)} - z_j| \leqslant \delta_s$, $\forall z_j \in \mathcal{S}_i$.

Using triangle inequality, we bound term $(T2)$ as:

$$\Big|\sum_{i\in s} \frac{n_i}{n}\ell_{[n]}(z_i) - \frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\Big| \leqslant \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_s(z)] - \mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)]\Big|^{(T2.1)}$$
$$+ \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)] - \sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)\Big|^{(T2.2)} + \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_s(z)] - \frac{1}{n}\sum_{i\in[n]}\ell_s(z_i)\Big|^{(T2.3)}$$

where we use $\frac{1}{m}\sum_s \ell_{[n]}(z_i) = \frac{1}{n}\sum_{[n]}\ell_s(z_i)$ in $(T2.3)$.

For $(T2.1)$ we have:

$$(T2.1) \leqslant \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\frac{1}{m}\sum_{i\in s}\ell(z_i,z) - \frac{1}{n}\sum_{i\in[n]}\ell(z_i,z)]\Big|$$

where we abuse the notation for the sake of clarity and drop parameter, $\mathcal{A}_{s\times[n]}$, dependency from the loss. Rearranging the terms, we have:

$$(T2.1) \leqslant \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\frac{1}{m}\sum_{i\in[m]}\frac{n-m\,n_i}{n}\ell(z_{s(i)},z)]\Big| + \Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\frac{1}{n}\sum_{i\in[m]}\sum_{j\in\mathcal{S}_i}\ell(z_{s(i)},z) - \ell(z_j,z)]\Big|$$

where similar to $(T1)$, the second summand is upper bounded by $\zeta^\ell\,\delta_s$. Using triangle inequality for the first summand, we write:

$$\Big|\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\frac{1}{m}\sum_{i\in[m]}\frac{n-m\,n_i}{n}\ell(z_{s(i)},z)]\Big| \leqslant (T2.3) + e(\mathcal{A}_{s\times[n]})$$

Hence, we have:

$$(T2.1) \leqslant \zeta^\ell\,\delta_s + (T2.3) + e(\mathcal{A}_{s\times[n]})$$

where from Hoeffding's Bound, $(T2.3) \leqslant L\sqrt{\log\frac{1}{\gamma}/2n}$ with probability at least $1-\gamma$:

Finally, we express $(T2.2)$ as:

$$\begin{aligned}(T2.2) &= \Big|\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)\mid z\in\mathcal{S}_i]p(z\in\mathcal{S}_i) - \sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)\Big| \\ &\leqslant \Big|\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)\mid z\in\mathcal{S}_i]\frac{n_i}{n} - \sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)\Big| \\ &+ \Big|\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)\mid z\in\mathcal{S}_i]p(z\in\mathcal{S}_i) - \sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}[\ell_{[n]}(z)\mid z\in\mathcal{S}_i]\frac{n_i}{n}\Big|\end{aligned}$$

Rearranging the terms we have:

$$(T2.2) \leqslant \sum_{i\in[m]}\frac{n_i}{n}\max_{z\in\mathcal{S}_i}|\ell_{[n]}(z) - \ell_{[n]}(z_{s(i)})| + \max_{z\in\mathcal{Z}}|\ell_{[n]}(z)|\sum_{i\in[m]}\Big|\frac{n_i}{n} - p(z\in\mathcal{Z})\Big|$$

where the first summand is bounded above by $\zeta^\ell\,(\delta_s + \varepsilon(n))$ owing to loss being ζ^ℓ-Lipschitz. Here, we denote $\varepsilon(n)$ as the covering radius of $\mathcal{Z}$, *i.e.*, the dataset, $\{x_i, y_i\}_{[n]}$ is $\varepsilon(n)$-cover of $\mathcal{X}\times\mathcal{Y}$. We note that $(n_i)_{i\in[m]}$ is an $i.i.d.$ multinomial random variable with parameters n and $(p_{\mathcal{Z}}(z\in\mathcal{S}_i))_{i\in[m]}$. Thus, by the Breteganolle-Huber-Carol inequality (Proposition A6.6 of Van Der Vaart et al. (2013)), we have :

$$(T2.2) \leqslant \zeta^\ell\,(\delta_s + \varepsilon(n)) + L\sqrt{\frac{2m\log 2 + 2\log 1/\gamma}{n}}$$

Finally, with probability at least $1-\gamma$, we end up with:

$$\Big|\frac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i) - \frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\Big| \leqslant \zeta^\ell\,(3\,\delta_s + \varepsilon(n)) + e(\mathcal{A}_{s\times[n]}) + L\,(\sqrt{\log\frac{1}{\gamma}/2n} + \sqrt{\frac{2m\log 2 + 2\log 1/\gamma}{n}})$$

□

Corollary 4.2.1. *Generalization performance of the proxy-based methods can be limited by the maximum of distances between the proxies and the corresponding class samples in the dataset.*

Proof. The covering radius for each class subset is the maximum distance between the corresponding class samples and the class proxy. We at least know that the generalization error is bounded above with a term proportional to that distance. □