Variational Continual Proxy-Anchor for Deep Metric Learning

Minyoung Kim
Samsung AI Center
Cambridge, UK

Ricardo Guerrero
Samsung AI Center
Cambridge, UK

Hai X. Pham
Samsung AI Center
Cambridge, UK

Vladimir Pavlovic
Rutgers University
New Jersey, USA

Abstract

The recent proxy-anchor method achieved outstanding performance in deep metric learning, which can be acknowledged to its data efficient loss based on hard example mining, as well as far lower sampling complexity than pair-based approaches. In this paper we extend the proxy-anchor method by posing it within the continual learning framework, motivated from its batch-expected loss form (instead of instance-expected, typical in deep learning), which can potentially incur the catastrophic forgetting of historic batches. By regarding each batch as a task in continual learning, we adopt the Bayesian variational continual learning approach to derive a novel loss function. Interestingly the resulting loss has two key modifications to the original proxy-anchor loss: i) we inject noise to the proxies when optimizing the proxy-anchor loss, and ii) we encourage momentum update to avoid abrupt model changes. As a result, the learned model achieves higher test accuracy than proxy-anchor due to the robustness to noise in data (through model perturbation during training), and the reduced batch forgetting effect. We demonstrate the improved results on several benchmark datasets.

1 INTRODUCTION

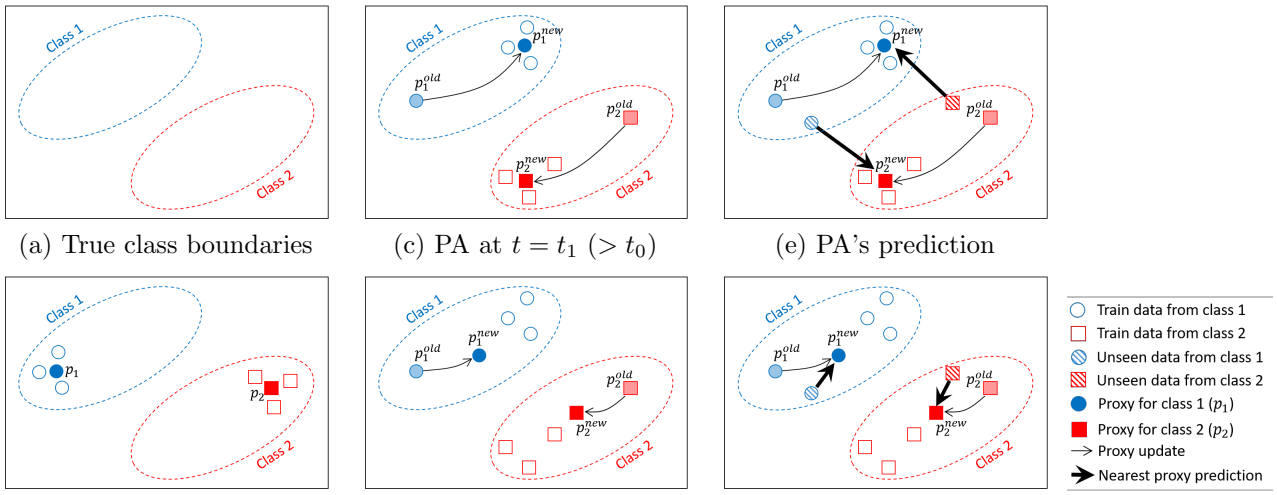
Deep metric learning (DML) is the task of learning a metric (similarity or distance measure) between two data points (e.g., images), which can be useful for numerous downstream computer vision tasks including image retrieval (Sohn, 2016; Song et al., 2016b; Movshovitz-Attias et al., 2017), few-shot learning (Qiao et al., 2019; Sung et al., 2018), face verification (Schroff et al., 2015; Liu et al., 2017), among others. Although metric learning has been a relatively well-studied problem in machine learning for the last decades (Shalev-Shwartz et al., 2004; Chopra et al., 2005; Goldberger et al., 2005; Weinberger and Saul, 2009), DML was recently resurrected with great success due to the advance and popularity of deep learning.

In most DML approaches, learning a metric between two data points x and x' typically boils down to learning an embedding function $f : \mathcal{X} \to \mathbb{R}^d$, where the similarity is defined as a dot product (or angle) between $f(x)$ and $f(x')$ in the d-dimensional embedded space. The typical setup (which we follow in this paper as well) is that training data consist of labeled data points (x, y), where the labels come from a predefined class label set $C(\ni y)$, which is disjoint from the class labels of the test points. That is, we need to exploit the supervision so that the embedding function learns the intrinsic semantic (dis)similarity between data points that is generalizable to unseen class labels.

Recent DML approaches can be broadly categorized into two schools: *pair-based* and *proxy-based*. The former is based on the intuitive idea of contrastive learning (Chopra et al., 2005; Hadsell et al., 2006; Wang et al., 2014; Schroff et al., 2015; Sohn, 2016; Song et al., 2016b; Wu et al., 2017; Harwood et al., 2017; Yuan et al., 2017; Wang et al., 2019b), where the instances of the same (different) classes should be pulled together (pushed away from each other, resp.). Proxy-based methods on the other hand introduce learnable proxy vectors in the embedded space which allow the loss function to be defined only in terms of distances between proxy vectors and data instances (Qian et al., 2019; Movshovitz-Attias et al., 2017; Kim et al., 2020).

In this paper, we seek for further improvement over the latest proxy-based method called the *proxy-anchor* method (Kim et al., 2020) specifically, which showed the pronounced performance among the state-of-the-arts. In particular, we closely inspect the proxy-anchor loss function, and interpret it as a batch-based loss

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.



(a) True class boundaries (c) PA at $t = t_1$ ($> t_0$) (e) PA's prediction

(b) PA and VCPA at $t = t_0$ (d) VCPA at $t = t_1$ ($> t_0$) (f) VCPA's prediction

Figure 1: Illustration of momentum update for proxies (KL term in (12)). (a) *True* class boundaries for two classes 1 and 2. (b) For the first batch data at $t = t_0$, shown as unfilled circles (class 1) and squares (class 2), both proxy-anchor (Kim et al., 2020) (denoted by PA) and our approach (VCPA) update the proxies toward the empirical class centers of the batch data (filled circle p_1 and square p_2). (c) For batch at $t = t_1$ ($> t_0$), PA updates the proxies toward class centers of the new batch data (without momentum) since it ignores the previous proxies ($p^{old} \rightarrow p^{new}$). (d) On the other hand, VCPA regularizes the proxy update by momentum so that p^{new}'s are not far away from p^{old}'s. (e-f) As a result, for unseen data points (circle/square filled with diagonal pattern), PA fails to find the correct proxies while VCPA succeeds, indicated by the thick-arrow nearest prediction.

in the sense that the loss is not decomposed instance-wisely, but rather dependent on the samples in a batch in a complex way. Hence there is a potential issue of batch forgetting, namely that the current batch is rarely revisited in the future, making the model forget useful information from historic batches.

This batch forgetting issue is analogous to the well-known catastrophic forgetting in continual learning (Kirkpatrick et al., 2017; Zenke et al., 2017; Serra et al., 2018; Nguyen et al., 2018; Ebrahimi et al., 2020; Benjamin et al., 2019; Pan et al., 2020). To this end, we frame the proxy-anchor loss optimization within continual learning. We regard each batch as a *task* in continual learning, and adopt the variational continual learning approach (Nguyen et al., 2018), which leads to a novel loss function with intuitive interpretation. Specifically, we end up with two modifications to the proxy-anchor loss: Random perturbation of proxy vectors, and momentum update of the variational parameters of the proxies. The former is beneficial for producing a robust model (less sensitive to the unknown noise or domain shift at test time), while the latter is useful for preventing the model from forgetting useful information from historic batches. These two core concepts are illustrated in diagrams in Fig. 1 (momentum update) and Fig. 2 (stochastic proxies).

We evaluate the proposed variational continual proxy-anchor (VCPA) method on the standard benchmark datasets (CUB, Cars, In-Shop, and SOP datasets), and achieve significant improvements over the proxy-anchor. Additionally, our approach exhibits less sensitivity to small batch sizes (due to the momentum update of the variational proxies) and is more generalizable when small amounts of training data are used (due to the stochastic sampling of the proxies).

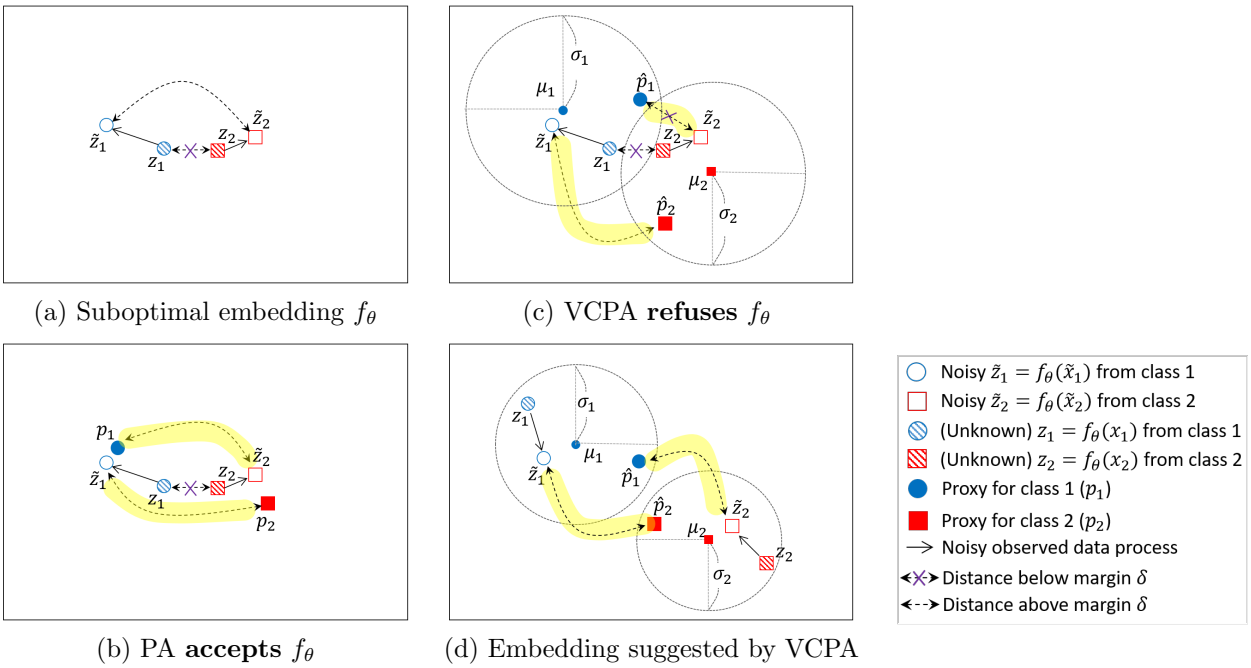
2 OUR APPROACH

We begin with introducing notations. We let x be a data instance (e.g., image), $z = f_\theta(x) \in \mathbb{R}^d$ be the d-dimensional embedded vector for x (the output of the embedding network f with weight parameters θ), and $p \in \mathbb{R}^d$ be a proxy vector (Sec. 2.1 for details). In proxy-based methods, only the similarity score between data instance and proxy vector is required during training, where the similarity between image x and proxy p is defined as the cosine similarity,

$$s(x, p) = \frac{z^\top p}{||z|| \cdot ||p||}, \quad z = f(x). \tag{1}$$

2.1 Proxy-Anchor Method Revisited

The recent proxy-anchor (PA) method (Kim et al., 2020) aims to achieve the goal of so-called *proxy-centered separation of data samples*, where each proxy is



(a) Suboptimal embedding f_θ (c) VCPA **refuses** f_θ

(b) PA **accepts** f_θ (d) Embedding suggested by VCPA

Figure 2: Illustration of stochastic proxies of the proposed VCPA. Suppose that we have noisy observation $\tilde{x}_1$ for the true (but unknown) x_1 from class 1 ($\tilde{x}_2$ and x_2 similarly). (a) The embedding network f_θ is suboptimal in that $z_1 = f_\theta(x_1)$ and $z_2 = f_\theta(x_2)$ are too close (distance below the margin δ), although the embeddings $\tilde{z}_1$ and $\tilde{z}_2$ of noisy data are distant enough (above the margin). (b) However, PA would accept f_θ since the distance between proxy p_1 and $\tilde{z}_2$ is large enough (above the margin), and similarly for p_2 and $\tilde{z}_1$ (highlighted as yellow). (c) On the other hand, VCPA refuses this embedding since the sampled proxy $\hat{p}_1 \sim q(p_1) = \mathcal{N}(p_1; \mu_1, \sigma_1^2)$ (shown in the circle with center μ_1 and radius σ_1) is too close to $\tilde{z}_2$, violating the margin constraint. (d) To meet the VCPA's constraint, both the embedding network and the variational proxy density $q(p_1) = \mathcal{N}(p_1; \mu_1, \sigma_1^2)$ (and also $q(p_2)$) need to be adjusted such a way that the sampled proxy $\hat{p}_1$ is distant from $\tilde{z}_2$ with large margin (similarly for $\hat{p}_2$ and $\tilde{z}_1$), as suggested in the layout.

a representative for each class. This specifically means that for each proxy p_j for the class $j \in C$, the similarity $s(x, p_j)$ between the data point x and p_j should be large (e.g., greater than some margin $\delta = 0.1$) if x belongs to the class j (denoted as $x \in D_j^+$), whereas the similarity needs to be small (e.g., less than $-\delta$) if x belongs to a different class ($x \in D_j^-$). This desideratum can be expressed succinctly in the follow (full-batch) loss:

$$\mathcal{L}_{PA}^{full} = \frac{1}{|C|} \sum_{j \in C} \left\{ \log \left(1 + \sum_{x \in D_j^+} e^{\alpha(\delta - s(x,p_j))} \right) + \log \left(1 + \sum_{x \in D_j^-} e^{\alpha(s(x,p_j)+\delta)} \right) \right\}. \quad (2)$$

Note that (2) is the *full-batch* version of the loss function, in which D_j^+ consists of *entire training data points* that have class label j (D_j^- defined similarly).

Since dealing with the full-batch data D_j^+/D_j^- is infeasible, in (Kim et al., 2020) they instead used minibatch versions, namely B_j^+, the positive (class j) samples *in a minibatch* B, and B_j^- likewise. And it boils down to the proxy-anchor loss:

$$\mathcal{L}_{PA} = \mathbb{E}_B[\mathcal{L}_{PA}(B)] \quad \text{where} \quad (3)$$

$$\mathcal{L}_{PA}(B) = \frac{1}{|C|} \sum_{j \in C} \left\{ \log \left(1 + \sum_{x \in B_j^+} e^{\alpha(\delta - s(x,p_j))} \right) + \log \left(1 + \sum_{x \in B_j^-} e^{\alpha(s(x,p_j)+\delta)} \right) \right\}, \quad (4)$$

where the expectation is taken over the i.i.d. random batch set B, sampled from the training data. In (Kim et al., 2020) they performed the stochastic gradient descent (SGD) optimization of (3); that is, sample a batch B, and update the embedding network and proxies by the gradient of $\mathcal{L}_{PA}(B)$. Despite the fact that the expectation of each log term in (3-4) cannot be an unbiased estimator for the corresponding term in (2), the proxy-anchor method showed significant improvement over the existing DML approaches. In this paper, we seek for further improvement over the proxy-anchor method by modifying the optimization strategy for the loss (3).

Our key motivation is that the loss (3) has the form of *expectation-over-batches*, apart from the conventional *expectation-over-instances* (i.e., $\mathbb{E}_x$) that arises in most machine learning and computer vision problems. In the latter, when we use a minibatch, we essentially perform multi-sample Monte-Carlo estimation, which can reduce the variance of the gradient estimate, and hence facilitates fast convergence to a local optimum. On the other hand, the batch-expected loss in (3) rather treats a batch B (as a whole) as a single instance. In this view, the optimization used in (Kim et al., 2020) can be seen as an SGD optimization with a *single-instance minibatch*; i.e., for each gradient update, we do not sample multiple different B's from data, but just one B. Although this still admits an unbiased estimate for (3), the sampling complexity becomes high since there is little chance that a current batch B is revisited in the future[1]. And the consequence is that the model may easily forget useful information from the historic batches in the course of learning, and be biased towards the latest batches.

To address this issue, one can treat instances as a set, such as the Neural Statistician (Edwards and Storkey, 2017), where we sample a minibatch of batches B's. However, this becomes computationally infeasible due to the large GPU memory requirement. Under this scenario, if batch B is understood as a *task*, then this can be recognized as *catastrophic forgetting* in continual learning (Kirkpatrick et al., 2017; Zenke et al., 2017; Serra et al., 2018; Nguyen et al., 2018; Ebrahimi et al., 2020; Benjamin et al., 2019; Pan et al., 2020), although the latter typically deals with a *sequence of datasets/tasks* instead of a *sequence of minibatches*. Our motivation here is that we can apply continual learning approaches to the loss of (3) to attenuate the aforementioned issue of the minibatch approximation.

2.2 Continual Learning Approach

If we follow the SGD learning, we are given a sequence of batches $B_1, B_2, \dots$ sampled from the training data, and update the model for each B_t using the loss function (3). Due to the aforementioned sampling complexity, it is easy for the model to forget the information from the historic batches. We need a mechanism to prevent the model from forgetting the previous batches, and come up with the learning strategy that can make the model perform well on *all* historic batches. If we regard the *batches* as *tasks*, this resembles the issues of catastrophic forgetting in the continual learning, whose goal is to make the model that performs well on *all* historic tasks (*batches* in our case).

Motivated from the probabilistic continual learning approaches (Kirkpatrick et al., 2017; Nguyen et al., 2018), we adopt a Bayesian framework. We first impose a prior distribution on the proxy vectors[2] $P = \{p_j\}_{j\in C}$. We adopt the factorized Gaussian as our prior,

$$p(P) = \prod_{j\in C} \mathcal{N}(p_j; 0, I). \tag{5}$$

Then the observed sequence of batches $B_1, B_2, \dots$, are presumed to be i.i.d. samples given the model, i.e., $p_\theta(B|P)$, which we define as:

$$p_\theta(B|P) = \exp(-\mathcal{L}_{PA}(B)/\tau), \tag{6}$$

where $\mathcal{L}_{PA}(B)$ is the proxy-anchor loss (4) and τ is a scaling (temperature) parameter. We use the subscript θ to indicate dependency on θ.

Preventing the model (i.e., the proxies P) from forgetting historic batches in the course of training, can be naturally and effectively carried out by posterior inference in the Bayesian framework. That is, we infer $p_\theta(P|B_1, B_2, \dots)$. Using the Bayes rule, this can be done recursively: at the current batch iteration t (once we observe B_t),

$$p_\theta(P|B_1 \cdots B_t) \propto p_\theta(P|B_1 \cdots B_{t-1}) \cdot p_\theta(B_t|P). \tag{7}$$

Note that the density normalizer in (7), denoted by $Z_t(\theta) = \int p_\theta(P|B_1 \cdots B_{t-1}) p_\theta(B_t|P) dP$, depends only on the data $B_1, \dots, B_t$, given θ. Initially at $t = 0$ (before we observe the first batch B_1), the posterior coincides with the prior $p(P)$ in (5).

Due to the complex dependency of the loss $\mathcal{L}_{PA}$ on P in (6), the normalizing constant $Z_t(\theta)$ cannot be computed exactly. Similar to (Nguyen et al., 2018) in the continual learning, we adopt the variational approximation. We denote by $q_t(P)$ the Gaussian approximation of the posterior at the t-th step, that is,

$$p_\theta(P|B_1 \cdots B_t) \approx q_t(P) \stackrel{\text{def}}{=} \prod_{j\in C} \mathcal{N}(p_j; \mu_j^t, \mathrm{Dg}(\sigma_j^t)^2),$$

where $\mathrm{Dg}(a)$ is the diagonal matrix with vector a on the diagonal, and (μ^t, σ^t) (with abbreviation $\mu^t := \{\mu_j^t\}_{j\in C}$ and $\sigma^t := \{\sigma_j^t\}_{j\in C}$) are the variational parameters for $q_t(P)$. At $t = 0$, we have q_0 coincide with the

[1] Specifically, the probability of seeing the same batch again is $1/\binom{N}{|B|}$ where N is the training set size, and $|B|$ is the batch size. With $N = 10,000$ and batch size 100, the probability becomes less than 10^{-200}.

[2] Although we can impose a prior on both embedding network parameters θ and P, treating only one of them as random variate, and the other as deterministic, is sufficient. It is because the two variables appear in the loss function as an interactive form only through $s(x, p)$, and the continual learning (or reduced forgetting) effect endowed in one variable naturally propagates to the other accordingly by alignment/misalignment. Treating P as random variate is more attractive in terms of computational cost than θ.

prior, that is, $\mu_j^0 =$ all-0 vector and $\sigma_j^0 =$ all-1 vector for all $j \in C$.

Now, we minimize the KL divergence between the posterior (7) and the variational $q_t(P)$. By approximating $p_\theta(P|B_1 \cdots B_{t-1})$ in (7) with $q_{t-1}(P)$, we can derive the KL divergence as follows:

$$\begin{aligned} &\mathrm{KL}(q_t(P) \,||\, p_\theta(P|B_1 \cdots B_t)) && (8)\\ &\approx \mathrm{KL}(q_t(P) \,||\, q_{t-1}(P) \cdot p_\theta(B_t|P) \,/\, Z_t(\theta)) && (9)\\ &= \mathbb{E}_{q_t(P)}\left[\log \frac{Z_t(\theta) \cdot q_t(P)}{q_{t-1}(P) \cdot e^{-\mathcal{L}_{PA}(B_t)/\tau}}\right] && (10)\\ &= \mathrm{KL}(q_t||q_{t-1}) + \frac{1}{\tau}\mathbb{E}_{q_t}[\mathcal{L}_{PA}(B_t)] + \log Z_t(\theta). && (11)\end{aligned}$$

As $Z_t(\theta)$ does not depend on the variational parameters of q_t, the variational approximation at the t-th step becomes:

$$\min_{\mu^t,\sigma^t} \; \mathcal{L}_t := \mathrm{KL}(q_t||q_{t-1}) + \frac{1}{\tau}\mathbb{E}_{q_t(P)}[\mathcal{L}_{PA}(B_t)]. \quad (12)$$

Note that the KL term in (12) admits a closed form (Supplement for details). The second term is the expected proxy-anchor loss with respect to the variational density $q_t(P)$. As widely adopted, this can be approximated by the one-sample Monte-Carlo estimate with the reparametrization trick (Kingma and Welling, 2014). That is, we sample $\hat{P} = \mu^t + \sigma^t \bullet \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$, and evaluate the loss $\mathcal{L}_{PA}(B_t)$ with $\hat{P}$.

Looking closely at (12), there are two modifications to the proxy-anchor method of (Kim et al., 2020). First, we use noisy perturbed proxies $\hat{P}$ when evaluating the proxy-anchor loss. This can help making the model more robust to noise since the proxies and embedded data points get aligned to each other with larger margin, making small perturbation in the embedded data points less influencing. And this increased margin can lead to better generalization (e.g., viable to small training data) (Vapnik, 1998; Grandvalet and Bengio, 2004). Secondly, the KL term can be seen as momentum (proximal) regularization, which discourages abrupt proxy changes from the previous iterates, useful for keeping historic information and reducing catastrophic forgetting. Along this line, the proposed model can be affected less by small batch sizes (Sec. 4 for related empirical results).

To optimize (12), although one can follow several (first-order) gradient descent, we realize that $\mathcal{L}_t$ is *nearly* a convex function of (μ^t, σ^t): the KL is convex in both parameters, and $\mathcal{L}_{PA}(B_t)$ (4) has the form of log-sum-exp of $s(x, p)$ where $s(x, p)$ is linear in p (except for the normalization by $||p||$). Hence we can employ a second-order method (esp., the Newton update) which usually converges quickly to the optimum (Nocedal and Wright, 2006). To deal with the non-convexity originating from $||p||$, we regard it as constant during the gradient/Hessian computation. The (approximate) gradient/Hessian of $\mathcal{L}_t$ admits closed forms (Supplement for details). Then we take M steps (e.g., $M = 10$) of the Newton update:

$$\mu^t \leftarrow \mu^t - H_{\mu^t}^{-1} g_{\mu^t}, \quad \sigma^t \leftarrow \sigma^t - H_{\sigma^t}^{-1} g_{\sigma^t}, \qquad (13)$$

where H and g are approximate Hessian and gradient of $\mathcal{L}_t$, respectively. For tractability, H is diagonalized.

Since the objective $\mathcal{L}_t$ of (12) is an upper bound (negative ELBO) of the negative conditional log-likelihood, i.e., $\mathcal{L}_t \geq -\log Z_t(\theta) = -\log p_\theta(B_t|B_1 \cdots B_{t-1})$, we can use it to update the model θ. Once the Newton optimization of (12) with respect to (μ^t, σ^t) is done, we fix (μ^t, σ^t) and update θ by gradient descent with $\mathcal{L}_t$, which boils down to the second term $\mathcal{L}_{PA}(B_t)$ evaluated on $\hat{P}$. Our method, called **Variational Continual Proxy Anchor** (**VCPA**), can be described as pseudo codes[3] in Alg. 1. Note that we use the clamping operation after each update of σ^t to keep it positive. We usually set the number of Newton steps $M = 10$.

Algorithm 1 Variational Continual Proxy Anchor.

Input: Scale τ, learning rate γ, # Newton steps M.
Initialize: θ (e.g., pre-trained model), $\mu^0 = 0, \sigma^0 = 1$.
for $t = 1, 2, \ldots$ **do**
  Sample a batch $B_t \sim$ data.
  Warm start: $(\mu^t, \sigma^t) \leftarrow (\mu^{t-1}, \sigma^{t-1})$.
  for $m = 1, \ldots, M$ **do**
    $\hat{P} = \mu^t + \sigma^t \bullet \epsilon, \;\; \epsilon \sim \mathcal{N}(0, I)$.
    Compute $g = \nabla\mathcal{L}_t$, $H = \nabla^2\mathcal{L}_t$ wrt (μ^t, σ^t).
    Newton update: $(\mu^t, \sigma^t) \leftarrow (\mu^t, \sigma^t) - H^{-1}g$.
    Clamp: $\sigma^t \leftarrow \max\{\sigma^t, \sigma_{min}\}$ (e.g., $\sigma_{min} = 10^{-5}$).
  end for
  Evaluate $\mathcal{L}_{PA}(B_t)$ with $\hat{P} = \mu^t + \sigma^t \bullet \epsilon, \;\; \epsilon \sim \mathcal{N}(0, I)$.
  Update: $\theta \leftarrow \theta - \gamma\nabla_\theta\mathcal{L}_{PA}(B_t)$.
end for

3 RELATED WORK

Deep metric learning approaches broadly fall into two schools: *pair-based* and *proxy-based*. Pair-based approaches commonly rely on the idea of contrastive learning (Chopra et al., 2005; Hadsell et al., 2006), where the instances of the same (different) classes should be pulled together (pushed away from each other, resp.). Beyond the intuitive triplet forms (Wang et al., 2014; Schroff et al., 2015), more sophisticated losses were introduced; in (Sohn, 2016; Song et al., 2016b) each anchor is contrasted with a positive and multiple negative pairs to take into account hard examples, while in (Wang et al., 2019a,b) every pair in a batch is incorporated into the loss with some importance weighting

[3] In the actual implementation, we use the re-scaled loss function, $\mathcal{L}_t := \tau\mathrm{KL}(q_t||q_{t-1}) + \mathbb{E}_{q_t(P)}[\mathcal{L}_{PA}(B_t)]$ for numerical stability.

schemes. Pair-based approaches inherently suffer from computational overhead as the sampling complexity becomes super-linear in the training data size. Although there have been attempts to focus on a small number of the most important samples in the batch, they typically resort to sophisticated sampling strategies (Wu et al., 2017; Harwood et al., 2017; Yuan et al., 2017; Wang et al., 2019b), which often involve difficult hyperparameter search.

Proxy-based methods reduced the computational complexity to linear by introducing the so-called proxy vectors in the embedded space. The proxies are learnable parameters, and typically serve as class representatives (e.g., class centers), and hence the loss function can be defined solely with distances between data instances and proxy vectors, without requiring pairwise distances, reducing the sampling complexity to linear. Whereas proxy-NCA (Movshovitz-Attias et al., 2017) builds a loss function based on the popular neighborhood component analysis (Goldberger et al., 2005), SoftTriple (Qian et al., 2019) derives the loss function by extending the softmax classification. More recently, proxy-anchor (Kim et al., 2020) addresses the data inefficiency issue of the proxy-NCA by incorporating data-to-data relations into the loss function.

Other recent approaches in DML focus on different aspects. In (Mohan et al., 2020), a direction regularization strategy was introduced to explicitly account for the layout of sampled pairs as well as orthogonality in the representations. In (Zheng et al., 2020) they aimed to learn a sample assessment strategy, via an episode-based meta learning algorithm, to maximize the generalization of the trained metric. While some approaches aim to generate synthetic data points for the purpose of augmentation and generalization (Zhao et al., 2018; Duan et al., 2018; Zheng et al., 2019), learning additional generative networks along with the main embedding network can be a considerable overhead. This issue was addressed in (Ko and Gu, 2020) by generating synthetic points from feature combination while keeping the augmentation information.

Continual learning emerged recently to address the drawback of the current deep learning methods where the models tend to quickly forget previously learned skills when learning a new task. The main goal of continual learning is to make the model perform well on *all* tasks it has seen during training. To achieve the goal, the most popular approach is to regularize the network weights to stay close to those learned from the previous tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Serra et al., 2018; Nguyen et al., 2018; Ebrahimi et al., 2020). Recently, there have been attempts to directly regularize the function outputs of the model instead of network weights, which often yields performance improvement at the expense of added computational overhead (Benjamin et al., 2019; Pan et al., 2020).

4 EXPERIMENTS

We empirically test our VCPA on the popular benchmark DML datasets (Sec. 4.1) following the experimental setups (Sec. 4.2). The comparison with the state-of-the-art DML methods is reported in Sec. 4.3. We also empirically highlight the two benefits of the proposed method, namely robustness to biased noisy data, esp., small training data (Sec. 4.4), and reduced effect of historic batch forgetting via small batch sizes (Sec. 4.5). In Sec. 4.6, we perform ablation study to show that both stochastic proxy sampling and the momentum proxy KL loss in our VCPA are important to achieve the highest performance. We also compare our VCPA with other popular momentum update approach (He et al., 2019), to show the effectiveness of our approach (Sec. 4.7).

4.1 Datasets

The proposed method is evaluated on CUB-200-2011 (Welinder et al., 2010), Cars-196 (Krause et al., 2013), Stanford Online Product (SOP) (Song et al., 2016a) and In-shop Clothes Retrieval (In-Shop) (Liu et al., 2016) datasets following the standard data split protocols as used in previous literature. For CUB-200-2011 the first 100 classes (5,864 images) are used for training, while the other 100 classes (5,924 images) for testing. For Cars-196, the first 98 classes (8,054) are used for training, while the remaining 98 classes are used for testing. In SOP the standard data splits as provided in (Song et al., 2016a; Kim et al., 2020) are used, which consist of 11,318 classes (59,551 images) for training and 11,316 classes (60,502 images) for testing. Finally, for In-Shop we use data splits as described in (Kim et al., 2020), mainly the first 3,997 classes (25,882 images) are used for training, while the remainder test classes 7970 (28,760 images) are split into query and gallery sets of 3,985 (14,218 images) and 3,985 classes (12,612 images), respectively.

4.2 Experimental Setups

For fair comparison with many of the existing DML methods, we use the BN-Inception backbone, which is initialized with the ImageNet pre-trained model. Also, the embedding dimension is chosen as 512, and a fully connected output head is attached to the backbone final layer. For the optimization, we follow the similar setups as (Kim et al., 2020) for the learning rates, schedules, number of epochs, and so on. The scaling and margin parameters in the PA loss are $\alpha = 32$,

$\delta = 0.1$. Moreover, the images are randomly cropped and resized to (224×224) pixels, as is standard.

4.3 Comparison with State-Of-The-Arts

Our main results are summarized in Table 1. We follow the standard protocols, and the scores of the competing methods are excerpted from their publications.

PA vs. VCPA: Statistical Significance. In our main results (Table 1), we see that VCPA performs the best among all methods for all datasets. To check the statistical significance, we take the two best models, PA and VCPA, and perform further statistical analysis. Since there exist a few sources of randomness in the training of PA and VCPA (e.g., location of initial proxies in PA and proxy sampling in VCPA), we perform multiple runs on the CUB and Cars datasets. Table 2 shows that VCPA outperforms PA with strong statistical significance on both datasets.

4.4 Impact of Small Training Data

We verify our claim that the stochastic sampling for the proxies in our VCPA can improve robustness to input noise or domain shift. To this end, we train our VCPA and PA with small training data by reducing the number of samples per class. This experiment can be useful to judge how flexible the models are adapted from the initial pre-trained model to a new domain. Recall that the pre-trained model is used for embedding network initialization, and obtained from the ImageNet domain training, different from the domains of the datasets (e.g., CUB) that we train our models on.

We reduce the training data sizes to $2 \sim 10\%$ of the original training sets. On CUB, they roughly correspond to $1 \sim 6$ training samples per class. These few training sets exhibit highly biased and noisy samples, adequate for testing the robustness of the models. The results are summarized in Fig. 3, where we also report the performances of the 1024-dim features directly extracted from the pre-trained BN-Inception model (IncepFeat) and the random projection head attached on top of the Inception features (RandHead). As shown, VCPA is more robust than PA. Looking at 10% reduction, we see that the Inception features alone (without head) attains $R@1 = 54.7$, and adding a random head makes it worse (50.2), but training with 10% data using PA makes it better (56.3), while training with our VCPA achieves the best score 58.6. Similar result was obtained for the Cars 10% reduced data: IncepFeat: 48.8, RandHead: 43.3, PA: 58.6, and VCPA: 59.5.

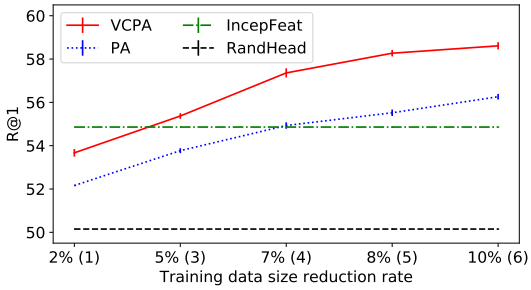


Figure 3: Small training data experiment. The parentheses in X axis contain numbers of samples per class.

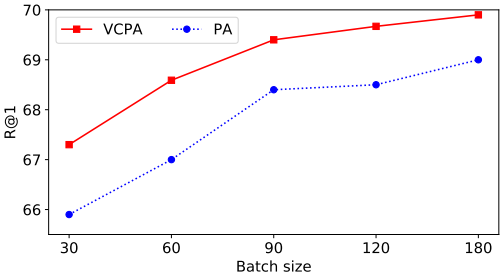


Figure 4: Small batch size experiment on CUB.

4.5 Impact of Small Batch Size

To show the effectiveness of the momentum update of the variational proxies in our VCPA, we conduct an experiment with small batch sizes. This is motivated from the intuition that with small batch size, the models are exposed to potentially more risky situation where historic batches are more easily forgotten. The result in Fig. 4 indicates that our VCPA is more robust to small batch sizes than PA.

4.6 Comparative & Ablation Study

Multiple Updates of Proxies in Proxy-Anchor. In our VCPA, we update proxies multiple ($M = 10$) times per batch by the Newton update. For a fair comparison we inspect what will happen if PA also does similar multiple proxy updates per batch. We perform two different update schemes with the same number of updates $M = 10$: the *first-order* update and the *second-order* Newton update. For the former, we use the AdamW (Loshchilov and Hutter, 2017) as is also employed for the main embedding network, while for the latter we follow the same approximate gradient/Hessian computation scheme that we proposed in our VCPA. The results on the CUB and Cars datasets are shown in Fig. 5. The Newton update leads to a better model than the first-order update for CUB, while it is the other way around for Cars. However, multiple update of the proxies does not help improving, but even deteriorates, the performance of the proxy-anchor method. This result indicates that the success of our

		CUB				Cars				SOP				In-Shop			
		R@1	2	4	8	R@1	2	4	8	R@1	10	100	1000	R@1	10	20	40
Margin^{128}	R50	63.6	74.4	83.1	90.0	79.6	86.5	91.9	95.1	72.7	86.2	93.8	98.0	–	–	–	–
HDC^{384}	G	53.6	65.7	77.0	85.6	73.7	83.2	89.5	93.8	69.5	84.4	92.8	97.7	–	–	–	–
A-BIER^{512}	G	57.5	68.7	78.3	86.2	82.0	89.0	93.2	96.1	74.2	86.9	94.0	97.8	83.1	95.1	96.9	97.8
ABE^{512}	G	60.6	71.5	79.8	87.4	85.2	90.5	94.0	96.1	76.3	88.4	94.8	98.2	87.3	96.7	97.9	98.5
HTL^{512}	BN	57.1	68.8	78.7	86.5	81.4	88.0	92.7	95.7	74.8	88.3	94.8	98.4	–	–	–	–
RLL-H^{512}	BN	57.4	69.7	79.2	86.9	74.0	83.6	90.1	94.1	76.1	89.1	95.4	–	–	–	–	–
MS^{512}	BN	65.7	77.0	86.3	91.2	84.1	90.4	94.0	96.5	78.2	90.5	96.0	98.7	89.7	97.9	98.5	**99.1**
SoftTri^{512}	BN	65.4	76.4	84.5	90.4	84.5	90.7	94.5	96.9	78.3	90.3	95.9	–	–	–	–	–
ALA-M^{512}	G	61.6	73.9	83.1	89.7	80.5	87.9	92.8	95.9	77.0	89.4	96.1	–	–	–	–	–
EE-MS^{512}	G	57.4	68.7	79.5	86.9	76.1	84.2	89.8	93.8	78.1	90.3	95.8	–	–	–	–	–
DR-MS^{512}	G	66.1	77.0	85.1	91.1	85.0	90.5	94.1	96.4	–	–	–	–	91.7	**98.1**	98.7	**99.1**
PA^{512}	BN	68.4	**79.2**	86.8	91.6	86.1	91.7	95.0	**97.3**	79.1	90.8	**96.2**	**98.7**	91.5	**98.1**	**98.8**	**99.1**
VCPA^{512}	BN	**69.9**	79.0	**86.9**	**92.2**	**86.9**	**92.3**	**95.2**	**97.3**	**79.5**	**91.0**	**96.2**	98.6	**92.2**	**98.1**	98.7	**99.1**

Table 1: Comparison on the benchmark datasets. Superscripts are embedding dimensions. The embedding backbone networks used are indicated in the second column, R50=ResNet-50 (He et al., 2016), BN=Inception with batch normalization (Ioffe and Szegedy, 2015), G=GoogleNet (Szegedy et al., 2015). Competing approaches are: Margin=(Wu et al., 2017), HDC=(Yuan et al., 2017), A-BIER=(Opitz et al., 2020), ABE=(Kim et al., 2018), HTL=(Ge et al., 2018), RLL-H=(Wang et al., 2019a), MS=(Wang et al., 2019b), SoftTri=(Qian et al., 2019), ALA-M=(Zheng et al., 2020), EE-MS=(Ko and Gu, 2020), DR-MS=(Mohan et al., 2020), PA=(Kim et al., 2020).

R@1	PA^{512} (Kim et al., 2020)	VCPA^{512}	p-value
CUB	68.04 ± 0.08	69.41 ± 0.11	$\ll 0.0001$
Cars	86.29 ± 0.14	86.58 ± 0.09	0.01

Table 2: Statistical analysis of R@1 in PA and VCPA. Mean, standard error and statistical significance (paired t-test), for CUB and Cars.

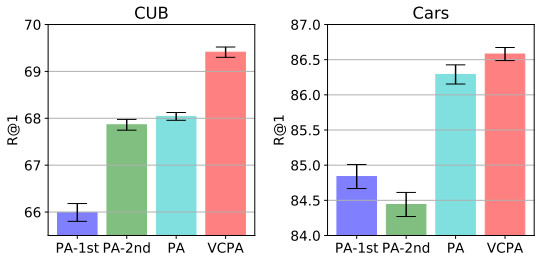


Figure 5: Multiple proxy updates per batch on CUB and Cars. PA-1st uses the first-order AdamW, while PA-2nd performs the (approximate) Newton update.

VCPA is not originated from merely over-optimizing the proxy parameters, but the sophisticated loss function derived from the Bayesian continual learning criteria.

Ablation I: Without Stochastic Sampling. In our variational learning we used the stochastic sampling $\hat{P} = \mu^t + \sigma^t \bullet \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ when we evaluate/back-prop the proxy-anchor loss $\mathcal{L}_{PA}(B_t)$. To see the impact of this stochastic sampling, we test our VCPA without sampling, by using deterministic $\hat{P} = \mu^t$ with $\epsilon = 0$. The results on the CUB and Cars datasets are shown in Fig. 6. Deterministic proxy significantly falls short of the VCPA with stochastic sampling, implying the importance of the stochastic treatment for robustness.

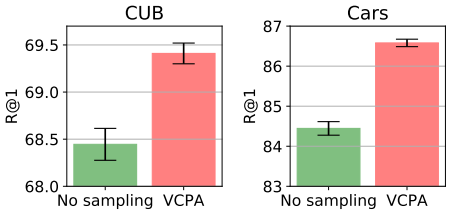


Figure 6: Without stochastic sampling in our VCPA.

Ablation II: Impact of Proxy KL Loss. The proxy KL loss term $\mathrm{KL}(q_t(P)||q_{t-1}(P))$ in our VCPA has the role of preventing the proxies from abrupt changes, crucial for avoiding historic batch forgetting issue. To see its impact, we perform the ablation study of reducing the impact of the KL term in the loss during training. Note that the scale τ in (12)[4] serves as the impact constant for the KL term, and we gradually reduce τ by one tenth from 10^3 down to 10^{-7}. The resulting test R@1 scores for the CUB and Cars datasets are summarized in Fig. 7. As shown reducing the impact of the KL term significantly degrades the performance.

4.7 Comparison to MoCo-like Model

As an alternative way of avoiding the historic batch forgetting issue, we consider carrying out momentum update for the embedding network f_θ as baseline comparison. To this end, we modify the original proxy-anchor method by adopting the idea from the recent MoCo (Momentum Contrast) approach (He et al., 2019), popular in self-supervised learning. More specifically, we maintain two embedding networks (of the same archi-

[4]Recall that we used the re-scaled version of the loss function (12), namely $\mathcal{L}_t := \tau \mathrm{KL}(q_t||q_{t-1}) + \mathbb{E}_{q_t(P)}[\mathcal{L}_{PA}(B_t)]$.

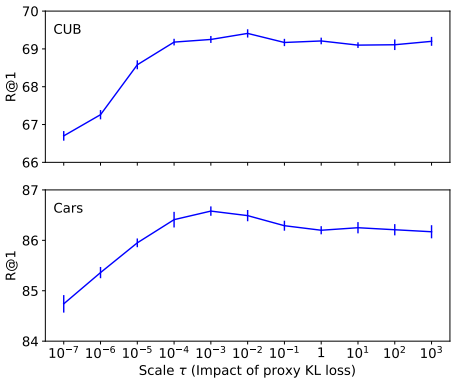


Figure 7: Impact of proxy KL loss. We vary the impact constant τ for the proxy KL loss term.

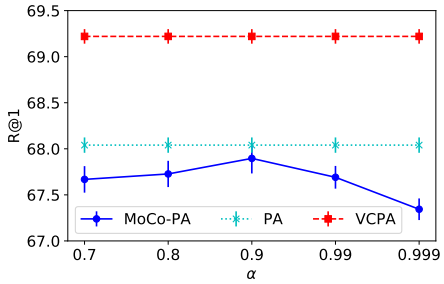


Figure 8: Comparison with MoCo-like momentum update scheme for proxy-anchor (denoted as MoCo-PA). We vary the momentum parameter α. Scores of PA and VCPA superimposed for comparison.

tecture); one is the main f_θ, and the other f_w is the momentum network (parameters w). The momentum network f_w smoothly follows the main network f_θ while memorizing the previous iterates. This is done by the momentum update of the network parameters,

$$w \leftarrow \alpha w + (1-\alpha)\theta, \tag{14}$$

with the momentum constant α (e.g., $\alpha = 0.999$). In MoCo, we also maintain a (FIFO) queue, of size typically far larger than minibatch size, which is filled with the latest batch data $\{(z = f_w(x), y)\}_{(x,y)\sim B}$ every iteration. At each iteration, we update θ by backprop, but we do stop-gradient for w. Hence one can harness a large number of samples from the queue, alongside a minibatch, to improve stochastic approximation of the original full-batch loss (2).

The results on different α are shown in Fig. 8. We set the queue size 900, five times the batch size 180, which deals with 1080 samples in the proxy-anchor loss function. However, as shown, this momentum update scheme does not improve the performance of PA, clearly falling short of both PA and VCPA. This implies that simply adopting off-the-shelf momentum update strategies may fail, and signifies the impact of the continual learning criteria employed in our VCPA to ameliorate the historic batch forgetting issue.

4.8 Computational Complexity

Computational overhead incurred by VCPA is mainly the M Newton update steps (against PA's one-step SGD). However, since the Newton update takes the same time complexity as gradient computation (Sec. A.2 in Supplement), the overall asymptotic complexity of PA and our VCPA is the same, whereas VCPA takes constant factor (M) more time than PA. The number of minibatches required in VCPA and PA is the same. We report the wall clock times; The per-batch (of size 180) training times are: **528 msec** (VCPA) and **383 msec** (PA) on a single GTX 2080ti machine.

5 CONCLUSION

In this paper we have proposed a novel variational continual proxy-anchor method for deep metric learning, to address the potential historic batch forgetting issue. With the Bayesian variational treatment, we derive a novel loss function that is also intuitively appealing and beneficial in two aspects: 1) random perturbation of proxy vectors to lead to a model robust to bias or domain shift in the data, and 2) momentum update of the variational proxies to reduce the batch forgetting effect. In the empirical study, the proposed model achieved the pronounced performance, outperforming the best state-of-the-arts with large margins, while the claimed benefits were well verified.

References

Ari S Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representation*, 2019.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Sayna Ebrahimi, Mohamed Elhoseiny, Marcus Rohrbach, and Trevor Darrell. Uncertainty guided continual learning with bayesian neural networks. In *International Conference on Learning Representation*, 2020.

Harrison Edwards and Amos Storkey. Towards a neural

statistician. In *International Conference on Learning Representations*, 2017.

Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. In *European Conference on Computer Vision (ECCV)*, 2018.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis, 2005. In Advances in Neural Information Processing Systems.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization, 2004. In Proc. of Advances in Neural Information Processing Systems.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

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

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes, 2014. In Proceedings of the Second International Conference on Learning Representations, ICLR.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Byungsoo Ko and Geonmo Gu. Embedding expansion: Augmentation in embedding space for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. doi: 10.1109/ICCVW.2013.77.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Noel Loo, Siddharth Swaroop, and Richard E. Turner. Generalized variational continual learning. In *arXiv preprint*, 2020. URL `https://arxiv.org/abs/2011.12328`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *arXiv preprint*, 2017. URL `https://arxiv.org/abs/1711.05101`.

Deen Dayal Mohan, Nishant Sankaran, Dennis Fedorishin, Srirangaraj Setlur, and Venu Govindaraju. Moving in the right direction: A regularization for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 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 (ICCV)*, 2017.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2): 276–290, 2020. doi: 10.1109/TPAMI.2018.2848925.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. *Advances in neural information processing systems*, 2020.

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

Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018.

S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *International Conference on Machine Learning*, 2004.

Kihyuk Sohn. Improved deep metric learning with multiclass n-pair loss objective, 2016. In Advances in Neural Information Processing Systems.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016a.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Jiang Wang, Yang Song, T. Leung, C. Rosenberg, Jingbin Wang, J. Philbin, Bo Chen, and Ying Wu. Learning finegrained image similarity with deep ranking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019b.

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

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

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 (ICCV)*, 2017.

Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

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

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

Wenzhao Zheng, Jiwen Lu, and Jie Zhou. Deep metric learning via adaptive learnable assessment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Supplementary Material: Variational Continual Proxy-Anchor for Deep Metric Learning

- Detailed derivations (Sec. A).
- Visualization of learned proxies and embeddings (Sec. B)
- Qualitative retrieval results (Sec. C).
- Hyperparameters and Additional Experiments (Sec. D).

A DETAILED DERIVATIONS

All mathematical operations in this section are element-wise.

A.1 Derivation of Proxy KL Loss

It follows from the KL divergence between two Gaussians that:

$$\mathrm{KL}(q_t(P) \mid\mid q_{t-1}(P)) = \sum_{j\in C} \mathrm{KL}(\mathcal{N}(\mu_j^t, \mathrm{Dg}(\sigma_j^t)^2) \mid\mid \mathcal{N}(\mu_j^{t-1}, \mathrm{Dg}(\sigma_j^{t-1})^2)) \tag{15}$$

$$= \sum_{j\in C} \frac{1}{2}\mathbb{1}^\top \left(\frac{(\sigma_j^t)^2}{(\sigma_j^{t-1})^2} + \frac{(\mu_j^t - \mu_j^{t-1})^2}{(\sigma_j^{t-1})^2} - 2\log\frac{\sigma_j^t}{\sigma_j^{t-1}} \right) + \text{const.}, \tag{16}$$

where $\mathbb{1}$ is the all-1 vector of the same dimension as the proxies.

A.2 Derivation of Gradients and Hessians for $\mathcal{L}_t$

We derive the gradients and Hessians, denoted by g_μ, g_σ and H_μ, H_σ, resp., of our VCPA loss function $\mathcal{L}_t$, which are required in the Newton update of the variational proxy parameters (μ, σ) of $q(P)$. Recall that the loss of our VCPA is (for the given embedding network θ):

$$\mathcal{L}_t(\mu^t, \sigma^t; \theta) \;=\; \mathrm{KL}(q_t(P) \mid\mid q_{t-1}(P)) + \frac{1}{\tau}\mathbb{E}_{q_t(P)}[\mathcal{L}_{PA}(B_t)], \quad \text{where} \tag{17}$$

$$\mathcal{L}_{PA}(B) = \frac{1}{|C|}\sum_{j\in C}\left\{ \log\left(1 + \sum_{x\in B_j^+} e^{\alpha(\delta - s(x,p_j))}\right) \;+\; \log\left(1 + \sum_{x\in B_j^-} e^{\alpha(s(x,p_j)+\delta)}\right)\right\}, \tag{18}$$

$$s(x,p) = \frac{z^\top p}{||z||\cdot||p||}, \quad z = f_\theta(x). \tag{19}$$

For the expectation term in (17), we use the Monte Carlo estimation with a reparametrized sample $\hat{P} \sim q_t(P)$,

$$\mathbb{E}_{q_t}[\mathcal{L}_{PA}(B)] \;\approx \frac{1}{|C|}\sum_{j\in C}\left\{ \log\left(1 + \sum_{x\in B_j^+} \exp\Big(\alpha\big(\delta - s(x,\hat{p}_j^t)\big)\Big)\right) \;+\; \log\left(1 + \sum_{x\in B_j^-} \exp\Big(\alpha\big(s(x,\hat{p}_j^t) + \delta\big)\Big)\right)\right\}, \tag{20}$$

where $\hat{p}_j^t = \mu_j^t + \sigma_j^t \bullet \epsilon_j$ with $\epsilon_j \sim \mathcal{N}(0, I)$ for $j \in C$.

The loss $\mathcal{L}_t$ in (17) is *nearly* a convex function of (μ^t, σ^t): the KL term in (16) is convex in both parameters, and (20) has the form of log-sum-exp of $s(x, \hat{p})$ where $s(x, \hat{p})$ is linear in $\hat{p}$, and hence linear in μ and σ, except for the normalization by $||\hat{p}||$. To deal with the non-convexity originating from $||\hat{p}||$, we regard it as constant during the gradient/Hessian computation. First,

$$\nabla_{\mu_j^t}\mathrm{KL} = \frac{\mu_j^t - \mu_j^{t-1}}{(\sigma_j^{t-1})^2}, \quad \nabla_{\sigma_j^t}\mathrm{KL} = \frac{\sigma_j^t}{(\sigma_j^{t-1})^2} - \frac{1}{\sigma_j^t}, \tag{21}$$

$$\nabla^2_{\mu_j^t}\mathrm{KL} = \frac{1}{(\sigma_j^{t-1})^2}, \qquad \nabla^2_{\sigma_j^t}\mathrm{KL} = \frac{1}{(\sigma_j^{t-1})^2} + \frac{1}{(\sigma_j^t)^2}. \tag{22}$$

Note that (22) ensures that Hessians are positive, thus KL being convex. For the proxy-anchor loss part, we first let:

$$h_j^+(x) := \exp\Big(\alpha\big(\delta - s(x, \hat{p}_j^t)\big)\Big), \quad h_j^-(x) := \exp\Big(\alpha\big(s(x, \hat{p}_j^t) + \delta\big)\Big), \quad \overline{f(x)} = \frac{f(x)}{||f(x)||}. \tag{23}$$

Then we first take the gradient with respect to $\hat{p}_j$ (superscript t dropped for simplicity in notation). By regarding $||\hat{p}_j||$ as constant, we have:

$$\nabla_{\hat{p}_j}\mathcal{L}_{PA} \approx \frac{\alpha}{|C|}\frac{1}{||\hat{p}_j||}\big(G_j^- - G_j^+\big), \quad \text{where} \quad G_j^+ = \frac{\sum_{x\in B_j^+} h_j^+(x)\overline{f(x)}}{1 + \sum_{x\in B_j^+} h_j^+(x)}, \quad G_j^- = \frac{\sum_{x\in B_j^-} h_j^-(x)\overline{f(x)}}{1 + \sum_{x\in B_j^-} h_j^-(x)}. \tag{24}$$

The Hessian with respect to $\hat{p}_j$ can be derived as:

$$\nabla^2_{\hat{p}_j}\mathcal{L}_{PA} \approx \frac{\alpha^2}{|C|}\frac{1}{||\hat{p}_j||^2}\left(\frac{\sum_{x\in B_j^+} h_j^+(x)\overline{f(x)}^2}{1 + \sum_{x\in B_j^+} h_j^+(x)} - (G_j^+)^2 + \frac{\sum_{x\in B_j^-} h_j^-(x)\overline{f(x)}^2}{1 + \sum_{x\in B_j^-} h_j^-(x)} - (G_j^-)^2\right). \tag{25}$$

Using the chain rule, the gradients and Hessians of $\mathcal{L}_{PA}$ with respect to the variational parameters μ_j and σ_j are derived as follows:

$$\nabla_{\mu_j}\mathcal{L}_{PA} = \nabla_{\hat{p}_j}\mathcal{L}_{PA}, \quad \nabla_{\sigma_j}\mathcal{L}_{PA} = \epsilon_j\nabla_{\hat{p}_j}\mathcal{L}_{PA}, \tag{26}$$

$$\nabla^2_{\mu_j}\mathcal{L}_{PA} = \nabla^2_{\hat{p}_j}\mathcal{L}_{PA}, \quad \nabla^2_{\sigma_j}\mathcal{L}_{PA} = \epsilon_j^2\nabla^2_{\hat{p}_j}\mathcal{L}_{PA}, \tag{27}$$

where ϵ_j's are those that were used in the Monte Carlo sampling, $\hat{p}_j = \mu_j + \sigma_j \bullet \epsilon_j$.

Finally, combining the above results, the gradients and Hessians of the VCPA loss function $\mathcal{L}_t$ can be written as:

$$g_{\mu_j^t} = \nabla_{\mu_j^t}\mathrm{KL} + \frac{1}{\tau}\nabla_{\mu_j^t}\mathcal{L}_{PA}, \quad g_{\sigma_j^t} = \nabla_{\sigma_j^t}\mathrm{KL} + \frac{1}{\tau}\nabla_{\sigma_j^t}\mathcal{L}_{PA}, \tag{28}$$

$$H_{\mu_j^t} = \nabla^2_{\mu_j^t}\mathrm{KL} + \frac{1}{\tau}\nabla^2_{\mu_j^t}\mathcal{L}_{PA}, \quad H_{\sigma_j^t} = \nabla^2_{\sigma_j^t}\mathrm{KL} + \frac{1}{\tau}\nabla^2_{\sigma_j^t}\mathcal{L}_{PA}, \tag{29}$$

B VISUALIZATION OF LEARNED PROXIES AND EMBEDDINGS

In this section we visualize the embeddings and the variational proxies that are learned from our VCPA. To visualize the 512-dimensional embedded space, we use the t-SNE for 2D visualization. We show the embeddings of the data points together with the proxies for the CUB dataset in Fig. 9. Overall, most of the classes (in different colors) are separated well in the embedded space, while each proxy roughly lies at the center of the corresponding class.

For the variational proxies, recall that we have mean vectors μ_j's and standard deviation vectors σ_j's for the training classes. As an intrinsic uncertainty measure, we define the *normalized standard deviation* for each variational proxy, as a trace of the empirical standard deviation matrix of the unit-norm normalized proxy samples. More specifically, the normalized standard deviation of the j-th proxy is defined as a scalar value from the follow formula:

$$\overline{\sigma}_j = \mathrm{Tr}\left(\mathrm{std}\left(\left\{\frac{\hat{p}_j^k}{||\hat{p}_j^k||}\right\}_{k=1}^K\right)\right), \quad \text{where} \quad \hat{p}_j^k = \mu_j + \sigma_j \bullet \epsilon^k, \quad \epsilon^k \sim \mathcal{N}(0, I), \quad k = 1, \ldots, K, \tag{30}$$

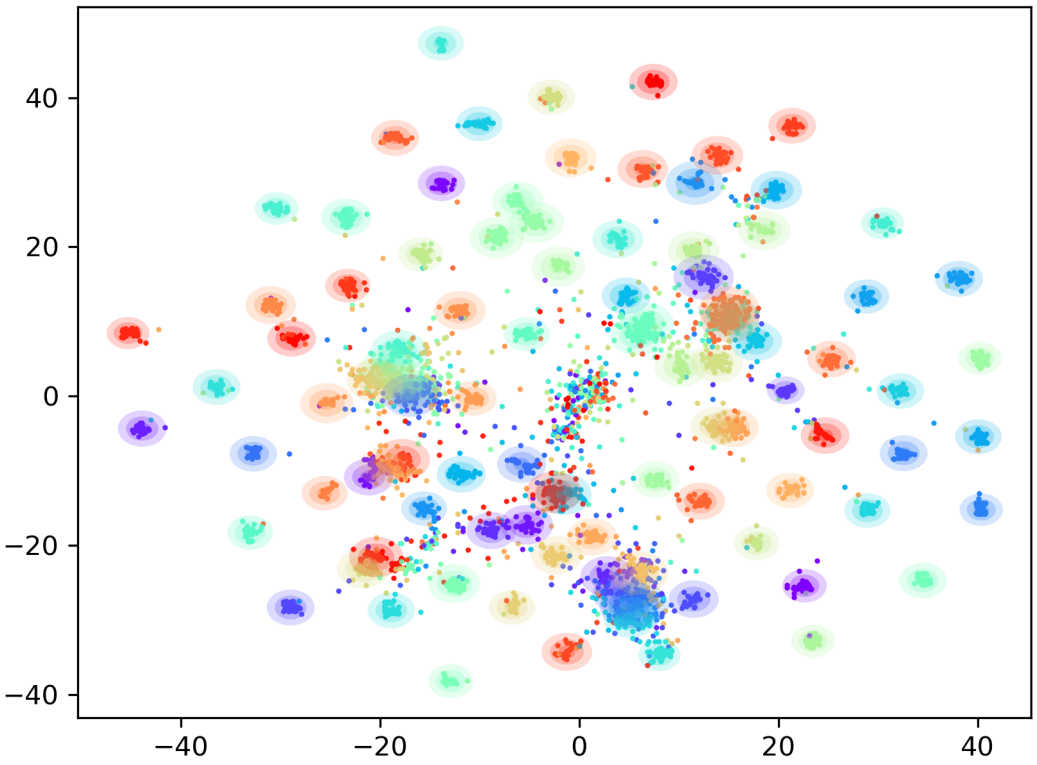
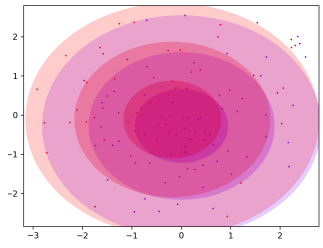


Figure 9: t-SNE visualization of the proxies and embeddings of the data samples on the CUB dataset. Our variational proxies (means and standard deviations) are represented as concentric circles at $\overline{\sigma}$ / $2\overline{\sigma}$ / $3\overline{\sigma}$. Different colors represent different classes.

where std($\{a_k\}_{k=1}^K$) means the empirical standard deviation using the K samples $\{a_k\}_{k=1}^K$. We use $K = 1000$. In the t-SNE plot (Fig. 9), we also visualize these normalized standard deviations together with the proxy means (shown as the circles of radii proportional to $\overline{\sigma}_j$'s). As shown, these intrinsic uncertainties ($\overline{\sigma}$) are visually well aligned with the degrees of dispersion of the samples in classes. E.g., if $\overline{\sigma}_j$, or the size of the circle, is larger (smaller), the samples in the class are more dispersed (concentrated, resp.). We also verify this quantitatively: the Pearson correlation score between $\overline{s}$ (the median cosine similarity between the proxy mean and the data samples in the class) and $\overline{\sigma}$ is $\mathbf{-0.81}$; meaning that if a class has large data-to-proxy similarity $\overline{s}$ (i.e., concentrated), the learned proxy uncertainty $\overline{\sigma}$ tends to be small, and vice versa.

Pairs of closest/farthest proxies. From the learned proxies, we visualize the pair of the closest classes and the pair of the farthest (according to the proxy-proxy similarity score) in Fig. 10. It can be seen that visually the images from the two closest classes look very similar to each other. On the other hand, the images from the two farthest classes exhibit substantially different appearance. This indicates that the proximity between the learned proxies are well aligned with the visual similarity between the corresponding classes.

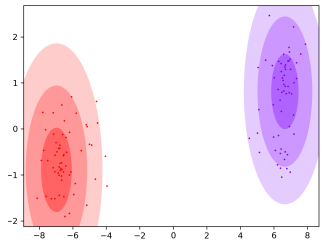
Comparison of the most certain and the most uncertain proxies. Next, we contrast the proxy with the smallest uncertainty $\overline{\sigma}$ against the proxy with the largest uncertainty. Intuitively, we conjecture that if a proxy has a large $\overline{\sigma}$ (high uncertainty), the corresponding class has much variation for its samples, and vice versa. We verify this in Fig. 11. Visually, the images from the RED class (the most certain proxy) have less diversity, while those from the PURPLE class (the most uncertain proxy) exhibit substantial appearance variation in its data samples. This signifies that the proxies and their intrinsic uncertainty measures learned by our VCPA successfully capture the degree of diversity of each class.



(a) Pair of the **closest** classes

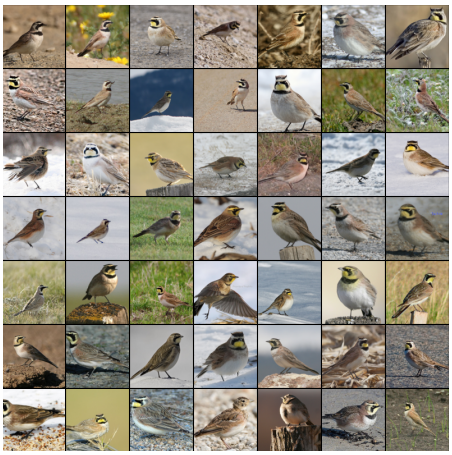
(b) Samples from the RED class

(c) Samples from the PURPLE class



(d) Pair of the **farthest** classes

(e) Samples from the RED class



(f) Samples from the PURPLE class

Figure 10: **TOP**: The pair of the closest classes according to their learned proxies (cosine similarity = 0.87). (a) t-SNE visualization of their variational proxies (shown as RED and PURPLE). (b) Samples from the RED class "Acadian Flycatcher" and (c) PURPLE "Least Flycatcher". Visually, the images from the two classes look very similar. **Bottom**: The pair of the farthest classes according to their learned proxies (cosine similarity = 0.22). (d) t-SNE visualization of their variational proxies (shown as RED and PURPLE). (e) Samples from the RED class "Ringed Kingfisher" and (f) PURPLE "Horned Lark". The images from the two classes exhibit substantially different appearance.

(a) The most (un)certain proxies (b) "RED" samples (certain) (c) "PURPLE" samples (uncert.)

Figure 11: Visualization of the most certain proxy and the most uncertain proxy. (a) t-SNE visualization of the two variational proxies: the most certain proxy (RED; the smallest $\overline{\sigma} = 1.25$) and the most uncertain proxy (PURPLE; the largest $\overline{\sigma} = 2.04$). (b) Samples from the RED class "American Crow" and (c) PURPLE "Dark eyed Junco". Visually, the images from the RED class (certain) have less diversity, while those from the PURPLE class (uncertain) exhibit substantial appearance variation.

Results on the Cars dataset. We perform similar experiments for the Cars dataset, and the results are shown in Fig. 12 (t-SNE), Fig. 13 (pairs of closest/farthest proxies), and Fig. 14 (the most certain/uncertain proxies).

C QUALITATIVE RETRIEVAL RESULTS

We present qualitative results on the four benchmark datasets to demonstrate the superiority of our proposed VCPA in comparison with Proxy-Anchor (Kim et al., 2020) and Proxy-NCA (Movshovitz-Attias et al., 2017). In these examples, all methods share the underlying embedding network with Inception-BN backbone, but it is trained with different loss functions hence the results are distinctively dissimilar. Fig. 15 illustrates the top-3 retrievals of the three methods on the CUB dataset. The inherent large intra-class variance (view-point, background) and inter-class similarity (foreground appearance) make this dataset challenging for the image retrieval task in general. However, the embedding network trained by our VCPA outperforms Proxy-Anchor and Proxy-NCA, with more accurate retrieval results. In particular, the third and fourth rows in Fig. 15 show that our proposed method can retrieve the correct samples despite the large variation in the view-point and background between the query and retrievals, whereas Proxy-Anchor and Proxy-NCA are unable to produce accurate results. Fig. 16 demonstrates the top-3 retrievals of the three methods on the Cars dataset. In the second and fifth rows of this figure, our proposed method can retrieve the correct samples despite the significant color difference from the query. Proxy-Anchor and Proxy-NCA also retrieve the samples with similar color (red), but they belong to different classes. The retrievals on the SOP and In-Shop datasets are shown in Fig. 17 and 18, respectively. In the second row of Fig. 17, the query image is vague as it shows the label with the small BMX bike at the corner and the barcode, but the embedding network trained by our VCPA can still retrieve the correct samples, while the other two methods fail to return accurate results. Fig. 18 shows that our proposed method is also robust to view-point changes.

D HYPERPARAMETERS AND ADDITIONAL EXPERIMENTS

D.1 Hyperparameters

The optimization hyperparameters used in our experiments are as follows.

- For the CUB dataset: embedding-dim $= 512$, $\tau = 0.01$, σ_{min} for proxies $= 1e-5$, $lr = 0.0001$, PA loss

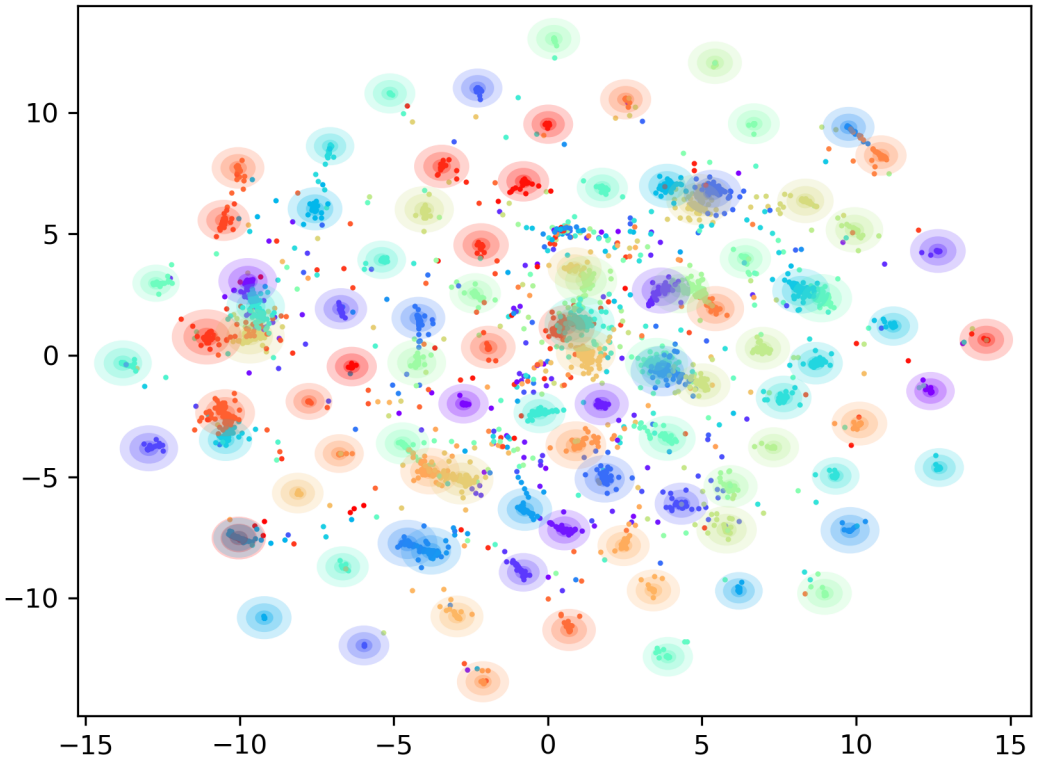
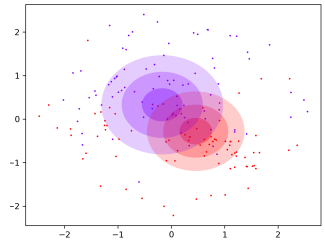


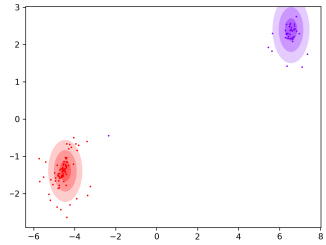
Figure 12: t-SNE visualization of the proxies and embeddings of the data samples on the Cars dataset. The same interpretation as Fig. 9.



(a) Pair of the **closest** classes

(b) Samples from the RED class

(c) Samples from the PURPLE class

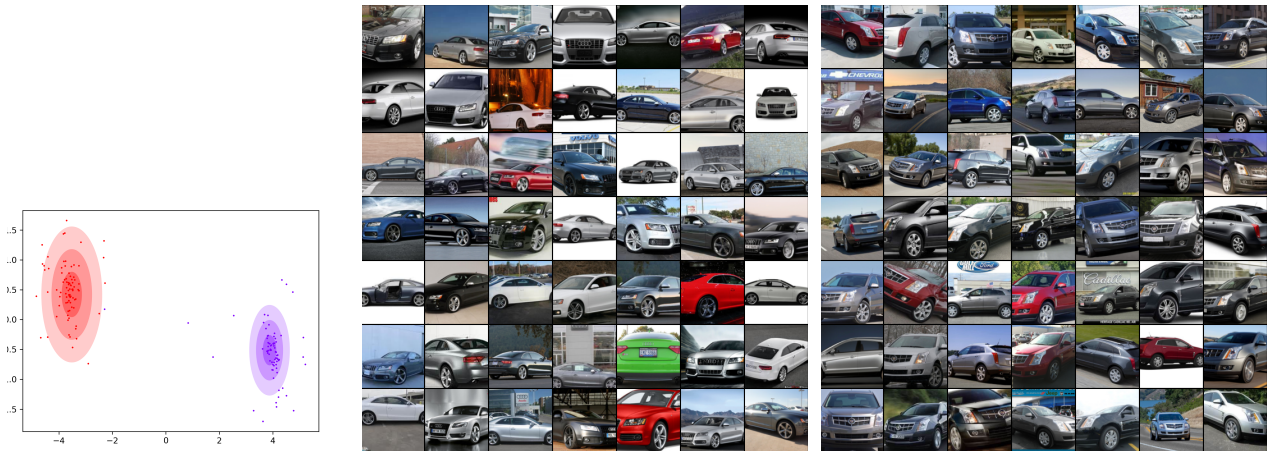


(d) Pair of the **farthest** classes

(e) Samples from the RED class

(f) Samples from the PURPLE class

Figure 13: On the Cars dataset. **TOP**: The pair of the closest classes according to their learned proxies (cosine similarity = 0.77). (a) t-SNE visualization of their variational proxies (shown as RED and PURPLE). (b) Samples from the RED class "Audi A5 Coupe 2012" and (c) PURPLE "Audi S5 Coupe 2012". Visually, the images from the two classes look very similar. **Bottom**: The pair of the farthest classes according to their learned proxies (cosine similarity = 0.24). (d) t-SNE visualization of their variational proxies (shown as RED and PURPLE). (e) Samples from the RED class "BMW X5 SUV 2007" and (f) PURPLE "Buick Regal GS 2012". The images from the two classes exhibit substantially different appearance (SUVs for RED and Sedans for PURPLE).



(a) The most (un)certain proxies (b) "PURPLE" samples (certain) (c) "RED" samples (uncertain)

Figure 14: On the Cars dataset. Visualization of the most certain proxy and the most uncertain proxy. (a) t-SNE visualization of the two variational proxies: the most certain proxy (PURPLE; the smallest $\overline{\sigma} = 0.51$) and the most uncertain proxy (RED; the largest $\overline{\sigma} = 0.76$). (b) Samples from the PURPLE class "Audi S5 Coupe 2012" and (c) RED "Cadillac SRX SUV 2012". Visually, the images from the PURPLE class (certain) have less diversity, while those from the RED class (uncertain) exhibit substantial appearance variation.

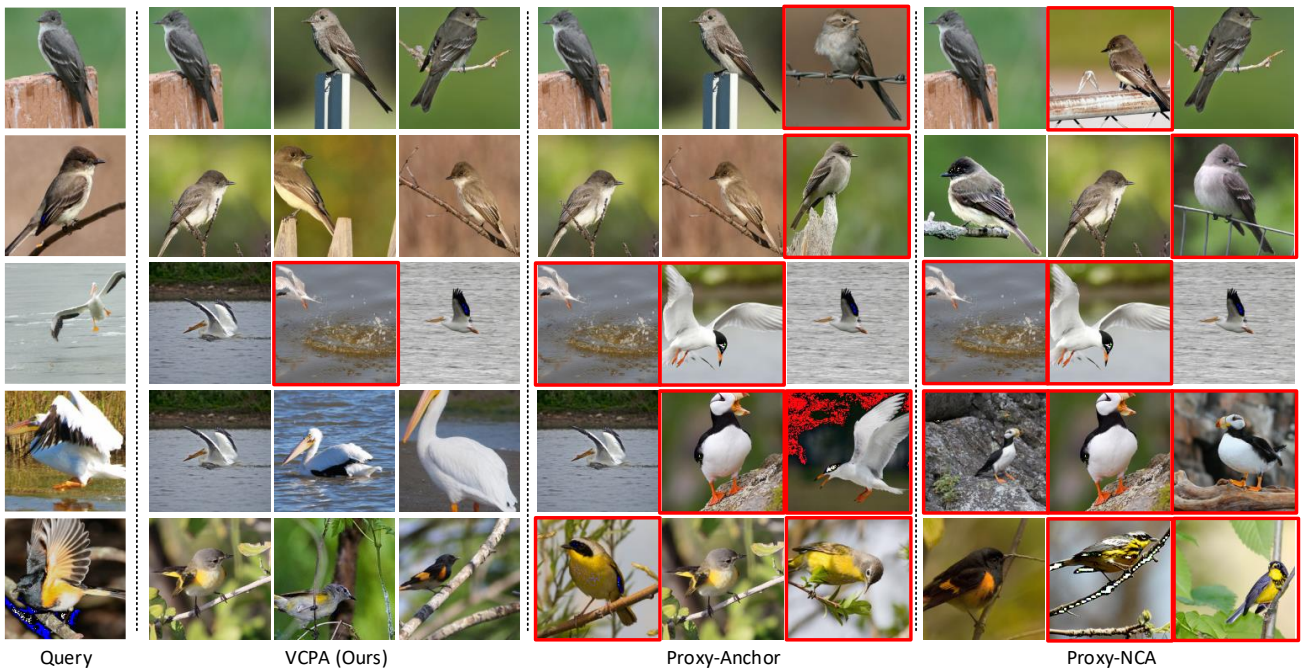


Figure 15: Qualitative results on the CUB dataset comparing our proposed method with Proxy-Anchor and Proxy-NCA methods. For each query image (leftmost), top-3 retrievals are presented. The result with red boundary is a failure case.

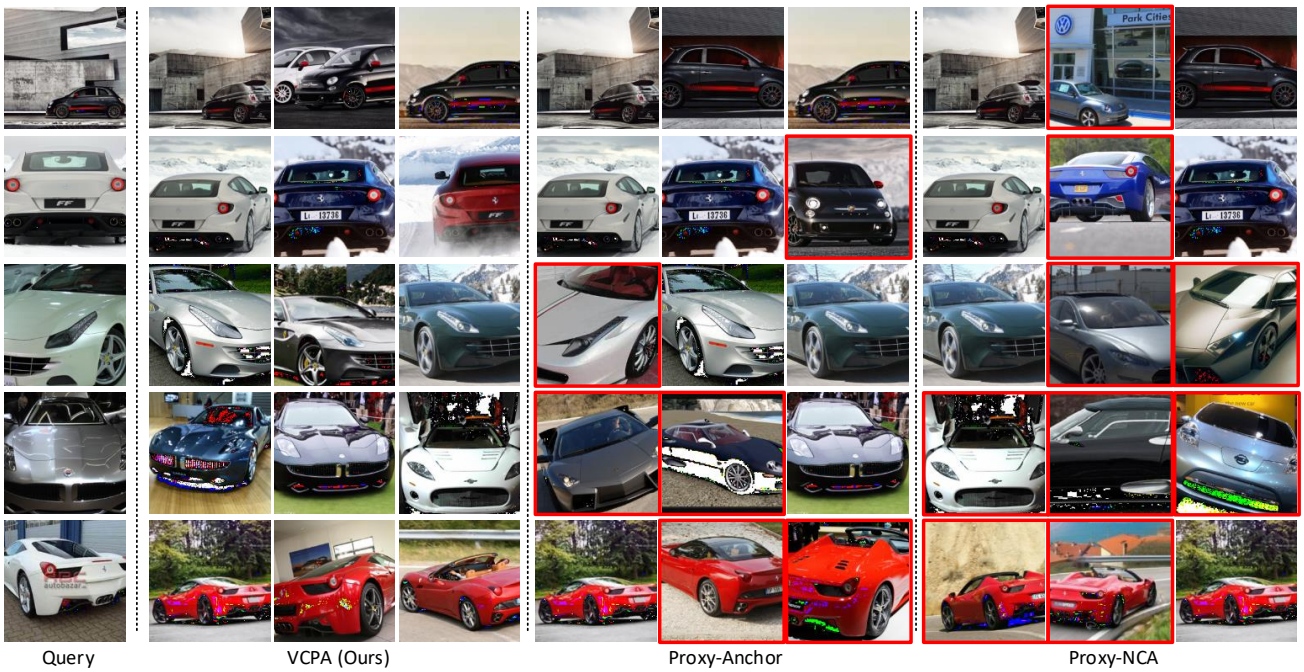


Figure 16: Qualitative results on the Cars dataset comparing our proposed method with Proxy-Anchor and Proxy-NCA methods. For each query image (leftmost), top-3 retrievals are presented. The result with red boundary is a failure case.

Figure 17: Qualitative results on the SOP dataset comparing our proposed method with Proxy-Anchor and Proxy-NCA methods. For each query image (leftmost), top-3 retrievals are presented. The result with red boundary is a failure case.

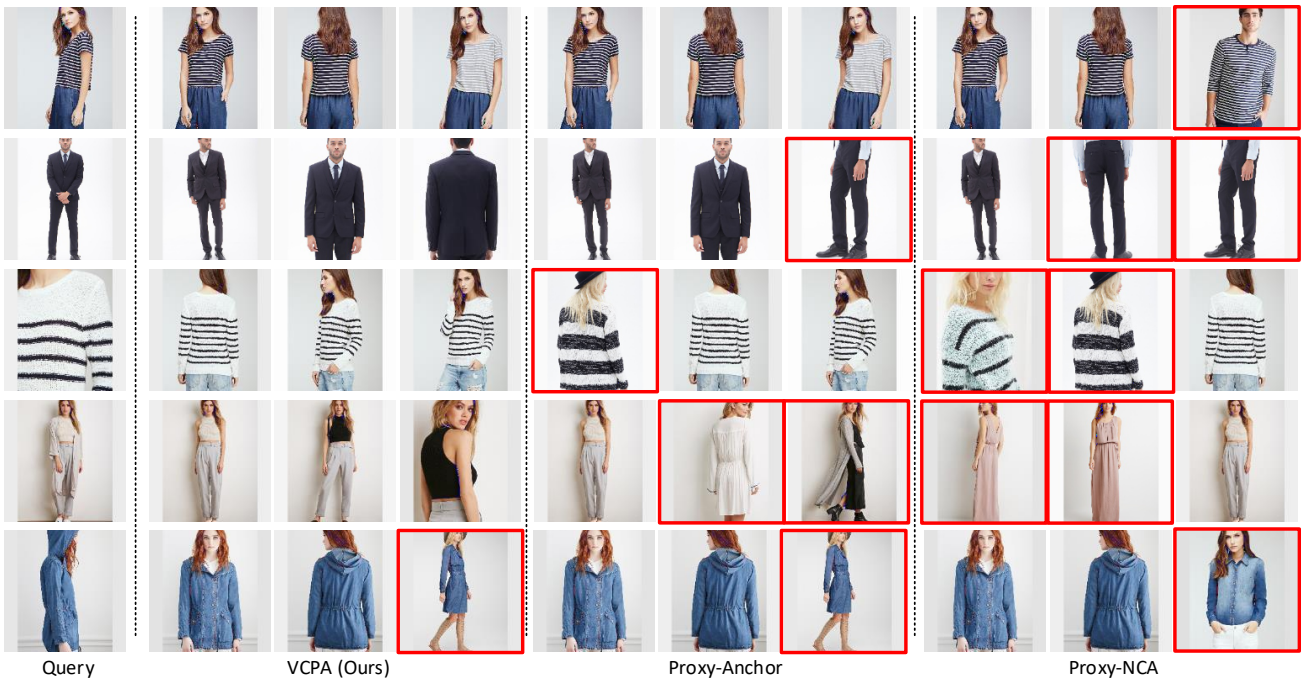


Figure 18: Qualitative results on the In-Shop dataset comparing our proposed method with Proxy-Anchor and Proxy-NCA methods. For each query image (leftmost), top-3 retrievals are presented. The result with red boundary is a failure case.

Size	224×224	256×256	324×324	448×448
PA	68.4	71.1	74.0	77.3
VCPA	69.9	71.5	74.9	77.7

Table 3: Large image resolution experiments (R@1, CUB).

margin $\alpha = 32$, $\delta = 0.1$, and AdamW optimizer with batch size 180, 60 epochs, weight-decay $= 1e-4$, lr-decay-step $= 10$, lr-decay-gamma $= 0.5$, and newton updates per batch $nupb = 10$.

- For the Cars dataset: embedding-dim $= 512$, $\tau = 0.01$, σ_{min} for proxies $= 1e-5$, $lr = 0.0001$, PA loss margin $\alpha = 32$, $\delta = 0.1$, and AdamW optimizer with batch size 180, 100 epochs, weight-decay $= 1e-4$, lr-decay-step $= 20$, lr-decay-gamma $= 0.5$, and newton updates per batch $nupb = 10$.

- For the SOP dataset: embedding-dim $= 512$, $\tau = 0.001$, $lr = 0.0006$, PA loss margin $\alpha = 32$, $\delta = 0.1$, and AdamW optimizer with batch size 180, 100 epochs, weight-decay $= 1e-4$, lr-decay-step $= 20$, lr-decay-gamma $= 0.5$, and no newton update.

- For the In-Shop dataset: embedding-dim $= 512$, $\tau = 0.001$, $lr = 0.0006$, PA loss margin $\alpha = 32$, $\delta = 0.1$, and AdamW optimizer with batch size 180, 60 epochs, weight-decay $= 1e-4$, lr-decay-step $= 20$, lr-decay-gamma $= 0.25$, and no newton update.

Note that we had best results with $\tau < 1$ (e.g., $\tau = 0.01$ (CUB) and $\tau = 0.001$ (Cars)), which is in line with GVCL (Loo et al., 2020), related to the structural granularity of learned distributions.

D.2 Large Image Resolutions

We also test VCPA on large image resolutions. For the CUB dataset, we increase the image sizes from (224×224) to (256×256), (324×324), and (448×448). The results are summarized in Table 3. For all image resolutions, our VCPA outperforms PA consistently.

Table 4: PA and VCPA on CUB and Cars datasets. Averaged over 5 random runs.

CUB	R@1	2	4	8
PA	$68.04^{\pm 0.08}$	$78.33^{\pm 0.16}$	$85.89^{\pm 0.16}$	$91.31^{\pm 0.09}$
VCPA	$69.41^{\pm 0.11}$	$79.40^{\pm 0.10}$	$87.01^{\pm 0.08}$	$92.15^{\pm 0.04}$

Cars	R@1	2	4	8
PA	$86.29^{\pm 0.14}$	$91.71^{\pm 0.18}$	$95.06^{\pm 0.11}$	$97.23^{\pm 0.10}$
VCPA	$86.58^{\pm 0.09}$	$91.98^{\pm 0.08}$	$95.28^{\pm 0.07}$	$97.31^{\pm 0.05}$

D.3 Standard Errors in Main Results

Table 4 shows the standard errors in R@1 for the CUB and Cars datasets, which were missing in Table 1 due to the space limit. For SOP and InShop, it is rather difficult to perform multiple runs due to large training time.