PADS: Policy-Adapted Sampling for Visual Similarity Learning

Karsten Roth*  Timo Milbich*  Björn Ommer
Heidelberg Collaboratory for Image Processing / IWR
Heidelberg University, Germany

Abstract

Learning visual similarity requires to learn relations, typically between triplets of images. Albeit triplet approaches being powerful, their computational complexity mostly limits training to only a subset of all possible training triplets. Thus, sampling strategies that decide when to use which training sample during learning are crucial. Currently, the prominent paradigm are fixed or curriculum sampling strategies that are predefined before training starts. However, the problem truly calls for a sampling process that adjusts based on the actual state of the similarity representation during training. We, therefore, employ reinforcement learning and have a teacher network adjust the sampling distribution based on the current state of the learner network, which represents visual similarity. Experiments on benchmark datasets using standard triplet-based losses show that our adaptive sampling strategy significantly outperforms fixed sampling strategies. Moreover, although our adaptive sampling is only applied on top of basic triplet-learning frameworks, we reach competitive results to state-of-the-art approaches that employ diverse additional learning signals or strong ensemble architectures. Code can be found under https://github.com/Confusezius/CVPR2020_PADS.

1. Introduction

Capturing visual similarity between images is the core of virtually every computer vision task, such as image retrieval[60, 53, 39, 36], pose understanding [35, 8, 3, 54], face detection[49] and style transfer [28]. Measuring similarity requires to find a representation which maps similar images close together and dissimilar images far apart. This task is naturally formulated as Deep Metric Learning (DML) in which individual pairs of images are compared[17, 53, 38] or contrasted against a third image[49, 60, 57] to learn a distance metric that reflects image similarity. Such triplet learning constitutes the basis of powerful learning algorithms[45, 39, 47, 62]. However, with growing training set size, leveraging every single triplet for learning becomes computationally infeasible, limiting training to only a subset of all possible triplets. Thus, a careful selection of those triplets which drive learning best, is crucial. This raises the question: How to determine which triplets to present when
to our model during training?
As training progresses, more and more triplet relations will be correctly represented by the model. Thus, ever fewer triplets will still provide novel, valuable information. Conversely, leveraging only triplets which are hard to learn[49, 9, 63] but therefore informative, impairs optimization due to high gradient variance[60]. Consequently, a reasonable mixture of triplets with varying difficulty would provide an informative and stable training signal. Now, the question remains, when to present which triplet? Sampling from a fixed distribution over difficulties may serve as a simple proxy[60] and is a typical remedy in representation learning in general[27, 5]. However, (i) choosing a proper distribution is difficult; (ii) the abilities and state of our model evolves as training progresses and, thus, a fixed distribution cannot optimally support every stage of training; and (iii) triplet sampling should actively contribute to the learning objective rather than being chosen independently. Since a manually predefined sampling distribution does not fulfill these requirements, we need to learn and adapt it while training a representation.

Such online adaptation of the learning algorithm and parameters that control it during training is typically framed as a teacher-student setup and optimized using Reinforcement Learning (RL). When modelling a flexible sampling process (the student), a controller network (the teacher) learns to adjust the sampling such that the DML model is steadily provided with an optimal training signal. Fig. 1 compares progressions of learned sampling distributions adapted to the DML model with a typical fixed sampling distribution[60].

This paper presents how to learn a novel triplet sampling strategy which is able to effectively support the learning process of a DML model at every stage of training. To this end, we model a sampling distribution so it is easily adjustable to yield triplets of arbitrary mixtures of difficulty. To adapt to the training state of the DML model we employ Reinforcement Learning to update the adjustment policy. Directly optimizing the policy so it improves performance on a held-back validation set, adjusts the sampling process to optimally support DML training. Experiments show that our adaptive sampling strategy significantly improves over fixed, manually designed triplet sampling strategies on multiple datasets. Moreover, we perform diverse analyses and ablations to provide additional insights into our method.

2. Related Work

Metric learning has become the leading paradigm for learning distances between images with a broad range of applications, including image retrieval[37, 31, 60], image classification[11, 64], face verification[49, 20, 32] or human pose analysis[35, 8]. Ranking losses formulated on pairs[53, 17], triplets[49, 60, 57, 12] or even higher order tuples of images[7, 38, 58] emerged as the most widely used basis for DML[46]. As with the advent of CNNs datasets are growing larger, different strategies are developed to cope with the increasing complexity of the learning problem.

Complexity management in DML: The main line of research are negative sampling strategies[49, 60, 18] based on distances between an anchor and a negative image. FaceNet[49] leverages only the hard negatives in a mini-batch. Wu et al. [60] sample negatives uniformly over the whole range of distances to avoid large variances in the gradients while optimization. Harwood et al. [18] restrict and control the search space for triplets using pre-computed sets of nearest neighbors by linearly regressing the training loss. Each of them successfully enable effective DML training. However, these works are based on fixed and manually predefined sampling strategies. In contrast, we learn an adaptive sampling strategy to provide an optimal input stream of triplets conditioned on the training state of our model. Orthogonal to sampling negatives from the training set is the generation of hard negatives in form of images[9] or feature vectors[65, 63]. Thus, these approaches also resort to hard negatives, while our sampling process yields negatives of any mixture of difficulty depending on the model state.

Finally, proxy based techniques reduce the complexity of the learning problem by learning one[37] or more [43] virtual representatives for each class, which are used as negatives. Thus, these approaches approximate the negative distributions, while our sampling adaptively yields individual negative samples.

Advanced DML: Based on the standard DML losses many works improve model performance using more advanced techniques. Ensemble methods [39, 62, 47] learn and combine multiple embedding spaces to capture more information. HORDE[23] additionally forces feature representations of related images to have matching higher moments. Roth et al. [45] combines class-discriminative features with features learned from characteristics shared across classes. Similarly, Lin et al. [31] proposes to learn the intra-class distributions, next to the inter-class distribution. All these approaches are applied in addition to the standard ranking losses discussed above. In contrast, our work presents a novel triplet sampling strategy and, thus, is complementary to these advanced DML methods.

Adaptive Learning: Curriculum Learning[4] gradually increases the difficulty of the the samples presented to the model. Hacohen et al. [16] employ a batch-based learnable scoring function to provide a batch-curriculum for training, while we learn how to adapt a sampling process to the training state. Graves et al. [15] divide the training data into fixed subsets before learning in which order to use them from training. Further, Gopal et al. [14] employs an empirical online importance sampling distribution over inputs based on their gradient magnitudes during training. Similarly, Shreyas et al. [48] learn an importance sampling over
instances. In contrast, we learn an online policy for selecting triplet negatives, thus instance relations. Meta Learning aims at learning how to learn. It has been successfully applied for various components of a learning process, such as activation functions [44], input masking [10], self-supervision [6], finetuning [52], loss functions [21], optimizer parameters [2] and model architectures [42, 61]. In this work, we learn a sampling distribution to improve triplet-based learning.

3. Distance-based Sampling for DML

Let \( \phi_i := \phi(I_i; \zeta) \) be a D-dimensional embedding of an image \( I_i \in \mathbb{R}^{H \times W \times 3} \) with \( \phi(I_i; \zeta) \) being represented by a deep neural network parametrized by \( \zeta \). Further, \( \phi \) is normalized to a unit hypersphere \( S \) for regularization purposes [49]. Thus, the objective of DML is to learn \( \phi: \mathbb{R}^{H \times W \times 3} \rightarrow \Phi \subseteq S \) such that images \( I_i, I_j \in \mathcal{I}_{\text{train}} \) are mapped close to another if they are similar and far otherwise, under a standard distance function \( d(\phi_i, \phi_j) \). Commonly, \( d \) is the euclidean distance, i.e. \( d_{ij} := ||\phi_i - \phi_j||_2 \).

A popular family of training objectives for learning \( \phi \) are ranking losses [49, 60, 53, 38, 38, 17] operating on tuples of images. Their most widely used representative is arguably the triplet loss [49] which is defined as an ordering task between images \( \{I_a, I_p, I_n\} \), formulated as

\[
\mathcal{L}_{\text{triplet}}(\{I_a, I_p, I_n\}; \zeta) = \max(0, d_{ap}^2 - d_{an}^2 + \gamma)
\]

Here, \( I_a \) and \( I_p \) are the anchor and positive with the same class label. \( I_n \) acts as the negative from a different class.

Optimizing \( \mathcal{L}_{\text{triplet}} \) pushes \( I_a \) closer to \( I_p \) and further away from \( I_n \) as long as a constant distance margin \( \gamma \) is violated.

3.1. Static Triplet sampling strategies

While ranking losses have proven to be powerful, the number of possible tuples grows dramatically with the size of the training set. Thus, training quickly becomes infeasible, turning efficient tuple sampling strategies into a key component for successful learning as discussed here.

When performing DML using ranking losses like Eq. 1, triplets decreasingly violate the triplet margin \( \gamma \) as training progresses. Naively employing random triplet sampling entails many of the selected triplets being uninformative, as distances on \( \Phi \) are strongly biased towards larger distances \( d \) due to its regularization to \( S \). Consequently, recent sampling strategies explicitly leverage triplets which violate the triplet margin and, thus, are difficult and informative.

(Semi-)Hard negative sampling: Hard negative sampling methods focus on triplets violating the margin \( \gamma \) the most, i.e. by sampling negatives \( I_n^* = \arg \min_{I_n \in \mathcal{I}; d_{an} < d_{ap}} d_{an} \). While it speeds up convergence, it may result in collapsed models [49] due to a strong focus on few data outliers and very hard negatives. Facenet [49] proposes a relaxed, semi-hard negative sampling strategy restricting the sampling set to a single mini-batch \( \mathcal{B} \) by employing negatives \( I_n^* = \arg \min_{I_n \in \mathcal{B}; d_{an} > d_{ap}} d_{an} \). Based on this idea, different online [40, 53] and offline [18] strategies emerged.

(Static) Distance-based sampling: By considering the hardness of a negative, one can successfully discard easy and uninformative triplets. However, triplets that are too hard lead to noisy learning signals due to overall high gradient variance [60]. As a remedy, to control the variance while maintaining sufficient triplet utility, sampling can be extended to also consider easier negatives, i.e. introducing a sampling distribution \( I_n \sim p(I_n|I_a) \) over the range of distances \( d_{an} \) between anchor and negatives. Wu et al. [60] propose to sample from a static uniform prior on the range of \( d_{an} \), thus equally considering negatives from the whole spectrum of difficulties. As pairwise distances on \( \Phi \) are strongly biased towards larger \( d_{an} \), their sampling distribution requires to weigh \( p(I_n|I_a) \) inversely to the analytical distance distribution on \( \Phi \): \( q(d) \propto d^{D-2} \left[ 1 - \frac{1}{2}d^2 \right]^{\frac{D-2}{2}} \) for large \( D \geq 128 [1] \). Distance-based sampling from the static, uniform prior is then performed by

\[
I_n \sim p(I_n|I_a) \propto \min (\lambda, q^{-1}(d_{an}))
\]

with \( \lambda \) being a clipping hyperparameter for regularization.

4. Learning an Adaptive Negative Sampling

Distance-based sampling of negatives \( I_n \) has proven to offer a good trade-off between fast convergence and a stable, informative training signal. However, a static sampling distribution \( p(I_n|I_a) \) provides a stream of training data independent of the the changing needs of a DML model during learning. While samples of mixed difficulty may be useful at the beginning, later training stages are calling for samples of increased difficulty, as e.g. analyzed by curriculum learning [4]. Unfortunately, as different models and even different model intializations [13] exhibit distinct learning dynamics, finding a generally applicable learning schedule is challenging. Thus, again, heuristics [16] are typically employed, inferring changes after a fixed number of training...
we can learn a policy which effectively alters $p$ which directly facilitate maximal DML performance. Since $u$ optimal training signals for learning $\phi$ implemented as a neural network parametrized by $\theta$ task by learning a stochastic adjustment policy $\phi$ to be updated while learning when to use changes during learning, which effectively controls discretized sampling distribution.

**4.1. Modelling a flexible sampling distribution**

Since learning benefits from a diverse distribution $p(I_n|I_a)$ of negatives, uni-modal distributions (e.g. Gaussians, Binomials, $\chi^2$) are insufficient. Thus, we utilize a discrete probability mass function $p(I_n|I_a) := p(\{d_{a_k} \in u_k\})$, where the bounded interval $U = [\lambda_{\text{min}}, \lambda_{\text{max}}]$ of possible distances $d_{a_n}$ is discretized into disjoint equidistant bins $u_1, \ldots, u_K$. The probability of drawing $I_n$ from bin $u_k$ is $p_k$ with $p_k \geq 0$ and $\sum_k p_k = 1$. Fig. 2 illustrates this discretized sampling distribution.

This representation of the negative sampling distribution effectively controls which samples are used to learn $\phi$. As $\phi$ changes during learning, $p(I_n|I_a)$ should also adapt to always provide the most useful training samples, i.e. to control when to use which sample. Hence the probabilities $p_k$ need to be updated while learning $\phi$. We subsequently solve this task by learning a stochastic adjustment policy $\pi_\theta$ for the $p_k$, implemented as a neural network parametrized by $\theta$.

**4.2. Learning an adjustment policy for $p(I_n|I_a)$**

Our sampling process based on $p(I_n|I_a)$ should provide optimal training signals for learning $\phi$ at every stage of training. Thus, we adjust the $p_k$ by a multiplicative update $a \in A$ conditioned on the current representation (or state) $s \in S$ of $\phi$ during learning. We introduce a conditional distribution $\pi_\theta(a|s)$ to control which adjustment to apply at which state $s$ of training $\phi$. To learn $\pi_\theta$, we measure the utility of these adjustments for learning $\phi$ using a reward signal $r = r(s, a)$. We now first describe how to model each of these components, before presenting how to efficiently optimize the adjustment policy $\pi_\theta$ alongside $\phi$.

**Adjustments $a$:** To adjust $p(I_n|I_a)$, $\pi_\theta(a|s)$ proposes adjustments $a$ to the $p_k$. To lower the complexity of the action space, we use a limited set of actions $A = \{\alpha, 1, \beta\}$ to individually decrease, maintain, or increase the probabilities $p_k$ for each bin $u_k$, i.e. $a := \{a_k \in \{\alpha, 1, \beta\}\}_{k=1}^K$. Further, $\alpha, \beta$ are fixed constants $0 < \alpha < 1, \beta > 1$ and $\alpha + \beta = 1$. Updating $p(I_n|I_a)$ is then simply performed by bin-wise updates $p_k \leftarrow p_k \cdot a_k$ followed by re-normalization. Using a multiplicative adjustment accounts for the exponential distribution of distances on $\Phi$ (cf. Sec. 3.1).

**Training states $s$:** Adjustments $a$ depend on the present state $s \in S$ of the representation $\phi$. Unfortunately, we cannot use the current model weights $\zeta$ of the embedding network, as the dimensionality of $s$ would be too high, thus making optimization of $\pi_\theta$ infeasible. Instead, we represent the current training state using representative statistics describing the learning progress: running averages over Recall@1[24], NMI[33] and average distances between and within classes on a fixed held-back validation set $\mathcal{I}_{\text{val}}$. Additionally we use past parametrizations of $p(I_n|I_a)$ and the relative training iteration (cf. Implementation details, Sec. 5).

**Rewards $r$:** An optimal sampling distribution $p(I_n|I_a)$ yields triplets whose training signal consistently improves the evaluation performance of $\phi$ while learning. Thus, we compute the reward $r$ for adjustments $a \sim \pi_\theta(a|s)$ by directly measuring the relative improvement of $\phi(\cdot; \zeta)$ over...
\( \phi(\cdot; \zeta') \) from the previous training state. This improvement is quantified through DML evaluation metrics \( e(\phi(\cdot; \zeta), I_{\text{val}}) \) on the validation set \( I_{\text{val}} \). More precisely, we define \( r \) as

\[
r = \text{sign}(e(\phi(\cdot; \zeta), I_{\text{val}}) - e(\phi(\cdot; \zeta'), I_{\text{val}}))
\]

where \( \zeta \) was reached from \( \zeta' \) after \( M \) DML training iterations using \( p(I_n|I_a) \). We choose \( e \) to be the sum of Recall@1[24] and NMI[33]. Both metrics are in the range [0, 1] and target slightly different performance aspects. Further, similar to [21], we utilize the sign function for consistent learning signals even during saturated training stages.

**Learning of \( \pi_{\theta} \):** Adjusting \( p(I_n|I_a) \) is a stochastic process controlled by actions \( a \) sampled from \( \pi_{\theta}(a|s) \) based on a current state \( s \). This defines a Markov Decision Process (MDP) naturally optimized by Reinforcement Learning. The policy objective \( J(\theta) \) is formulated to maximize the total expected reward \( R(\tau) = \sum_{t=0}^{T} r_t(a_t, s_t) \) over training episodes of tuples \( \tau = \{ (a_t, s_t, r_t) | t = 0, \ldots, T \} \) collected from sequences of \( T \) time-steps, i.e.

\[
J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [R(\tau)]
\]

Hence, \( \pi_{\theta} \) is optimized to predict adjustments \( a \) for \( p(I_n|I_a) \) which yield high rewards and thereby improving the performance of \( \phi \). Common approaches use episodes \( \tau \) comprising long state trajectories which potentially cover multiple training epochs[10]. As a result, there is a large temporal discrepancy between model and policy updates. However, in order to closely adapt \( p(I_n|I_a) \) to the learning of \( \phi \), this discrepancy needs to be minimized. In fact, our experiments show that single-step episodes, i.e. \( T = 1 \), are sufficient for optimizing \( \pi_{\theta} \) to infer meaningful adjustments \( a \) for \( p(I_n|I_a) \).

In summary, our training episodes \( \tau \) consists of updating \( p(I_n|I_a) \) using a sampled adjustment \( a \), performing \( M \) DML training iterations based on the adjusted \( p(I_n|I_a) \) and updating \( \pi_{\theta} \) using the resulting reward \( r \). Optimizing Eq. 4 is then performed by standard RL algorithms which approximate different variations of the policy gradient based on the gain \( G(s, a) \),

\[
\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(a|s) G(s, a)]
\]

The choice of the exact form of \( G = G(s, a) \) gives rise to different optimization methods, e.g. REINFORCE[59] (\( G = R(\tau) \)), Advantage Actor Critic (A2C)[55] (\( G = A(s, a) \)), etc. Other RL algorithms, such as TRPO[50] or PPO[51] replace Eq. 4 by surrogate objective functions. Fig. 3 provides an overview over the learning procedure. Moreover, in the supplementary material we compare different RL algorithms and summarizes the learning procedure in Alg. 1 using PPO[51] for policy optimization.

**Initialization of \( p(I_n|I_a) \):** We find that an initialization with a slight emphasis towards smaller distances \( d_{\text{sm}} \) works best. However, as shown in Tab. 5, also other initializations work well. In addition, the limits of the distance interval \( U = [\lambda_{\text{min}}, \lambda_{\text{max}}] \) can be controlled for additional regularization as done in [60]. This means ignoring values above \( \lambda_{\text{max}} \) and clipping values below \( \lambda_{\text{min}} \), which is analysed in Tab. 5.

**Self-Regularisation:** As noted in [45], the utilisation of intra-class features can be beneficial to generalization. Our approach easily allows for a learnable inclusion of such features. As positive samples are generally closest to anchors, we can merge positive samples into the set of negative samples and have the policy learn to place higher sampling probability on such low-distance cases. We find that this additionally improves generalization performance.

**Computational costs:** Computational overhead over fixed sampling strategies[49, 60] comes from the estimation of \( r \) requiring a forward pass over \( I_{\text{val}} \) and the computation of the evaluation metrics. For example, setting \( M = 30 \) increases the computation time per epoch by less than 20%.

### 5. Experiments

In this section we provide implementation details, evaluations on standard metric learning datasets, ablations studies and analysis experiments.

**Implementation details.** We follow the training protocol of [60] with ResNet50. During training, images are resized to \( 256 \times 256 \) with random crop to \( 224 \times 224 \) and random horizontal flipping. For completeness, we also evaluate on Inception-BN [22] following standard practice in the supplementary. The initial learning rates are set to \( 10^{-5} \). We choose triplet parameters according to [60], with \( \gamma = 0.2 \). For margin loss, we evaluate margins \( \beta = 0.6 \) and \( \beta = 1.2 \). Our policy \( \pi \) is implemented as a two-layer fully-connected network with ReLU-linearity inbetween and 128 neurons per layer. Action values are set to \( \alpha = 0.8, \beta = 1.25 \). Episode iterations \( M \) are determined via cross-validation within [30,150]. The sampling range \( [\lambda_{\text{min}}, \lambda_{\text{max}}] \) of \( p(I_n|I_a) \) is set to \( [0.1, 1.4] \), with \( K = 30 \). The sampling probability of negatives corresponding to distances outside this interval is set to 0. For the input state we use running averages of validation recall, NMI and average intra- and interclass distance based on running average lengths of 2, 8, 16 and 32 to account for short- and longterm changes. We also incorporate the metrics of the previous 20 iterations. Finally, we include the sampling distributions of the previous iteration and the training progress normalized over the total training length. For optimization, we utilize an A2C + PPO setup with ratio limit \( \epsilon = 0.2 \). The history policy is updated every 5 policy iterations. For implementation we use the
| Dataset       | CUB200-2011[56] | CARS196[29] | SOP[38] |
|--------------|----------------|-------------|---------|
| Approach     | Dim | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI | R@1 | R@10 | R@100 | NMI |
| Margin[60] + $\mathcal{U}$-dist (orig) | 128 | 63.6 | 74.4 | 83.1 | 69.0 | 79.6 | 86.5 | 90.1 | 69.1 | 72.7 | 86.2 | 93.8 | 90.7 |
| Margin[60] + $\mathcal{U}$-dist (Relmp, $\beta = 1.2$) | 128 | 63.5 | 74.9 | 84.4 | 68.1 | 80.1 | 87.4 | 91.9 | 67.6 | 74.6 | 87.5 | 94.2 | 90.7 |
| Margin[60] + $\mathcal{U}$-dist (Relmp, $\beta = 0.6$) | 128 | 63.0 | 74.3 | 83.0 | 66.9 | 79.7 | 87.0 | 91.8 | 67.1 | 73.5 | 87.2 | 93.9 | 89.3 |
| Margin[60] + PADS (Ours) | 128 | 67.3 | 78.0 | 85.9 | 69.9 | 83.5 | 89.7 | 93.8 | 68.8 | 76.5 | 89.0 | 95.4 | 89.9 |
| Triplet[49] + semihard (orig) | 64 | 42.6 | 55.0 | 66.4 | 55.4 | 51.5 | 63.8 | 73.5 | 53.4 | 66.7 | 82.4 | 91.9 | 89.5 |
| Triplet[49] + semihard (Relmp) | 128 | 60.6 | 72.3 | 82.1 | 65.5 | 71.9 | 81.5 | 88.5 | 64.1 | 73.5 | 87.5 | 94.9 | 89.2 |
| Triplet[49] + $\mathcal{U}$-dist (Relmp) | 128 | 62.2 | 73.2 | 82.8 | 66.3 | 78.0 | 85.6 | 91.4 | 65.7 | 73.9 | 87.7 | 94.5 | 89.3 |
| Triplet[49] + PADS (Ours) | 128 | 64.0 | 75.5 | 84.3 | 67.8 | 79.9 | 87.5 | 92.3 | 67.1 | 74.8 | 88.2 | 95.0 | 89.5 |

Table 1: Comparison of our proposed adaptive negative sampling (PADS) against common static negative sampling strategies: semihard negative mining[38] (semihard) and static distance-based sampling ($\mathcal{U}$-dist)[60] using triplet[49] and margin loss[60]. Relmp. denotes our re-implementations and Dim the dimensionality of $\phi$.  

PyTorch framework[41] on a single NVIDIA Titan X. 

**Benchmark datasets.** We evaluate the performance on three common benchmark datasets. For each dataset the first half of classes is used for training and the other half is used for testing. Further, we use a random subset of 15% of the training images for our validation set $I_{\text{val}}$. We use: CARS196[29], with 16,185 images from 196 car classes. CUB200-2011[56], 11,788 bird images from 200 classes. Stanford Online Products (SOP)[38], containing 120,053 images divided in 22,634 classes.

### 5.1. Results

In Tab. 1 we apply our adaptive sampling strategy on two widely adopted basic ranking losses: triplet[49] and margin loss[60]. For each loss, we compare against the most commonly used static sampling strategies, semi-hard[49] (semihard) and distance-based sampling[60] ($\mathcal{U}$-dist) on the CUB200-2011, CARS196 and SOP dataset. We measure image retrieval performance using recall accuracy R@$k$[24] following [39]. For completeness we additionally show the normalized mutual information score (NMI)[33], despite not fully correlating with retrieval performance. For both losses and each dataset, our learned negative sampling significantly improves the performance over the non-adaptive sampling strategies. Especially the strong margin loss greatly benefits from the adaptive sampling, resulting in boosts up to 3.8% on CUB200-2011, 3.4% on CARS196 and 1.9% on SOP. This clearly demonstrates the importance of adjusting triplet sampling to the learning process a DML model, especially for smaller datasets.

Next, we compare these results with the current state-of-the-art in DML which extend these basic losses using diverse additional training signals (MIC[45], DVML[31], HORDE[23], A-BIER[39]), ensembles of embedding spaces (DREML[62], D&C[47], Rank[58]) and/or significantly more network parameters (HORDE[23], SOFT-TRIPLE[43]). Tab. 2 shows that our results, despite not using such additional extensions, compete and partly even surpass these strong methods. On CUB200-2011 we outperform all methods, including the powerful ensembles, by at least 1.2% in Recall accuracy. On CARS196[29] we rank second behind the top performing non-ensemble method D&C[47]. On SOP[38] we lose 0.7% to MIC[45] which, in turn, we surpass on both CUB200-2011 and CARS196. This highlights the strong benefit of our adaptive sampling.

### 5.2. Analysis

We now present various analysis experiments providing detailed insights into our learned adaptive sampling strategy. 

**Training progression of $p(I_n|I_a)$:** We now analyze in Fig. 4 how our adaptive sampling distribution progresses during training by averaging the results of multiple training runs on CUB200-2011, CARS196 and SOP.
Table 2: Comparison to the state-of-the-art DML methods on CUB200-2011[56], CARS196[29] and SOP[38]. Dim denotes the dimensionality of φ.

| Dataset | CUB200-2011[56] | CARS196[29] | SOP[38] |
|---------|-----------------|--------------|---------|
| Approach | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI |
| HTG[63] | 512 | 59.5 | 71.8 | 81.3 | - | 76.5 | 84.7 | 90.4 | - | - | - | - |
| HDML[65] | 512 | 53.7 | 65.7 | 76.7 | 62.6 | 79.1 | 87.1 | 92.1 | 69.7 | 68.7 | 83.2 | 92.4 | 89.3 |
| HTL[12] | 512 | 57.1 | 68.8 | 78.7 | - | 81.4 | 88.0 | 92.7 | - | 74.8 | 88.3 | 94.8 | - |
| DVML[31] | 512 | 52.7 | 65.1 | 75.5 | 61.4 | 82.0 | 88.4 | 93.3 | 67.6 | 70.2 | 85.2 | 93.8 | 90.8 |
| A-BIER[39] | 512 | 57.5 | 68.7 | 78.3 | - | 82.0 | 89.0 | 93.2 | - | 74.2 | 86.9 | 94.0 | - |
| MIC[45] | 128 | 66.1 | 76.8 | 85.6 | 69.7 | 82.6 | 89.1 | 93.2 | 68.4 | 77.2 | 89.4 | 95.6 | 90.0 |
| D&C[47] | 128 | 65.9 | 76.6 | 84.4 | 69.6 | 84.6 | 90.7 | 94.1 | 70.3 | 75.9 | 88.4 | 94.9 | 90.2 |
| Margin[60] | 128 | 63.6 | 74.4 | 83.1 | 69.0 | 79.6 | 86.5 | 90.1 | 69.1 | 72.7 | 86.2 | 93.8 | 90.8 |
| Ours (Margin[60] + PADS) | 128 | 67.3 | 78.0 | 85.9 | 69.9 | 83.5 | 89.7 | 93.8 | 68.8 | 76.5 | 89.0 | 95.4 | 89.9 |

Significant increase in network parameter:

| Ensemble Methods: | HORDE[23]+contrastive loss[17] | 512 | 66.3 | 76.7 | 84.7 | - | 83.9 | 90.3 | 94.1 | - | - | - | - |
| | SOFT-TRIPLE[43] | 512 | 65.4 | 76.4 | 84.5 | - | 84.5 | 90.7 | 94.5 | 70.1 | 78.3 | 90.3 | 95.9 | 92.0 |

Transfer of πθ and p(I_n|I_a): Tab. 3 investigates how well a trained policy πθ or final sampling distribution p(I_n|I_a) from a reference run transfer to differently (≠) or equally (=) initialized training runs. We find that applying a fixed trained policy (fix πθ) to a new training run with the same network initialization (=) improves performance by 0.4% due to the immediate utility of πθ for learning φ as πθ is already fully adapted to the reference learning process. In contrast, applying the trained policy to a differently initialized training run (≠) drops performance by 1.5%. Since the fixed πθ cannot adapt to the learning states of the new model, its support for optimizing φ is diminished. Note that the policy has only been trained on a single training run, thus it cannot fully generalize to different training dynamics. This shows the importance of an adaptive sampling.

Next, we investigate if the distribution p(I_n|I_a) obtained at the end of training can be regarded as an optimal sampling distribution over d_{an}, as πθ is fully trained. To this end we fix and apply the distribution p(I_n|I_a) after its last adjustment by πθ (fix last p(I_n|I_a)) in training the reference run. As intuitively expected, in both cases performance drops strongly as (i) we now have a static sampling process...
where the distribution is gradually shifted towards harder
with basic curriculum learning strategies, we pre-define two

Curriculum Learning:
To compare our adaptive sampling
results in Tab 1-2 are obtained with our best parameter
change with different parameter settings. In contrast, the
datasets further demonstrates the difficulty of finding broadly


Recall@1 NMI
64.7 67.5
65.7 69.2
64.8 68.2
63.7 67.5

(a) Varying the interval \(U = [\lambda_{\text{min}}, \lambda_{\text{max}}]\) of distances \(d_{\text{mn}}\) used for
learning \(p(I_n | I_a)\). The number of bins \(u_k\) is kept fixed to \(K = 30\).

| Num. bins \(K\) | 10 | 30 | 50 | 100 |
|-----------------|----|----|----|-----|
| Recall@1        | 63.8 | 65.7 | 65.3 | 64.9 |
| NMI             | 67.8 | 69.2 | 68.7 | 68.6 |

(b) Varying the number of bins \(u_k\) used to discretize the range of
distances \(U = [0.1, 1.4]\) used for learning \(p(I_n | I_a)\).

| Init. Distr.     | \(U_{[0,1.4]}\) | \(N(0.5, 0.05)\) | \(U_{[0,3,0.7]}\) |
|------------------|----------------|----------------|----------------|
| Recall@1         | 63.9 | 65.0 | 65.7 |
| NMI              | 67.0 | 68.6 | 69.2 |

(c) Comparison of \(p(I_n | I_a)\)-initializations on distance interval \(U = [0.1, 1.4]\). \(U_{[a,b]}\) denotes uniform emphasis in \([a, b]\) with low probabilities outside the interval. \(N(\mu, \sigma)\) denotes a normal distribution.

Table 5: Ablation experiments analyzing various parameters
for learning \(p(I_n | I_a)\).

and (ii) the sampling distribution is optimized to a specific
training state. Given our strong results, this proves that our
sampling process indeed adapts to the learning of \(\phi\).

Curriculum Learning: To compare our adaptive sampling
with basic curriculum learning strategies, we pre-define two
sampling schedules: (1) A linear increase of negative hard-
ness, starting from a semi-hard distance interval\([49]\) and (2)
a non-linear schedule using distance-based sampling\([60]\),
where the distribution is gradually shifted towards harder
negatives. We visualize the corresponding progression of the
sampling distribution in the supplementary material. Tab. 4
illustrates that both fixed, pre-defined curriculum schedules
perform worse than our learned, adaptive sampling distribu-
tion by at least 3.6% on CUB200-2011. On CARS196
the performance gap is even larger. The strong difference in
datasets further demonstrates the difficulty of finding broadly
applicable, effective fixed sampling strategies.

5.3. Ablation studies

Subsequently we ablate different parameters for learning
our sampling distribution \(p(I_n | I_a)\) on the CUB200-2011
dataset. More ablations are shown in the appendix. To make
the following experiments comparable, no learning rate
scheduling was applied, as convergence may significantly
change with different parameter settings. In contrast, the
results in Tab 1-2 are obtained with our best parameter
settings and a fixed learning rate scheduling. Without
scheduling, our best parameter setting achieves a recall
value of 65.7 and NMI of 69.2 on CUB200-2011.

Distance interval \(U\): As presented in Sec. 4.1, \(p(I_n | I_a)\) is
defined on a fixed interval \(U = [\lambda_{\text{min}}, \lambda_{\text{max}}]\) of distances.
Similar to other works\([60, 18]\), this allows us to additionally
regularize the sampling process by clipping the tails of the
true range of distances \([0, 2]\) on \(\Phi\). Tab. 5 (a) compares
different combinations of \(\lambda_{\text{min}}, \lambda_{\text{max}}\).
We observe that, while each option leads to significant performance boost
compared to the static sampling strategies, an interval
\(U = [0.1, 1.4]\) results in the most effective sampling process.

Number of bins \(K\): Next, we analyze the impact of the \(U\)
resolution in Tab. 5 (b), i.e. the number of bins \(K\). This
affects the flexibility of \(p(I_n | I_a)\), but also the complexity
of the actions \(a\) to be predicted. As intuitively expected,
increasing \(K\) allows for better adaption and performance
until the complexity grows too large.

Initialization of \(p(I_n | I_a)\): Finally, we analyze how the ini-
tialization of \(p(I_n | I_a)\) impacts learning. Tab. 5 (c) compares
the performance using different initial distributions, such
as a uniform random initialization (i.e. \( \mathcal{U}_{[0,1.4]} \)),
emphasizing semi-hard negatives \(I_a\) early on \( \mathcal{U}_{[0,3,0.7]} \)
or a proxy to \([60] (N(0.5, 0.05))\). We observe
that our learned sampling process benefits from a meaning-
ful, but generic initial configuration of \(p(I_n | I_a)\), \( \mathcal{U}_{[0,3,0.7]} \),
to effectively adapt the learning process of \(\phi\).

6. Conclusion

This paper presents a learned adaptive triplet sampling
strategy using Reinforcement Learning. We optimize
a teacher network to adjust the negative sampling distribu-
tion to the ongoing training state of a DML model. By
training the teacher to directly improve the evaluation metric
on a held-back validation set, the resulting training signal
optimally facilitates DML learning. Our experiments show
that our adaptive sampling strategy improves significantly
over static sampling distributions. Thus, even though only
built on top of basic triplet losses, we achieve competitive or
even superior performance compared to the state-of-the-art
of DML on multiple standard benchmarks sets.

Acknowledgements

We thank David Yu-Tung Hui (MILA) for valuable ins-
ights regarding the choice of RL Methods. This work has
been supported in part by Bayer AG, the German federal
ministry BMWi within the project “KI Absicherung”, and
a hardware donation from NVIDIA corporation.
References

[1] The sphere game in n dimensions. http://faculty.madisoncollege.edu/alehnen/sphere/hypers.htm., 2017. 3

[2] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Plau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In Advances in Neural Information Processing Systems. 2016. 3

[3] Miguel A Bautista, Artisom Sanakoyeu, Ekaterina Tikhoncheva, and Bjorn Ommer. Cliquecnn: Deep unsupervised exemplar learning. In Advances in Neural Information Processing Systems, pages 3846–3854, 2016. 1

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In International Conference on Machine Learning, 2009. 2, 3

[5] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In Proceedings of the 34th International Conference on Machine Learning, 2017. 2

[6] U. Büchler, B. Brattoli, and Bjorn Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 3

[7] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 2

[8] Huseyin Coskun, David Joseph Tan, Sailesh Conjeti, Nasir Navab, and Federico Tombari. Human motion analysis with deep metric learning. In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 1, 2

[9] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 2

[10] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. Learning what data to learn, 2017. 3, 5

[11] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2

[12] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In Proceedings of the IEEE Conference on Computer Vision (ECCV), pages 269–285, 2018. 2, 7, 13

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. JMLR Proceedings, 2010. 3

[14] Siddharth Gopal. Adaptive sampling for sgd by exploiting side information. In International Conference on Machine Learning, 2016. 2

[15] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In International Conference on Machine Learning, 2017. 2

[16] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. 2019. 2, 3

[17] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2006. 1, 2, 3, 7, 13

[18] 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, pages 2821–2829, 2017. 2, 3, 8

[19] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017. 13

[20] J. Hu, J. Lu, and Y. Tan. Discriminative deep metric learning for face verification in the wild. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014. 2

[21] Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Ángel Bautista, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss align- ment. In ICML, 2019. 3, 5

[22] Sergey Joffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal co-variate shift. International Conference on Machine Learning, 2015. 5, 12

[23] Pierre Jacob, David Picard, Aymier Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2, 6, 7, 13

[24] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence, 33(1):117–128, 2011. 4, 5, 6

[25] 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), 2018. 7, 13

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 12

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR), 2013. 2

[28] Dmytro Kotovenko, Artisom Sanakoyeu, Sabine Lang, and Björn Ommer. Content and style disentanglement for artistic style transfer. In Proceedings of the Intl. Conf. on Computer Vision (ICCV), 2019. 1

[29] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 554–561, 2013. 6, 7, 12, 13

[30] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, Advances in Neural Information Processing Systems 20, pages 817–824. Curran Associates, Inc., 2008. 5

[31] Xudong Lin, Yueqi Duan, Qiuyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In The European Conference on Computer Vision (ECCV), September 2018. 2, 6, 7, 13
[63] 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), pages 501–517, 2018. 2, 7, 13

[64] Xuefei Zhe, Shifeng Chen, and Hong Yan. Directional statistics-based deep metric learning for image classification and retrieval. Pattern Recognition, 93, 2018. 2

[65] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2, 7, 13
Supplementary Material

This part contains supporting or additional experiments to the main paper, such as additional ablations and qualitative evaluations.

A. Additional Ablation Experiments

We now conduct further ablation experiments for different aspects of our proposed approach based on the CUB200-2011[56] dataset. Note, that like in our main paper we did not apply any learning rate scheduling for the results of our approach to establish comparable training settings.

Performance with Inception-BN: For fair comparison, we also evaluate using Inception-V1 with Batch-Normalization [22]. We follow the standard pipeline (see e.g. [37, 43]), utilizing Adam [26] with images resized and random cropped to 224x224. The learning rate is set to $10^{-5}$. We retain the size of the policy network and other hyperparameters. The results on CUB200-2011[56] and CARS196[29] are listed in Table 6. On CUB200, we achieve results competitive to previous state-of-the-art methods. On CARS196, we achieve a significant boost over baseline values and competitive performance to the state-of-the-art.

Validation set $I_{val}$: The validation set $I_{val}$ is sampled from the training set $I_{train}$, composed as either a fixed disjoint, held-back subset or repetitively re-sampled from $I_{train}$ during training. Further, we can sample $I_{val}$ across all classes or include entire classes. We found (Tab. 7 (d)) that sampling $I_{val}$ from each class works much better than doing it per class. Further, resampling $I_{val}$ provides no significant benefit at the cost of an additional hyperparameter to tune.

Composition of states $s$ and target metric $e$: Choosing meaningful target metrics $e(\phi(\cdot; \zeta), I_{val})$ for computing rewards $r$ and a representative composition of the training state $s$ increases the utility of our learned policy $\pi_\theta$. To this end, Tab. 8 compares different combinations of state compositions and employed target metrics $e$. We observe that incorporating information about the current structure of the embedding space $\Phi$ into $s$, such as intra- and inter-class distances, is most crucial for effective learning and adaptation. Moreover, also incorporating performance metrics into $s$ which directly represent the current performance of the model $\phi$, e.g. Recall@1 or NMI, additional adds some useful information.

Frequency of updating $\pi_\theta$: We compute the reward $r$ for an adjustment $a$ to $p(I_n|I_a)$ every $M$ DML training iterations. High values of $M$ reduce the variance of the rewards $r$, however, at the cost of slow policy updates which result in potentially large discrepancies to updating $\phi$. Tab. 9 (a) shows that choosing $M$ from the range [30, 70] results in a good trade-off between the stability of $r$ and the adaptation of $p(I_n|I_a)$ to $\phi$. Moreover, we also show the result for setting $M = \infty$, i.e. using the initial distribution throughout training without adaptation. Fixing this distribution performs worse than the reference method Margin loss with static distance-based sampling[60]. Nevertheless, frequently adjusting $p(I_n|I_a)$ leads to significant superior performance, which indicates that our policy $\pi_\theta$ effectively adapts $p(I_n|I_a)$ to the training state of $\phi$.

Importance of long-term information for states $s$: For optimal learning, $s$ should not only contain information about the current training state of $\phi$, but also about some history of the learning process. Therefore, we compose $s$ of a set of running averages over different lengths $R$ for various training state components, as discussed in the implementation details of the main paper. Tab. 9 (b) confirms the importance of long-term information for stable adaptation and learning. Moreover, we see that the set of moving averages $R = \{2, 8, 16, 32\}$ works best.

B. Curriculum Evaluations

In Fig. 5 we visually illustrate the fixed curriculum schedules which we applied for the comparison experiment in Sec. 5.3 of our main paper. We evaluated various schedules - Linear progression of sampling intervals starting at semi-hard negatives going to hard negatives, and progressively moving $\mathcal{U}$-dist[60] towards harder negatives. The schedules visualized were among the best performing ones to work for both CUB200 and CARS196 dataset.

C. Comparison of RL Algorithms

We evaluate the applicability of the following RL algorithms for optimizing our policy $\pi_\theta$ (Eq. 4 in the main paper):

- REINFORCE algorithm[59] with and without Exponential Moving Average (EMA)
Table 6: Comparison to the state-of-the-art DML methods on CUB200-2011[56] and CARS196[29] using the Inception-BN Backbone (see e.g. [37, 43]) and embedding dimension of 512.

| Dataset          | CUB200-2011[56] | CARS196[29] |
|------------------|----------------|-------------|
| **Approach**     | Dim | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI |
| HTG[63]          | 512 | 59.5 | 71.8 | 81.3 | -   | 76.5 | 84.7 | 90.4 | -   |
| HDML[65]         | 512 | 53.7 | 65.7 | 76.7 | 62.6 | 79.1 | 87.1 | 92.1 | 69.7 |
| HTL[12]          | 512 | 57.1 | 68.8 | 78.7 | -   | 81.4 | 88.0 | 92.7 | -   |
| DVML[31]         | 512 | 52.7 | 65.1 | 75.5 | 61.4 | 82.0 | 88.4 | 93.3 | 67.6 |
| A-BIER[39]       | 512 | 57.5 | 68.7 | 78.3 | -   | 82.0 | 89.0 | 93.2 | -   |
| MIC[45]          | 128 | 66.1 | 76.8 | 85.6 | -   | 82.6 | 89.1 | 93.2 | 68.4 |
| D&G[47]          | 128 | 65.9 | 76.6 | 84.4 | 69.6 | 84.6 | 90.7 | 94.1 | 70.3 |
| Margin[60]       | 128 | 63.6 | 74.4 | 83.1 | -   | 79.6 | 86.5 | 90.1 | 69.1 |
| Reimpl. Margin[60], IBN | 512 | 63.8 | 75.3 | 84.7 | 67.9 | 79.7 | 86.9 | 91.4 | 67.2 |
| Ours(Margin[60] + PADS, IBN) | 512 | 66.6 | 77.2 | 85.6 | 68.5 | 81.7 | 88.3 | 93.0 | 68.2 |

Significant increase in network parameter:

| Approach          | Dim | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI |
|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| HORDE[23]+Contr.[17] | 512 | 66.3 | 76.7 | 84.7 | -   | 83.9 | 90.3 | 94.1 | -   |
| SOFT-TRIPLE[43]   | 512 | 65.4 | 76.4 | 84.5 | -   | 84.5 | 90.7 | 94.5 | 70.1 |

Ensemble Methods:

| Approach          | Dim | R@1 | R@2 | R@4 | NMI | R@1 | R@2 | R@4 | NMI |
|-------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Rank[38]          | 1536 | 61.3 | 72.7 | 82.7 | 66.1 | 82.1 | 89.3 | 93.7 | 71.8 |
| DREML[62]         | 9216 | 63.9 | 75.0 | 83.1 | 67.8 | 86.0 | 91.7 | 95.0 | 76.4 |
| ABE[25]           | 512 | 60.6 | 71.5 | 79.8 | -   | 85.2 | 90.5 | 94.0 | -   |

Table 7: Composition of $I_{val}$. Superscript $By/Per$ denotes usage of entire classes/sampling across classes. $R$ denotes re-sampling during training with best found frequency of 150 epochs.

| Validation Set: | $I_{val}^{By}$ | $I_{val}^{Per}$ | $I_{val}^{By,R}$ | $I_{val}^{Per,R}$ |
|----------------|----------------|----------------|------------------|------------------|
| Recall $@1$     | 62.6           | 65.7           | 63.0             | 65.8             |
| NMI             | 67.7           | 69.2           | 67.8             | 69.6             |

For a comparable evaluation setting we use the CUB200-2011[56] dataset without learning rate scheduling and fixed 150 epochs of training. Within this setup, the hyperparameters related to each method are optimized via cross-validation. Tab. 10 shows that all methods, except for vanilla Q-Learning, result in an adjustment policy $\pi_\theta$ for $p(I_n|I_a)$ which outperforms static sampling strategies. Moreover, policy-based methods in general perform better than Q-Learning based methods with PPO being the best performing algorithm. We attribute this to the reduced search space (Q-Learning methods need to evaluate in state-actions space, unlike policy-methods, which work directly over the action space), as well as not employing replay buffers, i.e. not
### Table 9: Ablation experiments: (a) evaluates the influence of the number of DML iterations $M$ performed before updating the policy $\pi_\theta$ using a reward $r$ and, thus, the update frequency of $\pi_\theta$. (b) analyzes the benefit of long-term learning progress information added to training states $s$ by means of using various moving average lengths $R$.

| $M$  | 10 | 30 | 50 | 70 | 100 | $\infty$ | 60 |
|------|----|----|----|----|-----|--------|----|
| R@1 | 64.4 | 65.7 | 65.2 | 65.1 | 61.9 | 63.5 |
| NMI | 68.3 | 69.2 | 68.9 | 69.0 | 67.0 | 68.1 |

(a) Evaluation of the policy update frequency $M$.

| $R$  | 2, 32 | 2, 8, 16, 32 | 2, 8, 16, 32, 64 |
|------|--------|----------------|------------------|
| R@1 | 64.5 | 65.7 | 65.6 |
| NMI | 68.6 | 69.2 | 69.3 |

(b) Evaluation of various sets $R$ of moving average lengths.

Table 10: Comparison of different RL algorithms. For policy-based algorithms (REINFORCE, PPO) we either use Exponential Moving Average (EMA) as a variance-reducing baseline or employ Advantage Actor Critic (A2C). In addition, we also evaluate Q-Learning methods (vanilla and Rainbow Q-Learning). For the Rainbow setup we use Priority Replay and 2-Step value approximation. Margin loss[60] is used as a representative reference for static sampling strategies.

| Approach         | R@1 | NMI |
|------------------|-----|-----|
| Margin[60]       | 63.5 | 68.1 |
| REINFORCE        | 64.2 | 68.5 |
| REINFORCE, EMA   | 64.8 | 68.9 |
| REINFORCE, A2C   | 65.0 | 69.0 |
| PPO, EMA         | 65.4 | 69.0 |
| PPO, A2C         | 65.7 | 69.2 |
| Q-Learn          | 63.2 | 67.9 |
| Q-Learn, PR/2-Step | 64.9 | 68.5 |

Table 10: Comparison of different RL algorithms. For policy-based algorithms (REINFORCE, PPO) we either use Exponential Moving Average (EMA) as a variance-reducing baseline or employ Advantage Actor Critic (A2C). In addition, we also evaluate Q-Learning methods (vanilla and Rainbow Q-Learning). For the Rainbow setup we use Priority Replay and 2-Step value approximation. Margin loss[60] is used as a representative reference for static sampling strategies.

Table 10: Comparison of different RL algorithms. For policy-based algorithms (REINFORCE, PPO) we either use Exponential Moving Average (EMA) as a variance-reducing baseline or employ Advantage Actor Critic (A2C). In addition, we also evaluate Q-Learning methods (vanilla and Rainbow Q-Learning). For the Rainbow setup we use Priority Replay and 2-Step value approximation. Margin loss[60] is used as a representative reference for static sampling strategies.

### D. Qualitative UMAP Visualization

Figure 6 shows a UMAP[34] embedding of test image features for CUB200-2011[56] learned by our model using PADS. We can see clear groupings for birds of the same and similar classes. Clusterings based on similar background is primarily due to dataset bias, e.g. certain types of birds occur only in conjunction with specific backgrounds.

**Figure 6: UMAP embedding** based on the image embeddings $\phi(\cdot; \zeta)$ obtained from our proposed approach on CUB200-2011[56] (Test Set).

### E. Pseudo-Code

Algorithm 1 gives an overview of our proposed PADS approach using PPO with A2C as underlying RL method. Before training, our sampling distributions $p(I_n|I_a)$ is initialized with an initial distribution. Further, we initialize both the adjustment policy $\pi_\theta$ and the pre-update auxiliary policy $\pi_\theta^{old}$ for estimating the PPO probability ratio. Then, DML training is performed using triplets with random anchor-positive pairs and sampled negatives from the current sampling distribution $p(I_n|I_a)$. After $M$ iterations, all reward and state metrics $E, E^*$ are computed on the embeddings $\phi(I_n; \zeta)$ of $I_{val}$. These values are aggregated in a training reward $r$ and input state $s$. While $r$ is used to update the current policy $\pi_\theta$, $s$ is fed into the updated policy to estimate adjustments $a$ to the sampling distribution $p(I_n|I_a)$. Finally, after $M^{old}$ iterations (e.g. we set to $M^{old} = 3$) $\pi_\theta^{old}$ is updated with the current policy weights $\theta$.

### F. Typical image retrieval failure cases

Fig. 7 shows nearest neighbours for good/bad test set retrievals. Even though the nearest neighbors do not always share the same class label as the anchor, all neighbors are very similar to the bird species depicted in the anchor images. Failures are due to very subtle differences.
Algorithm 1: Training one epoch via PADS by PPO

Input
$I_{\text{train}}, I_{\text{val}}, \text{Train labels } y_{\text{train}}, \text{Val. labels } y_{\text{val}}, \text{total iterations } n_e$

Parameter
: Reward metrics $\mathcal{E}$, State metrics $\mathcal{E}^* + \text{running average lengths } \mathcal{R}$, Num. of bins $K$, multiplier $\{\alpha, \beta\}$, $p_{\text{init}}(I_n \mid I_a)$, num. of iterations before updates $M$, $M^\text{old}$

// Initialization
$p(I_n \mid I_a) \leftarrow p_{\text{init}}(I_n \mid I_a)$
$\pi_0 \leftarrow \text{InitPolicy}(K, \alpha, \beta)$
$\pi_0^\text{old} \leftarrow \text{Copy}(\pi_0)$

for $i$ in $n_e / M$ do

// Update DML Model
for $j$ in $M$ do

// within batch $B \in I_{\text{train}}$
$I_a, I_p \leftarrow B$
$I_n \sim p(I_n \mid I_a)$
$\zeta \leftarrow \text{TrainDML}(\{I_a, I_p, I_n\}, \phi(\cdot; \zeta))$

end

// Update policy $\pi_\theta$
$E_i \leftarrow \mathcal{E}(I_{\text{val}}, y_{\text{val}}, \phi(\cdot; \zeta))$
$E_i^* \leftarrow \mathcal{E}^*(I_{\text{val}}, y_{\text{val}}, \phi(\cdot; \zeta))$
$s_i \leftarrow \text{GetState}(E_i^*, \mathcal{R}, p(I_n \mid I_a))$
$r \leftarrow \text{GetReward}(E_i, E_{i-1})$
$l_\pi \leftarrow \text{PPOLoss}(\pi_\theta, \pi_\theta^\text{old}, s_{i-1}, a_{i-1})$
$\theta \leftarrow \text{UpdatePolicy}(l_\pi, \pi_\theta)$
$a_i \sim \pi_\theta(a_i \mid s_i)$
$p(I_n \mid I_a) \leftarrow \text{Adjust}(p(I_n \mid I_a), a_i)$

if $i \bmod M^\text{old} = 0$ then

$\pi_\theta^\text{old} \leftarrow \text{Copy}(\pi_\theta)$

end

end

Figure 7: Selection of good and bad nearest neighbour retrieval cases on CUB200-2011 (Test). Orange bounding box marks query images, green/red boxes denote correct/incorrect retrievals.