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[57, 50, 36, 33], pose understanding [32, 8, 3, 51], face detection[46] and style transfer [26]. 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, 50, 35] or contrasted against a third image[46, 57, 54] to learn a distance metric that reflects image similarity. Such triplet learning constitutes the basis of powerful learning algorithms[42, 36, 44, 59]. However, with growing training
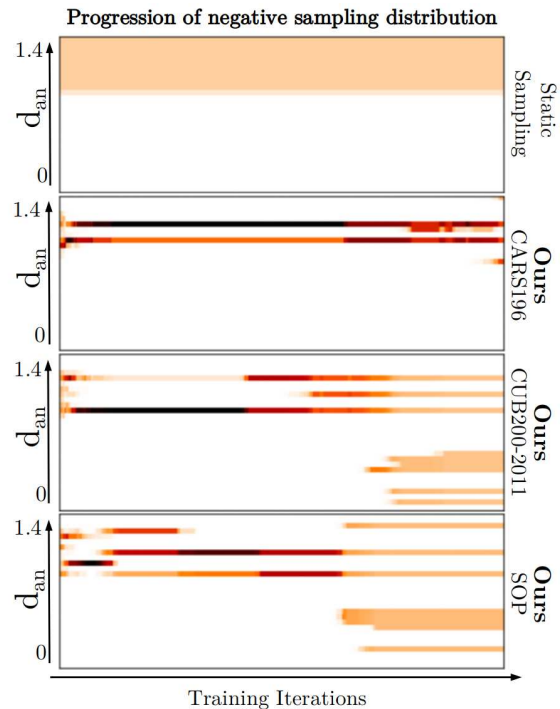

Figure 1: *Progression of negative sampling distributions over training iterations.* A static sampling strategy[57] follows a fixed probability distribution over distances d_{an} between anchor and negative images. In contrast, our learned, discretized sampling distributions change while adapting to the training state of the DML model. This leads to improvements on all datasets close to 4% compared to static strategies (cf. Tab. 1). Moreover, the progression of the adaptive distributions varies between datasets and, thus, is difficult to model manually which highlights the need for a learning based approach.

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*

---

[*]Authors contributed equally to this work.

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[46, 9, 60] but therefore informative, impairs optimization due to high gradient variance[57]. 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[57] and is a typical remedy in representation learning in general[25, 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 adjusts 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[57].

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[34, 29, 57], image classification [11, 61], face verification[46, 19, 30] or human pose analysis[32, 8]. Ranking losses formulated on pairs[50, 17], triplets[46, 57, 54, 12] or even higher order tuples of images[7, 35, 55] emerged as the most widely used

basis for DML [43]. 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[46, 57, 18] based on distances between an anchor and a negative image. FaceNet[46] leverages only the hard negatives in a mini-batch. Wu *et al*. [57] 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[62, 60]. 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[34] or more [40] 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 [36, 59, 44] learn and combine multiple embedding spaces to capture more information. HORDE[22] additionally forces feature representations of related images to have matching higher moments. Roth *et al*. [42] combines class-discriminative features with features learned from characteristics shared across classes. Similarly, Lin *et al*. [29] 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*. [45] learn an importance sampling over
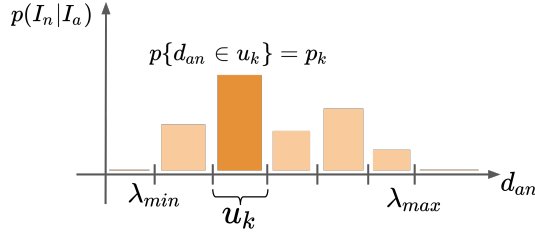
Figure 2: *Sampling distribution $p(I_n|I_a)$. We discretize the distance interval $U = [\lambda_{\min}, \lambda_{\max}]$ into K equisized bins u_k with individual sampling probabilities p_k.*

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[41], input masking[10], self-supervision [6], finetuning [49], loss functions[20], optimizer parameters[2] and model architectures[39, 58]. 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 ζ. Further, ϕ is normalized to a unit hypersphere $\mathbb{S}$ for regularization purposes [46]. Thus, the objective of DML is to learn $\phi : \mathbb{R}^{H \times W \times 3} \to \Phi \subseteq \mathbb{S}$ 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 ϕ are ranking losses[46, 57, 50, 35, 35, 17] operating on tuples of images. Their most widely used representative is arguably the triplet loss[46] which is defined as an ordering task between images $\{I_a, I_p, I_n\}$, formulated as

$$\mathcal{L}_{\text{triplet}}(\{I_a, I_p, I_n\}; \zeta) = \max(0, d_{ap}^2 - d_{an}^2 + \gamma) \quad (1)$$

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 γ 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 γ as training progresses. Naively employing random triplet sampling entails many of the selected triplets being uninformative, as distances on Φ are strongly biased towards larger distances d due to its regularization to $\mathbb{S}$. 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 γ 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[46] due to a strong focus on few data outliers and very hard negatives. Facenet[46] 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[37, 50] 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[57]. 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.* [57] 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 Φ 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 Φ: $q(d) \propto d^{D-2} \left[1 - \frac{1}{4}d^2\right]^{\frac{D-3}{2}}$ for large $D \geq 128$[1]. Distance-based sampling from the static, uniform prior is then performed by

$$I_n \sim p(I_n|I_a) \propto \min\left(\lambda, q^{-1}(d_{an})\right) \quad (2)$$

with λ 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

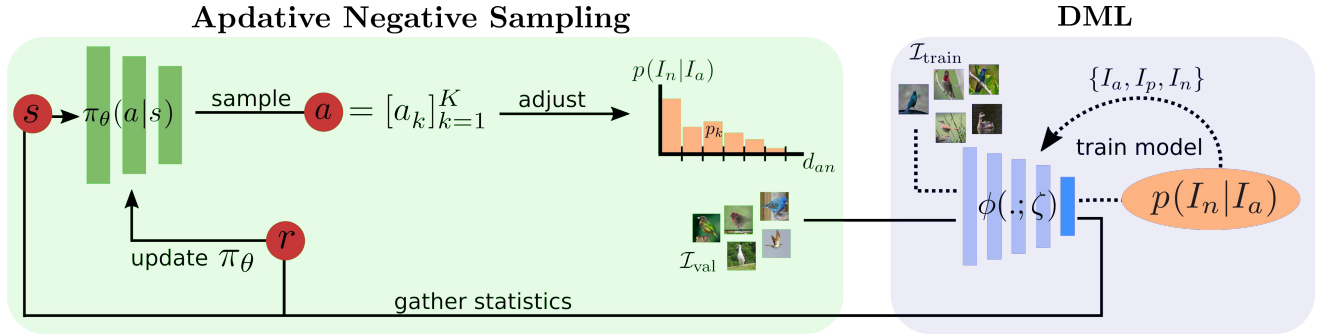**Apdative Negative Sampling**   **DML**

Figure 3: *Overview of approach.* Blue denotes the standard Deep Metric Learning (DML) setup using triplets $\{I_a, I_p, I_n\}$. Our proposed adaptive negative sampling is shown in green: (1) We compute the current training state s using $\mathcal{I}_{val}$. (2) Conditioned on s, our policy $\pi_\theta(a|s)$ predicts adjustments to p_k. (3) We perform bin-wise adjustments of $p(I_n|I_a)$. (4) Using the adjusted $p(I_n|I_a)$ we train the DML model. (5) Finally, π_θ is updated based on the reward r.

epochs or iterations. To provide an optimal training signal, however, we rather want $p(I_n|I_a)$ to adapt to the training state of the DML model than merely the training iteration. Such an adaptive negative sampling allows for adjustments which directly facilitate maximal DML performance. Since manually designing such a strategy is difficult, *learning* it is the most viable option.

Subsequently, we first present how to find a parametrization of $p(I_a|I_n)$ that is able to represent arbitrary, potentially multi-modal distributions, thus being able to sample negatives I_n of any mixture of difficulty needed. Using this, we can learn a policy which effectively alters $p(I_n|I_a)$ to optimally support learning of the DML model.

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, χ^2) are insufficient. Thus, we utilize a discrete probability mass function $p(I_n|I_a) := Pr\{d_{an} \in u_k\} = p_k$, where the bounded interval $U = [\lambda_{\min}, \lambda_{\max}]$ of possible distances d_{an} 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 ϕ. As ϕ 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 ϕ. We subsequently solve this task by learning a stochastic adjustment policy π_θ for the p_k, implemented as a neural network parametrized by θ.

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 ϕ at every stage of train-

ing. Thus, we adjust the p_k by a multiplicative update $a \in \mathcal{A}$ conditioned on the current representation (or state) $s \in \mathcal{S}$ of ϕ during learning. We introduce a conditional distribution $\pi_\theta(a|s)$ to control which adjustment to apply at which state s of training ϕ. To learn π_θ, we measure the utility of these adjustments for learning ϕ 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 π_θ alongside ϕ.

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 $\mathcal{A} = \{\alpha, 1, \beta\}$ 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, α, β are fixed constants $0 < \alpha < 1, \beta > 1$ and $\frac{\alpha+\beta}{2} = 1$. 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 Φ (cf. Sec. 3.1).

Training states s**:** Adjustments a depend on the present state $s \in \mathcal{S}$ of the representation ϕ. Unfortunately, we cannot use the current model weights ζ of the embedding network, as the dimensionality of s would be to high, thus making optimization of π_θ infeasible. Instead, we represent the current training state using representative statistics describing the learning progress: running averages over Recall@1[23], NMI[31] and average distances between and within classes on a fixed held-back validation set $\mathcal{I}_{val}$. 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 ϕ while learning. Thus, we compute the reward r for 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 $e(\phi(.;\zeta_t), \mathcal{I}_{\mathrm{val}})$ on the validation set $\mathcal{I}_{\mathrm{val}}$. More precisely, we define r as

$$r = \mathrm{sign}\left(e(\phi(.;\zeta), \mathcal{I}_{\mathrm{val}}) - e(\phi(.;\zeta'), \mathcal{I}_{\mathrm{val}})\right) \qquad (3)$$

where ζ was reached from ζ' after M DML training iterations using $p(I_n|I_a)$. We choose e to be the sum of Recall@1[23] and NMI[31]. Both metrics are in the range $[0, 1]$ and target slightly different performance aspects. Further, similar to [20], we utilize the sign function for consistent learning signals even during saturated training stages.

Learning of π_θ: 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 $R(\tau) = \sum_t r_t(a_t, s_t)$ 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.

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[R(\tau)] \qquad (4)$$

Hence, π_θ is optimized to predict adjustments a for $p(I_n|I_a)$ which yield high rewards and thereby improving the performance of ϕ. Common approaches use episodes τ 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 ϕ, this discrepancy needs to be minimized. In fact, our experiments show that single-step episodes, i.e. $T = 1$, are sufficient for optimizing π_θ to infer meaningful adjustments a for $p(I_n|I_a)$. Such a setup is also successfully adopted by contextual bandits [28] [1]. In summary, our training episodes τ 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 π_θ 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 $G(s, a)$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(a|s)G(s,a)\right] \qquad (5)$$

The choice of the exact form of $G = G(s, a)$ gives rise to different optimization methods, e.g REINFORCE[56] ($G = R(\tau)$), Advantage Actor Critic (A2C)[52] ($G = A(s, a)$), etc. Other RL algorithms, such as TRPO[47] or PPO[48] replace Eq. 4 by surrogate objective functions. Fig. 3 provides an overview over the learning procedure. Moreover,

---

[1]Opposed to bandits, in our RL setup, actions which are sampled from π_θ influence *future* training states of the learner. Thus, the policy implicitly learns state-transition dynamics.

in the supplementary material we compare different RL algorithms and summarizes the learning procedure in Alg. 1 using PPO[48] for policy optimization.

Initialization of $p(I_n|I_a)$: We find that an initialization with a slight emphasis towards smaller distances d_{an} works best. However, as shown in Tab. 5, also other initializations work well. In addition, the limits of the distance interval $U = [\lambda_{\min}, \lambda_{\max}]$ can be controlled for additional regularization as done in [57]. This means ignoring values above $\lambda_{\max}$ and clipping values below $\lambda_{\min}$, which is analysed in Tab. 5.

Self-Regularisation: As noted in [42], 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[46, 57] comes from the estimation of r requiring a forward pass over $\mathcal{I}_{\mathrm{val}}$ 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 [57] with ResNet50. During training, images are resized to 256×256 with random crop to 224×224 and random horizontal flipping. For completeness, we also evaluate on Inception-BN [21] following standard practice in the supplementary. The initial learning rates are set to 10^{-5}. We choose triplet parameters according to [57], with $\gamma = 0.2$. For margin loss, we evaluate margins $\beta = 0.6$ and $\beta = 1.2$. Our policy π is implemented as a two-layer fully-connected network with ReLU-nonlinearity 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 $[\lambda_{\min}, \lambda_{\min}]$ 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[53]				CARS196[27]				SOP[35]			
Approach	Dim	R@1	R@2	R@4	NMI	R@1	R@2	R@4	NMI	R@1	R@10	R@100	NMI
Margin[57] + $\mathcal{U}$-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[57] + $\mathcal{U}$-dist (ReImp, $\beta = 1.2$)	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[57] + $\mathcal{U}$-dist (ReImp, $\beta = 0.6$)	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[57] + *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[46] + 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[46] + semihard (ReImp)	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[46] + $\mathcal{U}$-dist (ReImp)	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[46] + *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[35] (*semihard*) and static distance-based sampling ($\mathcal{U}$-*dist*)[57] using triplet[46] and margin loss[57]. *ReImp.* denotes our re-implementations and *Dim* the dimensionality of ϕ.

PyTorch framework[38] 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 $\mathcal{I}_{val}$. We use: *CARS196*[27], with 16,185 images from 196 car classes. *CUB200-2011*[53], 11,788 bird images from 200 classes. *Stanford Online Products (SOP)*[35], 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[46] and margin loss[57]. For each loss, we compare against the most commonly used static sampling strategies, semi-hard[46]
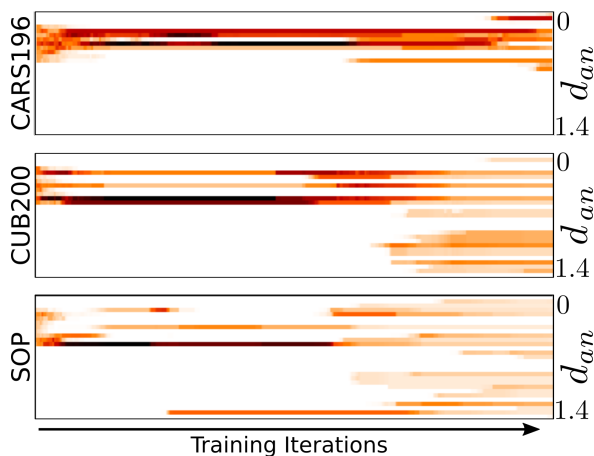

Figure 4: *Averaged progression of $p(I_n|I_a)$ over multiple training runs on CUB200-2011, CARS196 and SOP.*

(semihard) and distance-based sampling[57] ($\mathcal{U}$-dist) on the CUB200-2011, CARS196 and SOP dataset. We measure image retrieval performance using recall accuracy R@k[23] following [36]. For completeness we additonally show the normalized mutual information score (NMI)[31], 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[42], DVML[29], HORDE[22], A-BIER[36]), ensembles of embedding spaces (DREML[59], D&C[44], Rank[55]) and/or significantly more network parameters (HORDE[22], SOFT-TRIPLE[40]). 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[27] we rank second behind the top performing non-ensemble method D&C[44]. On SOP[35] we lose 0.7% to MIC[42] 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

Dataset		CUB200-2011[53]				CARS196[27]				SOP[35]			
Approach	Dim	R@1	R@2	R@4	NMI	R@1	R@2	R@4	NMI	R@1	R@2	R@4	NMI
HTG[60]	512	59.5	71.8	81.3	-	76.5	84.7	90.4	-	-	-	-	-
HDML[62]	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[29]	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[36]	512	57.5	68.7	78.3	-	82.0	89.0	93.2	-	74.2	86.9	94.0	-
MIC[42]	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[44]	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[57]	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[57] + *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:													
HORDE[22]+contrastive loss[17]	512	66.3	76.7	84.7	-	83.9	90.3	94.1	-	-	-	-	-
SOFT-TRIPLE[40]	512	65.4	76.4	84.5	-	84.5	90.7	94.5	70.1	78.3	90.3	95.9	**92.0**
Ensemble Methods:													
Rank[55]	1536	61.3	72.7	82.7	66.1	82.1	89.3	93.7	71.8	**79.8**	**91.3**	**96.3**	90.4
DREML[59]	9216	63.9	75.0	83.1	67.8	**86.0**	**91.7**	**95.0**	**76.4**	-	-	-	-
ABE[24]	512	60.6	71.5	79.8	-	85.2	90.5	94.0	-	76.3	88.4	94.8	-

Table 2: Comparison to the state-of-the-art DML methods on CUB200-2011[53], CARS196[27] and SOP[35]. *Dim* denotes the dimensionality of ϕ.

runs with different network initializations. While on CARS196 the distribution $p(I_n|I_a)$ strongly emphasizes smaller distances d_{an}, we observe on CUB200-2011 and SOP generally a larger variance of $p(I_n|I_a)$. Further, on each dataset, during the first half of training $p(I_n|I_a)$ quickly peaks on a sparse set of bins u_k, as intuitively expected, since most triplets are still informative. As training continues, $p(I_n|I_a)$ begins to yield both harder and easier negatives, thus effectively sampling from a wider distribution. This observation confirms the result of Wu *et al.* [57] which proposes to ease the large gradient variance introduced by hard negatives with also adding easier negatives. Moreover, for each dataset we observe a different progression of $p(I_n|I_a)$ which indicates that manually designing similar sampling strategies is difficult, as also confirmed by our results in Tab. 1 and 4.

Transfer of π_θ and $p(I_n|I_a)$: Tab. 3 investigates how well a trained policy π_θ or final sampling distribution $p(I_n|I_a)$ from a reference run transfer to differently ($\neq$) or equally

(=) initialized training runs. We find that applying a fixed trained policy (*fix π_θ*) to a new training run with the same network initialization (=) improves performance by 0.4% due to the immediate utility of π_θ for learning ϕ as π_θ is already fully adapted to the reference learning process. In contrast, applying the trained policy to a differently initialized training run ($\neq$) drops performance by 1.5%. Since the fixed π_θ 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 π_θ is fully trained. To this end we fix and apply the distribution $p(I_n|I_a)$ after its last adjustment by π_θ (*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

	Init.	Reference	fix π_θ	fix last $p(I_n|I_a)$
R@1	$\neq$	**65.4**	64.3	59.0
R@1	=	65.4	**65.8**	57.6

Table 3: Transferring a fixed trained policy π_θ and fixed final distribution $p(I_n|I_a)$ to training runs with different ($\neq$) and the same network initialization (=). *Reference* denotes the training run from which π_θ and $p(I_n|I_a)$ is obtained.

Dataset	CUB200-2011[53]		CARS196[27]	
Metrics	R@1	NMI	R@1	NMI
Ours	**67.3**	**69.9**	**83.5**	**68.8**
linear CL	59.1	63.1	72.2	64.0
non-linear CL	63.6	68.4	78.1	66.8

Table 4: Comparison to curriculum learning strategies with predefined linear and non-linear progression of $p(I_n|I_a)$.

$[\lambda_{\mathbf{min}}, \lambda_{\mathbf{max}}]$	$[0, 2]$	$[0.1, 1.4]$	$[0.25, 1.0]$	$[0.5, 1.4]$
Recall@1	64.7	**65.7**	64.8	63.7
NMI	67.5	**69.2**	68.2	67.5

(a) Varying the interval $U = [\lambda_{\min}, \lambda_{\max}]$ of distances d_{an} 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.	$\mathcal{U}_{[0.1,1.4]}$	$\mathcal{N}(0.5, 0.05)$	$\mathcal{U}_{[0.3,0.7]}$
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 $U = [0.1, 1.4]$. $\mathcal{U}_{[a,b]}$ denotes uniform emphasis in $[a, b]$ with low probabilities outside the interval. $\mathcal{N}(\mu, \sigma)$ 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 ϕ.

Curriculum Learning: To compare our adaptive sampling with basic curriculum learning strategies, we pre-define two sampling schedules: *(1)* A *linear* increase of negative hardness, starting from a semi-hard distance interval[46] and *(2)* a *non-linear* schedule using distance-based sampling[57], 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 distribution 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, $\tilde{p}(I_n|I_a)$ is defined on a fixed interval $U = [\lambda_{\min}, \lambda_{\max}]$ of distances. Similar to other works[57, 18], this allows us to additionally regularize the sampling process by clipping the tails of the true range of distances $[0, 2]$ on Φ. Tab. 5 (a) compares different combinations of $\lambda_{\min}, \lambda_{\max}$. 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 initialization of $p(I_n|I_a)$ impacts learning. Tab. 5 (c) compares the performance using different initial distributions, such as a neutral uniform initialization (i.e. random sampling) ($\mathcal{U}_{[0.1,1.4]}$), emphasizing semi-hard negatives I_n early on ($\mathcal{U}_{[0.3,0.7]}$) or a proxy to [57] ($\mathcal{N}(0.5, 0.05)$). We observe that our learned sampling process benefits from a meaningful, but generic initial configuration of $p(I_n|I_a)$, $\mathcal{U}_{[0.3,0.7]}$, to effectively adapt the learning process of ϕ.

6. Conclusion

This paper presents a learned adaptive triplet sampling strategy using Reinforcement Learning. We optimize a teacher network to adjust the negative sampling distribution 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

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 Pfau, 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, Artsiom 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 Björn 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, Nassir 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 European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 2, 7

[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

[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] 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

[20] 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 alignment. In *ICML*, 2019. 3, 5

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 2015. 5

[22] Pierre Jacob, David Picard, Aymeric 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

[23] 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

[24] 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

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

[26] Dmytro Kotovenko, Artsiom 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

[27] 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

[28] 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

[29] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2, 6, 7

[30] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[31] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010. 4, 5, 6

[32] Timo Milbich, Miguel Bautista, Ekaterina Sutter, and Björn Ommer. Unsupervised video understanding by reconciliation of posture similarities. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[33] Timo Milbich, Omair Ghori, Ferran Diego, and Björn Ommer. Unsupervised representation learning by discovering reliable image relations. *Pattern Recognition (PR)*, 102, June 2020. 1

[34] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017. 2

[35] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 1, 2, 3, 6, 7

[36] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 2, 6, 7

[37] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 3

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[39] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *International Conference on Machine Learning*, 2018. 3

[40] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. 2019. 2, 6, 7

[41] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. 3

[42] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8000–8009, 2019. 1, 2, 5, 6, 7

[43] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning, 2020. 2

[44] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 6, 7

[45] Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. Data parameters: A new family of parameters for learning a differentiable curriculum. In *Advances in Neural Information Processing Systems*. 2019. 2

[46] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 1, 2, 3, 5, 6, 8

[47] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015. 5

[48] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, 2017. 5

[49] Tianlin Shi, Jacob Steinhardt, and Percy Liang. Learning where to sample in structured prediction. In *Artificial Intelligence and Statistics*, 2015. 3

[50] Kihyuk Sohn. Improved deep metric learning with multiclass n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 1, 2, 3

[51] Ömer Sümer, Tobias Dencker, and Björn Ommer. Selfsupervised learning of pose embeddings from spatiotemporal relations in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[52] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998. 5

[53] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6, 7

[54] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 1, 2

[55] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M. Robertson. Ranked list loss for deep metric learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[56] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. 5

[57] 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*, pages 2840–2848, 2017. 1, 2, 3, 5, 6, 7, 8

[58] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. 2019. 3

[59] Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 723–734, 2018. 1, 2, 6, 7

[60] 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

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

[62] 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