Supplementary: Simultaneous Similarity-based Self-Distillation for Deep Metric Learning

A. More Benchmark & Implementation Details

In this part, we report all relevant benchmark details omitted in the main document as well as further implementation details.

A.1. Benchmarks

CUB200-2011 (Wah et al., 2011) contains 200 bird classes over 11,788 images, where the first and last 100 classes with 5864/5924 images are used for training and testing, respectively.
CARS196 (Krause et al., 2013) contains 196 car classes and 16,185 images, where again the first and last 98 classes with 8054/8131 images are used to create the training/testing split.
Stanford Online Products (SOP) (Oh Song et al., 2016) is build around 22,634 product classes over 120,053 images and contains a provided split: 11318 selected classes with 59551 images are used for training, and 11316 classes with 60502 images for testing.

A.2. Implementation

We now provide further details regarding the training and testing setup utilized. For any study except the comparison against the state-of-the-art (Table 2) which uses different backbones and embedding dimensions, we follow the setup used by (Roth et al., 2020b)[1]: This includes a ResNet50 (He et al., 2016) with frozen Batch-Normalization (Ioffe & Szegedy, 2015), normalization of the output embeddings with dimensionality 128 and optimization with Adam (Kingma & Ba, 2015) using a learning rate of 10^{-5} and weight decay of $3 \cdot 10{-}4$. The input images are randomly resized and cropped from the original image size to 224×224 for training. Further augmentation by random horizontal flipping with $p = 0.5$ is applied. During testing, center crops of size 224×224 are used. The batchsize is set to 112.

Training runs on CUB200-2011 and CARS196 are done over 150 epochs and 100 epochs for SOP for all experiments without any learning rate scheduling, except for the state-of-the-art experiments (see again 2). For the latter, we made use of slightly longer training to account for conservative learning rate scheduling, which is similarly done across reference methods noted in tab. 2. Schedule and decay values are determined over validation subset performances. All baseline DML objectives we apply our self-distillation module *S2SD* on use the default parameters noted in (Roth et al., 2020b) with the single exception of Margin Loss on SOP, where we found class margins $\beta = 0.9$ to be more beneficial for distillation than the default $\beta = 1.2$. This was done as changing from $\beta = 1.2$ to $\beta = 0.9$ had no notable impact on the baseline performance. Finally, similar to (Kim et al., 2020), we found a warmup epoch of all MLPs to improve convergence on SOP. Spectral decay computations in §5.3 follow the setting described in Supp. D.

We implement everything in PyTorch (Paszke et al., 2017). Experiments are done on GPU servers containing Nvidia Titan X, P100 and T4s, however memory usage never exceeds 12GB. Each result is averaged over five seeds, and for the sake of reproducibilty and result validity, we report mean and standard deviation, even though this is commonly neglected in DML literature.

[1]Repository: github.com/Confusezius/Revisiting_Deep_Metric_Learning_PyTorch

B. Baseline Methods

This section provides a more detailed explanation of the DML baseline objectives we used alongside our self-distillation module *S2SD* in the experimental section 4. For additional details, we refer to the supplementary material in (Roth et al., 2020b). For the mathematical notation, we refer to Section 3.1. We use $\psi = f \circ \phi$ to denote the feature network ϕ with embedding f, and ψ_i the embedding of a sample x_i. Finally, alongside the method descriptions we provide the used hyperparameters.

Margin Loss (Wu et al., 2017) builds on triplet/pair-based losses, but introduces both class-specific, learnable boundaries $\{\beta_{y_k}\}_{k=1...C}$ (with number of classes C) between positive and negative pairs, as well as distance-based sampling for negatives:

$$\mathcal{L}_{\text{margin}} = \sum_{x_i, x_j \in \mathcal{P}_\mathcal{B}} [m + (-1)^{\mathbb{I}_{y_i = y_j}} (\beta_{y_i} - d(\psi_i, \psi_j))]_+ \tag{9}$$

$$p(x_j | x_i, y_i \neq y_j) = \min \left(\lambda, \left[d(\psi_i, \psi_j)^{n-2} (1 - \frac{1}{4} d(\psi_i, \psi_j)^2)^{\frac{n-3}{2}} \right]^{-1} \right) \tag{10}$$

where $\mathcal{P}_\mathcal{B}$ denotes the available pairs in minibatch $\mathcal{B}$, and n the embedding dimension. Throughout this work, we use $\beta = 1.2$ except for *S2SD* on SOP, where we found $\beta = 0.9$ to work better without changing the baseline performance. We set the learning rate for the class boundaries as $5 \cdot 10^{-4}$, and margin $m = 0.2$.

Regularized Margin Loss (Roth et al., 2020b) proposes a simple regularization scheme on the margin loss that increases the number of directions of significant variance in the embedding space by randomly exchanging a negative sample with a positive one with probability p_{switch}. For ResNet-backbones, we use $p_{\text{switch}} = 0.4$ for CUB200, $p_{\text{switch}} = 0.35$ for CARS196 and $p_{\text{switch}} = 0.15$ for SOP as done in (Roth et al., 2020b). For Inception-based backbones, we set $p_{\text{switch}} = 0.15$ for CUB200 and CARS196 and $p_{\text{switch}} = 0.3$ for SOP.

Multisimilarity Loss (Wang et al., 2019) incorporates more similarities into training by operating directly on all positive and negative samples for an anchor x_i, while also incorporating a sampling operation that encourages the usage of harder training samples:

$$d_c^*(i,j) = \begin{cases} d_c(\psi_i, \psi_j) & d_c(\psi_i, \psi_j) > \min_{j \in \mathcal{P}_i} d_c(\psi_i, \psi_j) - \epsilon \\ d_c(\psi_i, \psi_j) & d_c(\psi_i, \psi_j) < \max_{k \in \mathcal{N}_i} d_c(\psi_i, \psi_k) + \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$\begin{aligned} \mathcal{L}_{\text{multisim}} = \frac{1}{b} \sum_{i \in \mathcal{B}} \left[\frac{1}{\alpha} \log[1 + \sum_{j \in \mathcal{P}_i} \exp(-\alpha(d_c^*(\psi_i, \psi_j) - \lambda))] \right] \\ + \sum_{i \in \mathcal{B}} \left[\frac{1}{\beta} \log[1 + \sum_{k \in \mathcal{N}_i} \exp(\beta(d_c^*(\psi_i, \psi_k) - \lambda))] \right] \end{aligned} \tag{12}$$

where d_c denotes the cosine similarity instead of the euclidean distance, and $\mathcal{P}_i / \mathcal{N}_i$ the set of positives and negatives for x_i in the minibatch, respectively. We use the default values $\alpha = 2$, $\beta = 40$, $\lambda = 0.5$ and $\epsilon = 0.1$.

C. Evaluation Metrics

The evaluation metrics used throughout this work are recall @ 1 (R@1), recall @ 2 (R@2) and Normalized Mutual Information (NMI), capturing two distinct embedding space properties.

Recall@K, see e.g. in (Jegou et al., 2011), especially Recall@1 and Recall@2, is the primary metric used to compare the performance of DML methods and approaches, as it offers strong insights into retrieval performances of the learned embedding spaces. Given the set of embedded samples $\psi_i \in \Psi$ with $\psi_i = \psi(x_i)$ and $x_i \in \mathcal{X}$, and the sorted set of k nearest neighbours for any sample ϕ_a,

$$\mathcal{F}_a^k = \min_{d(\phi_a, \cdot)} \text{sort} \ \arg\min_{\mathcal{F} \subset \mathcal{X}, |\mathcal{F}| = k} \sum_{x_f \in \mathcal{F}} d(\phi_a, \phi_f) \tag{13}$$

Recall@K is measured as

$$\text{Recall@K} = \frac{1}{|\mathcal{X}|} \sum_{x_i \in \mathcal{X}} \begin{cases} 1 & \exists x_k \in \mathcal{F}_i^k \text{s.t.} y_k = y_i \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

which evaluates how likely semantically corresponding pairs (as determined here by the labelling $y_i \in \mathcal{Y}$) will occur in a neighbourhood of size k.

Normalized Mutual Information (NMI), see (Manning et al., 2010), evaluates the clustering quality of the embedded samples Ψ (taken from $\mathcal{X}$). It is computed by first clustering with K cluster centers, usually corresponding to the number of classes available, using a cluster method of choice s.a. K-Means (Lloyd, 1982). This assigns each sample x_i a cluster label/id ω_i based on the nearest cluster centroid. With $\eta_k = \{i|\omega_i = \omega_k\}$ the set of samples with cluster label, $\Omega = \{\eta_k\}_k^K$ the set of cluster sets, $\nu_k = \{i|y_i = y_k\}$ the set of samples with true label y_k and $\Upsilon = \{\nu_k\}_k^K$ the set of class label sets, the Normalized Mutual Information is given as

$$\text{NMI}(\Omega, \Upsilon) = \frac{I(\Omega, \Upsilon)}{2 \cdot (H(\Omega) + H(\Upsilon))} \tag{15}$$

with mutual information $I(\cdot, \cdot)$ and entropy $H(\cdot)$.

D. Generalization Metrics

Embedding Space Density. Given sets of embeddings Ψ, we first define the average inter-class distance as

$$\pi_{\text{inter}}(\Psi) = \frac{1}{Z_{\text{inter}}} \sum_{y_l, y_k, l \neq k} d(\mu(\Psi_{y_l}), \mu(\Psi_{y_k})) \tag{16}$$

which measures the average distances between groups of embeddings with respective classes y_l and y_k, estimated by the respective class centers $\mu(\cdot)$. Z_{inter} denotes a normalization constant based on the number of available classes. We also introduce the average intra-class distance as the mean distance between samples within their respective class

$$\pi_{\text{intra}}(\Psi) = \frac{1}{Z_{\text{intra}}} \sum_{y_l \in \mathcal{Y}} \sum_{\psi_i, \psi_j \in \Psi_{y_l}, i \neq j} d(\psi_i, \psi_j) \tag{17}$$

again with normalization constant Z_{intra} and set of embeddings with class y_l, Ψ_{y_l}. Given these two quantities, the embedding space density is then defined as

$$\pi_{\text{ratio}}(\Psi) = \frac{\pi_{\text{intra}}(\Psi)}{\pi_{\text{inter}}(\Psi)} \tag{18}$$

and effectively measured how densely samples and classes are grouped together. (Roth et al., 2020b) show that optimizing the DML problem while keeping the embedding space density high, i.e. without aggressive clustering, encourages better generalization to unseen test classes.

Spectral Decay. The spectral decay metric $\rho(\Psi)$ defines the KL-divergence between the (sorted) spectrum of D singular values $\mathcal{S}_\Psi^{\text{singular}}$ (obtained via Singular Value Decomposition (SVD)) and a D-dimensional uniform distribution $\mathcal{U}_D$, and is inversly related to the entropy of the embedding space:

$$\rho(\Psi) = \mathcal{D}_{\text{KL}} \left(\mathcal{U}_D, \mathcal{S}_\Psi^{\text{singular}} \right) \tag{19}$$

It does not account for class distributions. (Roth et al., 2020b) show that doing DML while encouraging a high-entropy feature space notably benefits the generalization performance. In our experiments, we disregard the first 10 singular vectors (out of 128) to highlight the feature diversity. This is important, as we evaluate the spectral decay within the same objectives, which results in the first few singular values to be highly similar.

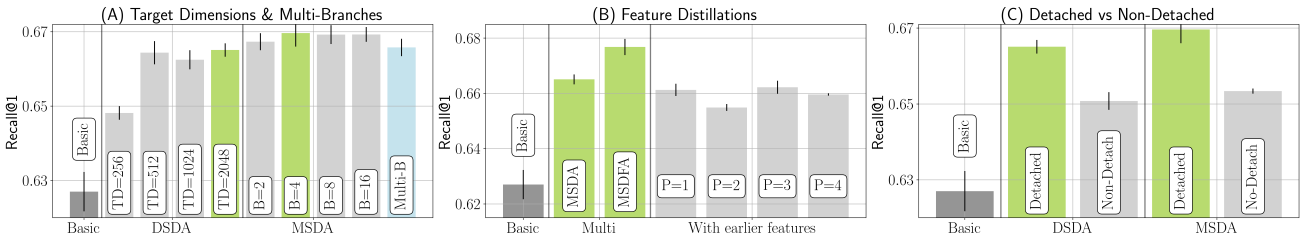

Figure 4. Additional ablations. **(A)** Increasing target dimensions offers notable improvements. We opt for a target dimension of 2048 due to slightly higher mean improvements. For multiple embedding branches (#B), there seems to be an optimum at four branches. **(B)** Furthermore, feature distillation gives another notable boost. However, this only holds for the globally averaged penultimate feature representation. When distilling more fine-grained feature representations, performance degenerates (where #P denotes smaller pooling windows applied to the penultimate feature representation). **(C)** We show that detached auxiliary branches for distillation are crucial to higher improvements, as we want the reference embedding space to approximate the higher-dimensional one.

E. Additional Experiments

This part extends the set of ablations experiments performed in section 5.4 in the main paper.

a. Detaching target spaces for distillation. We examine whether it is preferable to detach the target embeddings from the distillation loss (see eq. 3), as we want the reference embedding space to approximate the higher-dimensional relations. Similarly, we do not want the target embedding networks g_i to reduce high-dimensional to lower-dimensional relations to optimizer for the distillation constraint. As can be seen in fig 4C, it is indeed the case that detaching the target embedding spaces is notably beneficial for a stronger reference embedding, supporting the previous motivation.

b. Influence of varying target dimensions. As noted at the beginning of section 4, we set the target dimension for dual self-distillation (**DSD**) to $d = 2048$, which we motivate through a small ablation study in fig. 4A, with *TD* denoting the target dimension of choice. As can be seen, benefits plateau when the target dimension reaches more than four times the reference dimension. However, to be directly comparable to high-dimensional reference settings, we set $d = 2048$ as default.

c. Ablating multiple distillation scales. Going further, we extend the module with additional embedding branches to the multiscale self-distillation approach (**MSD**), all operating in different, but higher-than-reference dimension. As already shown in Figure 3B in the main paper, there is a benefit of multiscale distillations by encouraging reusable sample relations. In this part, we motivate the choice of four target branches (as noted in sec. 4). Looking at figure 4A, where B denotes the number of additional target spaces, we can see a benefit in multiple additional target spaces of ascending dimension. As the improvements saturate after $B = 4$, we simply set this as the default value. However, the additional benefits of going to multiscale from dual distillation are not as high as going from no to dual target space distillation, showcasing the general benefit of high-dimensional concurrent self-distillation. Finally, we highlight that a multiscale approach slightly outperforms a multibranch distillation setup (Fig. 4A, *Multi-B*) where each target branch has the same target dimension of 2048 while introducing less additional parameters.

d. Finer-grained feature distillation. As already shown in section 4 and again in figure 4B, we see benefits of feature distillation, using the (globally averaged) normalized penultimate feature space. It therefore makes sense to investigate the benefits of distilling even more fine-grained feature representation. Defining $\mathcal{P} = [(3,3),(1,1),(2,2),(4,1)]$ as the pooling window size applied to the non-average penultimate feature representation, we investigate less compressed feature representation space. As can be seen in fig. 4B, where P denotes the index to $\mathcal{P}$, there appears to be no benefits in distilling feature representations higher up the network.

e. Runtime comparison of base dimensionalities. We highlight relative retrieval times at different base dimensionalities in Tab. 3 using faiss (Johnson et al., 2017) on a NVIDIA 1080Ti and a synthetic set of $N = 250000$ embeddings of dimensionality $d \in [32, 64, 128, 256, 512, 1024, 2048]$. With *S2SD* matching $d = 64/128$ to base dimensionalities $d = 512/2048$ (see §5.3), runtime can be reduced by up to a magnitude.

Dimensionality d	32	64	128	256	512	1024	2048
Runtime (s)	1.54±0.00	1.98±0.00	2.71±0.00	4.35±0.00	7.38±0.01	13.83±0.02	27.21±0.17

Table 3. Sample retrieval times for 250000 embeddings with varying base dimensionalities.

F. Pseudo-Code

```
import torch, torch.nn as nn, torch.nn.functional as F
from F import normalize as norm

"""
Parameters:
    self.base_criterion: base DML objective
    self.trgt_criteria: list of DML objectives for target spaces
    self.trgt_nets: Module list of auxiliary embedding MLPs
    self.dist_gamma: distillation weight
    self.it_before_feat_distill: iterations before feature distill
"""

def forward(self, batch, labels, pre_batch, **kwargs):
    """
    Args:
        batch: image embeddings, shape: bs x d
        labels: image labels, shape: bs
        pre_batch: penultimate network features, shape: bs x d*
    """
    bs, batch = len(batch), norm(batch, dim=-1)

    ### Compute ref. sample relations and loss on ref. embedding space
    base_smat = batch.mm(batch.T)
    base_loss = self.base_criterion(batch, labels, **kwargs)

    ### Do global average pooling (and max. pool if wanted)
    avg_pre_batch  = nn.AdaptiveAvgPool2d(1)(pre_batch).view(bs,-1)
    avg_pre_batch += nn.AdaptiveMaxPool2d(1)(pre_batch).view(bs,-1)

    ### Computing MSDA loss (Targets & Distillations)
    dist_losses, trgt_losses  = [], []
    for trgt_crit,trgt_net in zip(self.trgt_criteria,self.trgt_nets):
        trgt_batch     = norm(trgt_net(avg_pre_batch),dim=-1)
        trgt_loss      = trgt_crit(trgt_batch, labels, **kwargs)
        trgt_smat      = trgt_batch.mm(trgt_batch.T)
        base_trgt_dist = self.kl_div(base_smat, trgt_smat.detach())
        trgt_losses.append(trgt_loss)
        dist_losses.append(base_trgt_dist)

    ### MSDA loss
    multi_dist_loss  = (base_loss+torch.stack(trgt_losses).mean())/2.
    multi_dist_loss += self.dist_gamma*torch.stack(dist_losses).mean()

    ### Distillation of penultimate features -> MSDFA
    src_feat_dist = 0
    if self.it_count>=self.it_before_feat_distill:
        n_avg_pre_batch = norm(avg_pre_batch, dim=-1).detach()
        avg_feat_smat   = n_avg_pre_batch.mm(n_avg_pre_batch.T)
        src_feat_dist   = self.kl_div(base_smat, avg_feat_smat.detach())

    ### Total S2SD training objective
    total_loss = multi_distill_loss + self.dist_gamma*src_feat_dist
    self.it_count+=1
    return total_loss

def kl_div(self, A, B, T=1):
    log_p_A = F.log_softmax(A/self.T, dim=-1)
    p_B     = F.softmax(B/self.T, dim=-1)
    kl_d    = F.kl_div(log_p_A, p_B,reduction='sum')*T**2/A.size(0)
    return kl_d
```

Listing 1. PyTorch Implementation for *S2SD*.

G. Detailed Evaluation Results

This table contains all method ablations for a fair evaluation as used in Section 5.2 and Table 1.

Table 4. Detailed Comparison of Recall@1 and NMI performances against well performing DML objectives examined in section 5.2. This is the complete version to table 1. All results are computed over 5-run averages. (*) For Margin Loss and SOP, we found $\beta = 0.9$ to give better distillation results without notably influencing baseline performance.

Benchmarks→	CUB200-2011		CARS196		SOP	
Approaches ↓	R@1	NMI	R@1	NMI	R@1	NMI
Margin(*)	63.09 ± 0.46	68.21 ± 0.33	79.86 ± 0.33	67.36 ± 0.34	78.43 ± 0.07	90.40 ± 0.03
+ DSD	65.11 ± 0.18	69.65 ± 0.44	83.19 ± 0.18	69.28 ± 0.56	79.05 ± 0.12	90.52 ± 0.18
+ DSDA	65.77 ± 0.55	69.85 ± 0.25	83.92 ± 0.08	69.95 ± 0.21	77.78 ± 0.15	90.29 ± 0.08
+ MSD	66.13 ± 0.34	70.83 ± 0.27	83.63 ± 0.31	69.80 ± 0.36	79.26 ± 0.15	90.60 ± 0.10
+ MSDA	66.14 ± 0.32	70.82 ± 0.18	84.31 ± 0.12	70.17 ± 0.30	78.04 ± 0.11	90.45 ± 0.05
+ MSDF	67.58 ± 0.32	$71,47 \pm 0.19$	85.55 ± 0.23	71.68 ± 0.54	79.63 ± 0.14	90.70 ± 0.09
+ MSDFA	67.21 ± 0.23	71.43 ± 0.25	86.45 ± 0.35	71.46 ± 0.24	78.82 ± 0.09	90.49 ± 0.06
R-Margin	64.93 ± 0.42	68.36 ± 0.32	82.37 ± 0.13	68.66 ± 0.47	77.58 ± 0.11	90.42 ± 0.03
+ DSD	66.58 ± 0.08	70.03 ± 0.41	84.64 ± 0.16	70.87 ± 0.18	77.86 ± 0.10	90.50 ± 0.03
+ DSDA	67.11 ± 0.43	70.39 ± 0.48	84.32 ± 0.36	70.85 ± 0.16	77.79 ± 0.11	90.37 ± 0.04
+ MSD	66.81 ± 0.27	70.47 ± 0.16	85.01 ± 0.10	71.67 ± 0.40	78.00 ± 0.06	90.47 ± 0.04
+ MSDA	67.31 ± 0.41	71.01 ± 0.24	85.34 ± 0.17	71.85 ± 0.20	77.93 ± 0.06	90.29 ± 0.08
+ MSDF	68.12 ± 0.30	71.80 ± 0.33	85.78 ± 0.22	72.24 ± 0.31	78.57 ± 0.09	90.58 ± 0.02
+ MSDFA	68.58 ± 0.26	71.64 ± 0.40	86.81 ± 0.35	71.48 ± 0.29	78.00 ± 0.11	90.41 ± 0.02
Multisimilarity	62.80 ± 0.70	68.55 ± 0.38	81.68 ± 0.19	69.43 ± 0.38	77.99 ± 0.09	90.00 ± 0.02
+ DSD	65.57 ± 0.26	70.08 ± 0.33	83.51 ± 0.20	70.30 ± 0.05	78.23 ± 0.04	90.08 ± 0.04
+ DSDA	66.60 ± 0.43	70.74 ± 0.40	84.42 ± 0.28	70.36 ± 0.34	77.92 ± 0.12	89.99 ± 0.04
+ MSD	65.80 ± 0.16	70.53 ± 0.01	83.98 ± 0.10	71.34 ± 0.09	78.42 ± 0.09	90.09 ± 0.03
+ MSDA	66.96 ± 0.36	70.77 ± 0.05	85.04 ± 0.14	71.09 ± 0.23	77.98 ± 0.05	90.02 ± 0.04
+ MSDF	67.04 ± 0.29	71.87 ± 0.19	85.69 ± 0.19	72.77 ± 0.13	78.59 ± 0.08	90.09 ± 0.06
+ MSDFA	67.68 ± 0.29	71.40 ± 0.21	85.89 ± 0.15	71.45 ± 0.26	78.07 ± 0.06	89.88 ± 0.10

H. Evaluation Results using mAP@R

This table measures performance of methods investigated in Table 1 using the mAP@R(@1000) metric used in (Roth et al., 2020b). The results here coincide with those measured using Recall@1. This comes at no surprise, as both metrics are strongly correlated when measuring the performance of Deep Metric Learning methods (Roth et al., 2020b).

I. Detailed Ablation Results

Detailed values to the ablation experiments done in section 5.4 and E.

Table 5. Detailed Comparison of mAP@R (as used in (Roth et al., 2020b) and (Musgrave et al., 2020) and based on the formulation used in (Roth et al., 2020b)) against well performing DML objectives examined in section 5.2.. All results are computed over 5-run averages. (*) For Margin Loss and SOP, we found $\beta = 0.9$ to give better distillation results without notably influencing baseline performance. **Bold** denotes best results per objective and dataset. **Bluebold** denotes best performance per dataset.

Benchmarks→	CUB200-2011	CARS196	SOP
Approaches ↓	mAP	mAP	mAP
Margin(*)	32.63 ± 0.40	32.50 ± 0.28	46.90 ± 0.16
+ DSD	33.85 ± 0.38	34.01 ± 0.39	47.39 ± 0.18
+ MSD	34.79 ± 0.35	34.64 ± 0.31	48.17 ± 0.07
+ MSDF	35.68 ± 0.29	35.26 ± 0.41	$\mathbf{48.24} \pm 0.10$
+ MSDFA	$\mathbf{35.98} \pm 0.23$	$\mathbf{35.98} \pm 0.40$	47.04 ± 0.26
R-Margin	33.38 ± 0.27	34.57 ± 0.30	46.02 ± 0.14
+ DSD	34.46 ± 0.30	35.12 ± 0.22	46.20 ± 0.19
+ MSD	35.11 ± 0.41	35.78 ± 0.40	46.59 ± 0.16
+ MSDF	35.99 ± 0.36	37.32 ± 0.40	$\mathbf{47.08} \pm 0.17$
+ MSDFA	$\mathbf{36.25} \pm 0.37$	$\mathbf{37.67} \pm 0.35$	46.71 ± 0.16
Multisimilarity	30.92 ± 0.49	31.92 ± 0.44	46.23 ± 0.08
+ DSD	33.20 ± 0.34	33.67 ± 0.27	46.21 ± 0.15
+ MSD	34.00 ± 0.35	34.67 ± 0.26	46.45 ± 0.11
+ MSDF	35.16 ± 0.32	$\mathbf{35.52} \pm 0.51$	$\mathbf{46.52} \pm 0.17$
+ MSDFA	$\mathbf{35.35} \pm 0.24$	35.13 ± 0.35	45.39 ± 0.28

Table 6. Additional ProxyAnchor (Kim et al., 2020) results with and without *S2SD* variants using the proposed, but different, default architecture in (Kim et al., 2020) to highlight that *S2SD* works equally well on already strong proxy-based objectives objectives with different architectural settings as well.

Benchmarks→	CUB200-2011		CARS196		SOP	
Setting	R@1	NMI	R@1	NMI	R@1	NMI
ProxyAnchor	64.58 ± 0.23	68.95 ± 0.24	82.55 ± 0.41	69.49 ± 0.30	78.33 ± 0.08	90.24 ± 0.06
+ DSD	65.50 ± 0.47	69.97 ± 0.55	83.52 ± 0.11	70.76 ± 0.17	78.33 ± 0.08	90.24 ± 0.06
+ MSD	65.92 ± 0.28	69.88 ± 0.21	83.99 ± 0.33	70.95 ± 0.19	78.47 ± 0.03	90.29 ± 0.06
+ MSDF	66.71 ± 0.12	70.60 ± 0.24	85.20 ± 0.09	71.19 ± 0.18	78.50 ± 0.04	90.31 ± 0.03

Table 7. Experiment: Comparison of concurrent self-distillation against standard 2-stage distillation. This table also shows that training without distillation (*Joint*) or training in high dimension while learning a detached low-dimensional embedding layer (*Concur.*) does not benefit performance notably. See fig. 3A. All results are computed over 5-run averages.

Experiment	Setting	R@1
Distillation	Best Teacher (d=1024)	66.04 ± 0.17
	Base Student (d=128)	62.70 ± 0.53
	Distill Student (d=128)	63.89 ± 0.14
	Concur. Student (d=128)	63.08 ± 0.42
	Joint Student (d=128)	62.93 ± 0.22
	DSD (d=128)	65.57 ± 0.26
	DSDA (d=128)	66.51 ± 0.18
	MSDA (d=128)	66.96 ± 0.36
	MSDFA (d=128)	67.68 ± 0.29

Table 8. Experiment: Benefit of self-distillation across embedding dimensionalities. These results go along with 3B. All results are computed over 5-run averages.

Experiment	Setting	R@1	Setting	R@1
Embedding Dimensionality	Base (d=16)	48.18 ± 0.54	MSD (d=256)	66.90 ± 0.11
	Basic (d=32)	54.54 ± 0.42	MSD (d=512)	67.07 ± 0.02
	Basic (d=64)	59.31 ± 0.26	MSD (d=1024)	66.69 ± 0.11
	Basic (d=128)	62.70 ± 0.53	MSD (d=2048)	66.68 ± 0.18
	Basic (d=256)	64.39 ± 0.30	MSDA (d=16)	51.57 ± 0.39
	Basic (d=512)	65.95 ± 0.19	MSDA (d=32)	61.55 ± 0.23
	Basic (d=1024)	66.04 ± 0.17	MSDA (d=64)	64.94 ± 0.50
	Basic (d=2048)	66.03 ± 0.20	MSDA (d=128)	66.96 ± 0.36
	DSD (d=16)	49.92 ± 0.09	MSDA (d=256)	67.63 ± 0.34
	DSD (d=32)	59.75 ± 0.78	MSDA (d=512)	67.76 ± 0.26
	DSD (d=64)	62.75 ± 0.15	MSDA (d=1024)	67.79 ± 0.10
	DSD (d=128)	65.57 ± 0.26	MSDA (d=2048)	67.33 ± 0.09
	DSD (d=256)	66.34 ± 0.07	MSDF (d=16)	51.99 ± 0.57
	DSD (d=512)	66.89 ± 0.15	MSDF (d=32)	61.61 ± 0.41
	DSD (d=1024)	66.57 ± 0.11	MSDF (d=64)	65.31 ± 0.23
	DSD (d=2048)	66.54 ± 0.28	MSDF (d=128)	66.66 ± 0.29
	DSDA (d=16)	50.77 ± 0.71	MSDF (d=256)	67.47 ± 0.11
	DSDA (d=32)	60.13 ± 0.45	MSDF (d=512)	67.59 ± 0.03
	DSDA (d=64)	63.84 ± 0.36	MSDF (d=1024)	67.40 ± 0.15
	DSDA (d=128)	66.51 ± 0.18	MSDF (d=2048)	67.01 ± 0.35
	DSDA (d=256)	67.13 ± 0.24	MSDFA (d=16)	52.96 ± 0.44
	DSDA (d=512)	67.54 ± 0.24	MSDFA (d=32)	61.98 ± 0.36
	DSDA (d=1024)	67.27 ± 0.22	MSDFA (d=64)	65.19 ± 0.40
	DSDA (d=2048)	67.33 ± 0.30	MSDFA (d=128)	67.68 ± 0.29
	MSD (d=16)	50.51 ± 0.21	MSDFA (d=256)	68.48 ± 0.28
	MSD (d=32)	60.00 ± 0.28	MSDFA (d=512)	69.06 ± 0.14
	MSD (d=64)	63.74 ± 0.24	MSDFA (d=1024)	69.08 ± 0.22
	MSD (d=128)	65.80 ± 0.16	MSDFA (d=2048)	69.29 ± 0.35

Table 9. Experiment: Methods of distillation between reference and target embedding spaces. See fig. 3C. Used Method: **DSDA**. All results are computed over 5-run averages.

Experiment	Setting	R@1
Distillation Methods	R-KL	66.51 ± 0.18
	Cos	65.73 ± 0.27
	Eucl	65.61 ± 0.28
	KL-Full	65.91 ± 0.32
	KL-Mean	64.00 ± 0.55
	Basic	62.70 ± 0.53

Table 10. Experiment: Structure of the secondary branch. More specifically, this table contains specific values used in fig. 3D. Used Method: **DSDA**. All results are computed over 5-run averages.

Experiment	Setting	R@1
Secondary Branch Structure	2 Layers	66.51 ± 0.18
	3 Layers	66.03 ± 0.29
	4 Layers	65.76 ± 0.65
	Linear	65.28 ± 0.48
	Basic	62.70 ± 0.53

Table 11. Experiment: Different distillation hierarchies. See fig. 3E. Used Method: **MSDA**. All results are computed over 5-run averages.

Experiment	Setting	R@1
Distillation Hierarchies	Basic	62.70 ± 0.53
	Straight	66.96 ± 0.36
	Fully	65.58 ± 0.46
	Stacked	65.21 ± 0.25

Table 12. Experiment: Influence of distillation weight γ. See fig. 3F. Used Method: **DSD**. All results are computed over 5-run averages.

Experiment	Setting	R@1 CUB200	R@1 CARS196	R@1 SOP
Weight Ablation	0.0	62.70 ± 0.53	81.32 ± 0.36	77.78 ± 0.06
	0.2	62.85 ± 0.41	81.65 ± 0.40	78.22 ± 0.07
	1.0	62.92 ± 0.16	81.92 ± 0.51	78.71 ± 0.10
	5.0	63.88 ± 0.19	82.96 ± 0.10	79.05 ± 0.12
	20.0	65.43 ± 0.21	83.50 ± 0.35	78.72 ± 0.10
	50.0	65.57 ± 0.26	83.51 ± 0.33	78.10 ± 0.13
	250.0	65.04 ± 0.33	82.25 ± 0.62	77.41 ± 0.12
	1000.0	63.32 ± 0.36	77.00 ± 0.66	77.01 ± 0.08
	2000.0	62.76 ± 0.31	72.72 ± 0.85	76.37 ± 0.11
	5000.0	62.05 ± 0.62	70.90 ± 0.97	75.42 ± 0.20

Table 13. Experiment: Evaluation target dimensions and levels of multiscale distillation. See fig. 4A. All results are computed over 5-run averages.

Experiment	Setting	R@1
Target Dimensionalities	Basic	62.70 ± 0.53
	DSDA, TD=256	64.82 ± 0.18
	DSDA, TD=512	66.44 ± 0.31
	DSDA, TD=1024	66.25 ± 0.26
	DSDA, TD=2048	66.51 ± 0.18
MultiScale distillation	MSDA, #B=2	66.76 ± 0.23
	MSDA, #B=4	66.96 ± 0.36
	MSDA, #B=8	66.72 ± 0.25
	MSDA, #B=16	66.59 ± 0.20

Table 14. Experiment: Is it beneficial to distill more fine-grained features? See fig. 4B. All results are computed over 5-run averages.

Experiment	Setting	R@1
Earlier Features	Basic	62.70 ± 0.53
	MSDA	66.96 ± 0.36
	MSDFA	67.68 ± 0.29
	MSDFA, #P=1	66.13 ± 0.22
	MSDFA, #P=2	65.49 ± 0.12
	MSDFA, #P=3	66.22 ± 0.24
	MSDFA, #P=4	65.96 ± 0.04

Table 15. Experiment: Is it necessary to detach auxiliary branches for distillation? See fig. 4C. All results are computed over 5-run averages.

Experiment	Setting	R@1
Branch Detaching	Basic	62.70 ± 0.53
	DSDA, Detached	66.51 ± 0.18
	DSDA, Non-Detach	65.08 ± 0.23
	MSDA, Detached	66.96 ± 0.36
	MSDA, No-Detach	65.34 ± 0.07