ProcSim: Proxy-based Confidence for Robust Similarity Learning

Oriol Barbany[1†] Xiaofan Lin[2] Muhammet Bastan[2] Arnab Dhua[2]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC [2]Visual Search & AR, Amazon
obarbany@iri.upc.edu, {xiaofanl,mbastan,adhua}@amazon.com

A. Additional implementation details

This section provides further details on the implementation of ProcSim. We use the PyTorch framework [34] for all the components below.

A.1. Data augmentation

We perform standard data augmentation techniques as in previous Deep Metric Learning (DML) works [39, 40, 60]: random cropping to 224×224 and horizontal flipping with probability 0.5.

A.2. Model

The DML model is a ResNet-50 [16] in which we replaced the output classification layer with a fully connected layer that provides embeddings. The batch normalization layers have been frozen for improved convergence and stability across batch sizes [39]. We take the ResNet-50 model implementation from the PyTorch library for computer vision `torchvision`, which also provides weights for ImageNet [41]. In particular, we use the second version of the pre-trained weights, *i.e.*, `IMAGENET1K_V2`. Throughout all the experiments, we use an embedding dimension of 512.

A.3. Optimization

We use the Adam [62] optimizer to update the parameters of the DML model. For CUB200 [49], we train the model for 150 epochs with a base learning rate of 10^{-4}. For both Cars196 [22] and Standard Online Products (SOP) [46], we use a base learning rate value of 10^{-5} and train for 250 epochs. In all cases, we use a weight decay [63] of $4 \cdot 10^{-4}$ and the default values in PyTorch [34] for the rest of the hyperparameters. We do not apply learning rate scheduling for unbiased comparison [39].

Proxies in Proxy-NCA are optimized independently using the Adam optimizer with all the default parameters. This choice is related to the observations Proxy-NCA++ [70], which indicate that using independent optimizers for updating the class proxies and the model parameters is one of the main drivers of performance that improves upon vanilla Proxy-NCA [30].

The training process uses 4 NVIDIA Tesla V100 SXM2 16 GB GPUs with a batch size of 90 each. Note that the effective batch size is 360, which allows full utilization of the hardware at disposal for faster training and is typically not considered an influential factor of variation [39]. While datasets with many classes like SOP [46] may benefit from a larger batch size, Wang *et al.* [50] showed that when training a model with the Multi-similarity (MS) loss, the performance on dataset like CUB200 [49] decreases with large batch sizes over 80.

A.4. Loss

The ProcSim loss is composed of two terms, as seen in Eq. (4). One such term is the supervised DML loss. By default, we use the MS loss [50], *cf*. Eq. (5), with the hyperparameters proposed in the original paper: $\alpha = 2$, $\beta = 40$, and $\delta = 0.1$. We adapt the original implementation[1] to perform batch operations and exclude pairs $(\mathbf{x}_i, \mathbf{x}_i)$ in $\mathcal{P}$ instead of removing all pairs with a similarity higher than $1 - \epsilon$, where we set $\epsilon = 10^{-5}$.

The Pseudolabel Language Guidance (PLG) loss is computed using the original implementation[2], in which the language part of CLIP [37] (ViT-B/32 variant) is the chosen Large Language Model (LLM). In the experiment with the BERT language model in Tab. 2, we use the model and weights from hugging face[3]. The parameter ω scaling the PLG loss is set to $\omega = 10$ for CUB200 [49] and $\omega = 5.5$ for Cars196 [22], the values reported in the official code repository. For SOP [46], they recommend using $\omega \in [0.1, 1]$, and we chose $\omega = 0.5$ after testing with $\omega \in \{0.1, 0.5, 1.0\}$.

To compute the sample confidence, we treat τ and σ as constant during backpropagation. We calculate the Proxy-NCA loss [30] using the PyTorch metric learning library

† Work performed during an internship at Amazon.

[1]https://github.com/msight-tech/research-ms-loss
[2]https://github.com/ExplainableML/LanguageGuidance_for_DML
[3]https://huggingface.co/bert-base-uncased

[31] with the default hyperparameters. The value of λ in Eq. (3) determines how much the confidence of a sample decreases for losses greater than Otsu's threshold [33]. Asymptotically, $\sigma_i \rightarrow 1$ as $\lambda \rightarrow \infty$, and as $\lambda \rightarrow 0$, $\sigma_i \rightarrow 0$ if $\ell_i^{\text{Proxy}} > \tau^{\text{Otsu}}$, and $\sigma_i \rightarrow 1$ if $\ell_i^{\text{Proxy}} \leq \tau^{\text{Otsu}}$. We tested $\lambda \in \{0.01, 0.1, 1.0, 10.0\}$ and found the values of $\lambda = 0.1$ on Cars196 [22] and SOP [46], and $\lambda = 1.0$ on CUB200 [49], to give good performance across different levels of noise.

Note that a larger λ on CUB200 [49] implies that samples with a high loss are more penalized. This penalization explains the behavior observed in Tab. 1, in which ProcSim obtained the best performance on noisy data. That is because the contribution of clean samples was potentially decreased in the absence of synthetic noise.

B. Computation of confidence values

Proof of Claim 1. The confidence score in Eq. (3) is claimed to satisfy Conditions (i) to (v). In the following, we prove each of these conditions:

(i) σ_i is translation invariant w.r.t. ℓ_i^{Proxy}:

Note that each value ℓ_i^{Proxy} is subtracted by Otsu's threshold τ. Thus, proving that τ is equivariant to translations of the proxy loss suffices (as those translations get canceled out). τ is computed as the cost minimizer threshold among those in $\mathcal{T}$ (see Alg. 1). $\mathcal{T}$ are the midpoints between consecutive loss values. Hence, τ is translation equivariant. Finally, the cost is unaltered as the variance is translation invariant.

(ii) $\sigma_i \geq \sigma_j \iff \ell_i^{\text{Proxy}} \leq \ell_j^{\text{Proxy}}$ (i, j in the same batch):

Since (i, j) are in the same batch, they will share the same threshold τ. Then, we have

$$\frac{\ell_i^{\text{Proxy}} - \tau}{2\lambda} \leq \frac{\ell_j^{\text{Proxy}} - \tau}{2\lambda} \qquad \text{Since } \lambda \in \mathbb{R}_+ . \tag{5}$$

The function $\max\{0, \cdot\}$ is increasing and hence the order is preserved. Its image is $\mathbb{R}_+$, and the restriction of $W(\cdot)$ to the domain of positive numbers is monotonously increasing. Therefore, for $a \leq b$

$$W(a) \leq W(b) \iff e^{-W(a)} \geq e^{-W(b)} , \tag{6}$$

since the exponential function is monotonously increasing.

(iii) $\sigma_i \in [0, 1]$:

The image of the restriction of the Lambert W function to $\mathbb{R}_+$ is $[0, \infty)$, so the $\exp(\cdot)$ will be restricted to $(-\infty, 0]$. Therefore, $\sigma_i \in [0, 1]$ as claimed.

(iv) As $\lambda \rightarrow 0$, $\sigma_i \rightarrow 1$ if clean, $\sigma_i \rightarrow 0$ otherwise:

The input of the Lambert W function

$$\lim_{\lambda \rightarrow 0^+} \frac{\ell_i^{\text{Proxy}} - \tau}{2\lambda} = \begin{cases} -\infty & \text{If } \ell_i^{\text{Proxy}} < \tau \\ \infty & \text{Otherwise} \end{cases} , \tag{7}$$

where the first case corresponds to the definition of clean. Note that it cannot happen that $\ell_i^{\text{Proxy}} = \tau$ as the possible thresholds are mid-points between consecutive loss values. For the first case

$$\lim_{x \rightarrow -\infty} \exp\left[-W\left(\max\{0, x\}\right)\right] \tag{8a}$$

$$= \exp\left[-W(0)\right] = \exp[0] = 1 , \tag{8b}$$

and for the second case

$$\lim_{x \rightarrow \infty} \exp\left[-W\left(\max\{0, x\}\right)\right] \tag{9a}$$

$$= \lim_{x \rightarrow \infty} \exp\left[-W(x)\right] \tag{9b}$$

$$= \exp[-\infty] = 0 . \tag{9c}$$

(v) As $\lambda \rightarrow \infty$, $\sigma_i \rightarrow 1$:

In this case, the input of the Lambert W function tends to 0, so we can leverage Eqs. (8a) and (8b).

□

As acknowledged in the paper, the expression in Eq. (3) is inspired by SuperLoss [5], which yields a simple and clean equation for the computation of the confidence that satisfies Conditions (i), (ii), and (v). In the remainder of this section, we focus on the key differences between our confidence score and that of SuperLoss. While these changes might seem subtle, they conceptually make a huge difference and significantly improve performance (see Tab. 1).

B.1. Constraining the confidence

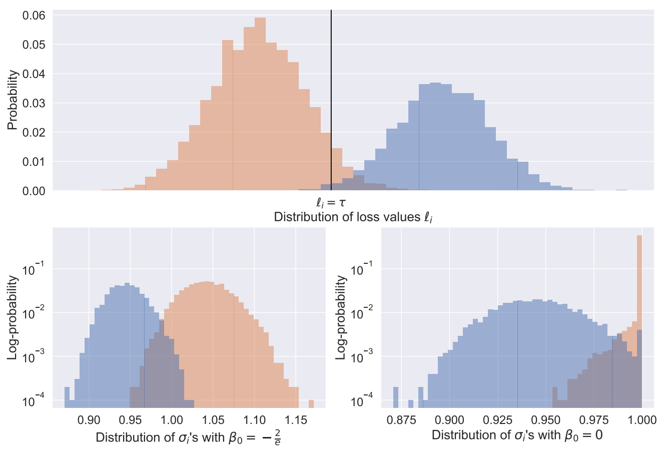
As stated in Condition (iii), we want to constrain the confidence $\sigma_i \in [0, 1]$. Plugging the constraint into the sample-level confidence version of Eq. (1) with constrained minimization, *i.e.*

$$\mathbb{E}_i \left[\min_{\sigma_i \in \Sigma} (\ell_i - \tau_i)\sigma_i + \lambda(\log \sigma_i)^2 \right] , \tag{10}$$

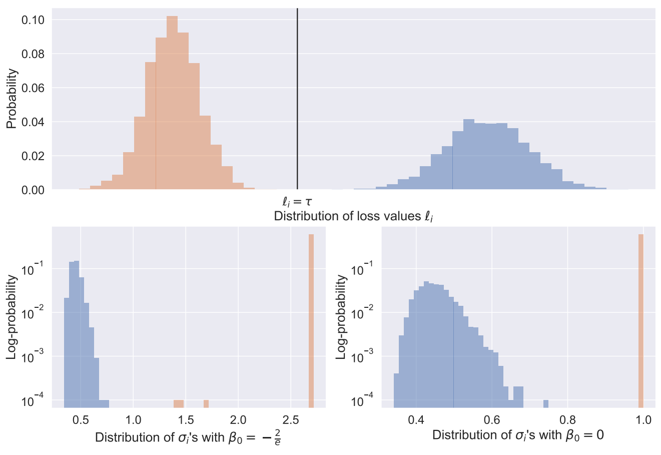
yields an analytical expression to compute the confidence score corresponding to

$$\sigma_i = \exp\left[-W\left(\frac{1}{2} \max\left\{ \beta_0, \frac{\ell_{ij} - \tau_{ij}}{\lambda} \right\} \right) \right] , \tag{11}$$

where $\beta_0 = -\frac{2}{e}$ when $\Sigma = \mathbb{R}$, as in SuperLoss [5], *cf.* Eq. (2). When $\Sigma = [0, 1]$ as required by Condition (iii) we obtain $\beta_0 = 0$. By constraining the confidence, we avoid over-weighting the samples with a low loss and, at the same time, obtain the following desirable properties:



(a) Non-separable distributions.



(b) Separable distributions.

Figure 4. Distribution of sample confidences computed using Eq. (2) with τ being the average of loss values and $\lambda = 1$. In this toy example, the loss values follow a mixture of two Gaussians, shown in different colors. We had to decrease the precision of floating point numbers from 64 to 32 bits to avoid numerical errors for $\beta_0 = -\frac{2}{e}$.

Asymptotic behavior: With $\beta_0 = -\frac{2}{e}$ as in SuperLoss, as $\lambda \to 0$, $\sigma_i \to 0$ if $\ell_i > \tau$, $\sigma_i \to e$ if $\ell_i < \tau$, and $\sigma_i \to 1$ if $\ell_i = \tau$. Instead, with $\beta_0 = 0$, we satisfy Condition (iv).

Numerical stability: The evaluation of $W(\cdot)$ can become inaccurate close to $-\frac{1}{e}$, the so-called branch point. Particularly at the branch point, attained at $\ell_i - \tau \leq \lambda\beta_0$ with $\beta_0 = -\frac{2}{e}$, the estimators used by well-known scientific computing libraries such as SciPy [72] can fail to converge. The choice $\beta_0 = 0$ avoids these numerical problems.

Fig. 4 presents a toy example illustrating the distributions with both values of β_0. When we have a bimodal distribution with separable modes (Fig. 4b), selecting $\beta_0 = 0$ assigns a confidence of 1 to all samples with loss belonging to the distribution of a smaller mean. If the small loss assumption is satisfied, these loss values probably belong to clean samples, so we don't want to alter their contribution. The confidence score for the other samples can be controlled by λ and be made arbitrarily close to 0.

In the non-separable case (Fig. 4a), using $\beta_0 = -2/e$ assigns diverse confidence scores to the samples belonging to the same distribution. By contrast, using $\beta_0 = 0$ assigns a unit confidence score to all values at the left of the threshold (the supposedly clean samples).

B.2. Thresholding

Even if the loss can differentiate a wrong label and follows the ideal bimodal distribution, we can see that the global average is not suited. In Fig. 5, we include a toy example to illustrate this observation, where we only consider one isolated iteration (so that the change of hard samples w.r.t. time is not an issue). Otsu's method selects τ based on the assumption that the distribution of losses is bimodal,

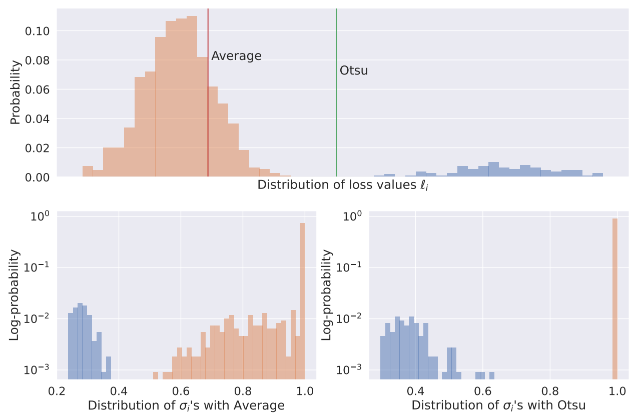
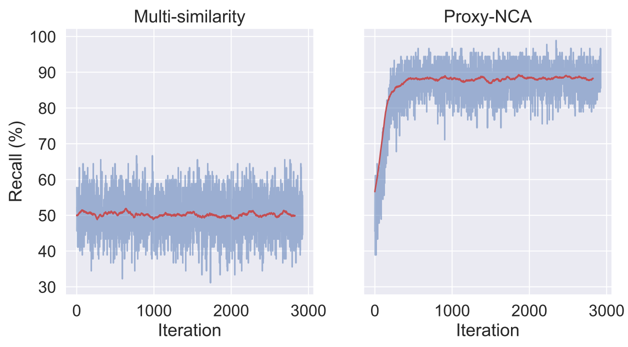


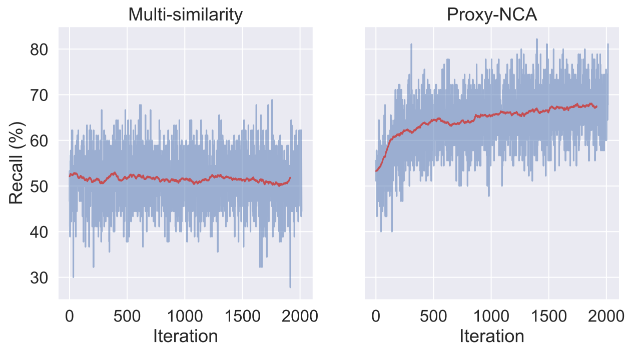
Figure 5. Distribution of sample confidences computed using Eq. (2) with τ being either the average as in SuperLoss [5] or Otsu's threshold [33] as in ProcSim. In both cases, we set $\beta_0 = 0$ and $\lambda = 0.1$. In this toy example, the loss values follow a mixture of two Gaussians, shown in different colors.

which allows for treating clean and noisy samples differently.

Regarding the change of hard samples across iterations, we can also justify the choice of τ with a simple example. Imagine that the distribution of sample values is the same but just gets shifted. In the usual case, loss values decrease as training progresses, so the global average is larger than the average at a given iteration. Under this scenario, the number of samples whose contribution will be reduced decreases at every iteration. That is precisely the idea of curriculum learning, in which harder samples are included progressively at later training stages. However, it is not justifiable from the perspective of discerning clean from noisy



(a) Results on CUB200 [49] with 50% uniform noise.



(b) Results on CUB200 [49] with 50% semantic noise.

Figure 6. Classification recall (%) for the task of noisy sample identification using Otsu's method [33]. The red line shows the moving average of the values obtained in a window of 100 iterations.

samples since the number of noisy labels in a dataset stays constant.

B.3. Confidence score and training loss

SuperLoss [5] proposes minimizing $\ell_i \sigma_i$ treating the confidence score σ_i as a constant and using any training objective ℓ_i for both the parameter update and the computation of σ_i. Consequently, if the training objective is composed of more than one term, they should be treated equally and as a whole. Instead, ProcSim applies different treatments to the supervised and self-supervised objectives implicated in the training loss. This simple modification is motivated by the fact that the self-supervised objective is unaffected by wrong annotations. Empirically this improves the DML performance, as seen in Tab. 1.

Another notable difference with SuperLoss [5] is that ProcSim disentangles the training loss and the objective used for the confidence computation. Doing so is similar to works relying on two independent models for unbiased noisy sample identification [15, 17, 23, 51, 59, 60]. Moreover, it allows using losses with different properties.

On the one hand, we use Proxy-NCA loss [30] for its usefulness in noise identification, which is justified from a probabilistic perspective in Sec. 3.2 and empirically in Fig. 2. For further evidence, even though ProcSim does not perform a hard classification into clean and noisy samples, we evaluated the usefulness of Otsu's method [33] over Proxy-NCA [30] and MS [50] in the task of noisy sample identification. Fig. 6 depicts the evolution of the classification recall during training. As expected, we cab correctly identify most noisy samples by thresholding the Proxy-NCA loss [30]. However, using the same procedure on the MS loss [50] results in random classification. When the injected noise follows the semantic model proposed in this paper, Fig. 6b shows that Proxy-NCA is also better at spotting noisy samples, although the classification recall is significantly lower than when using the uniform noise model. We expected this behavior as semantic noise generates wrong labels that are harder to identify.

On the other hand, as shown in Tab. 1, the base performance of Proxy-NCA [30] falls behind the MS loss [50]. At the same time, the MS loss is ineffective for spotting noisy samples, as shown in the example above. The ability to employ different and independent loss functions enhances the flexibility of ProcSim and enables us to leverage the strengths of various approaches, combining the best of both worlds.

Table 6. Recall@1 on the benchmark datasets for different levels of uniform noise when CLIP [37] image guidance is used instead of relying on an ImageNet classifier and an LLM. Inside the parentheses, we indicate the performance difference with ProcSim.

NOISE LEVEL →	10%	20%	50%
CUB200 [49]	67.5 (-1.8)	69.1 (-1.3)	60.5 (-0.3)
CARS196 [22]	86.4 (-0.8)	85.5 (-0.5)	74.4 (-0.8)
SOP [46]	79.0 (-0.3)	77.9 (-0.5)	73.2 (-0.1)

C. CLIP image embeddings

The PLG objective [40] is a clever way to consider semantics to determine inter-class relationships. However, the number of ImageNet classes determines the maximum number of distinct language embeddings we can obtain with this procedure. ImageNet [41] covers a wide range of items but, especially when using datasets with low inter-class variations such as SOP [46], thousands of different classes fall into the same ImageNet category. The semantic ambiguity of those classes given by the language guidance regularization hinders resolving inter-class relations. In general, when the domain of the downstream task has little overlap with ImageNet classes, the resolution of inter-class relationships is somehow limited.

ImageNet contains categories covering 2 or 3 classes in the CUB200 dataset [49], such as `hummingbird`, `albatross`, `jay`, and `pelican`. We can observe a

Table 7. Recall@1 (%) on the benchmark datasets corrupted with different probabilities of uniform noise. The reported results for all methods except ProcSim (ours) are taken from the PRISM paper [25] and rounded to one decimal place for consistency with the other tables. Best results are shown in **bold**. While MCL+PRISM [25] performs slightly better than ProcSim for low levels of noise on SOP [46], our method consistently and considerably outperforms it in the other datasets.

BENCHMARKS →	CUB200 [49]			CARS196 [22]			SOP [46]		
METHODS ↓	10%	20%	50%	10%	20%	50%	10%	20%	50%
Algorithms for image classification under label noise									
Co-teaching [15]	53.7	51.1	45.0	73.5	70.4	59.6	62.6	60.3	52.2
Co-teaching+ [59]	53.3	51.0	45.2	71.5	69.6	62.4	63.4	67.9	58.3
Co-teaching [15] w/ Temperature [61]	55.6	54.2	50.7	77.5	76.3	66.9	73.7	72.0	64.1
F-correction [35]	53.4	52.6	48.8	71.0	69.5	59.5	51.2	46.3	48.9
DML with Proxy-based Losses									
FastAP [4]	54.1	53.7	51.2	66.7	66.4	58.9	69.2	67.9	65.8
nSoftmax [61]	52.0	49.7	42.8	72.7	70.1	54.8	70.1	68.9	57.3
ProxyNCA [30]	47.1	46.6	41.6	69.8	70.3	61.8	71.1	69.5	61.5
Soft Triple [36]	51.9	49.1	41.5	76.2	71.8	52.5	68.6	55.2	38.5
DML with Pair-based Losses									
MS [50]	57.4	54.5	40.7	66.3	67.1	38.2	69.9	67.6	59.6
Circle [47]	47.5	45.3	13.0	71.0	56.2	15.2	72.8	70.5	41.2
Contrastive Loss [8]	51.8	51.5	38.6	72.3	70.9	22.9	68.7	68.8	61.2
MCL [52]	56.7	50.7	31.2	74.2	69.2	46.9	79.0	76.6	67.2
MCL + PRISM [25]	58.8	58.7	56.0	80.1	78.0	72.9	**80.1**	**79.5**	72.9
ProcSim (ours)	**69.3**	**70.4**	**60.8**	**87.2**	**86.0**	**75.2**	79.3	78.4	**73.3**

Table 8. Performance of methods with ResNet-50 [16] backbone and embedding dimension 512 on clean datasets. The best results are in **bold**. The results are taken from Roth *et al.* [40]. Inside the parentheses, we indicate the boost in performance of ProcSim w.r.t. the mean performance of MS+PLG, which is equivalent to setting unit confidence for all samples in the ProcSim framework (by letting $\lambda \to \infty$).

BENCHMARKS →	CUB200 [49]			CARS196 [22]			SOP [46]		
METHODS ↓	R@1	R@2	NMI	R@1	R@2	NMI	R@1	R@10	NMI
EPSHN [73]	64.9	75.3	-	82.7	89.3	-	78.3	90.7	-
NormSoft [61]	61.3	73.9	-	84.2	90.4	-	78.2	90.6	-
DiVA [65]	69.2	79.3	71.4	87.6	92.9	72.2	79.6	91.2	90.6
DCML-MDW [74]	68.4	77.9	71.8	85.2	91.8	73.9	79.8	90.8	90.8
IB-DML [69]	70.3	80.3	**74.0**	88.1	93.3	**74.8**	**81.4**	91.3	**92.6**
MS+*PLG* [40]	69.6 ± 0.4	79.5 ± 0.2	70.7 ± 0.1	87.1 ± 0.2	92.3 ± 0.3	73.0 ± 0.2	79.0 ± 0.1	91.0 ± 0.1	90.0 ± 0.1
S2SD+*PLG* [40]	**71.4** ± **0.3**	**81.1** ± **0.2**	73.5 ± 0.3	**90.2** ± **0.3**	**94.4** ± **0.2**	72.4 ± 0.3	81.3 ± 0.2	**92.3** ± **0.2**	91.1 ± 0.2
ProcSim (ours)	70.1 (+0.5)	79.6 (+0.1)	69.5 (-1.2)	87.7 (+0.6)	92.4 (+0.1)	72.2 (-0.8)	80.3 (+1.3)	91.4 (+0.4)	89.8 (-0.2)

similar coverage for the Cars196 dataset [22], in which, e.g., `sports car`, `cab`, `wagon`, `convertible`, `land rover`, `racing car`, and `minivan` are present in ImageNet. This coverage provides a level of specificity that allows differentiating some of the classes and assessing their similarity. However, for the SOP dataset [46], we find superclasses such as `stapler` or `kettle` that, although being ImageNet categories, account for thousands of different classes. While some superclasses such as `chair`, `cabinet`, and `lamp` have multiple ImageNet classes adequate for each, the instance retrieval nature of SOP and its large number of classes inside a superclass potentially reduces the knowledge transfer effectiveness.

Tab. 6 presents the results obtained using CLIP image embeddings [37] instead of relying on a classifier and a language model. In this case, we bypass the ImageNet classifier and directly obtain embeddings encoding semantic information from images without limiting the number of different embeddings. We can see that this approach performs on par with standard PLG on SOP [46] but underperforms it on the other datasets.

D. Additional comparisons

In Tab. 4, we compared ProcSim to the methods reported in the PRISM paper [25]. For the sake of space, we excluded the algorithms for image classification under label noise. However, it may be interesting to compare these methods, especially those derived from Co-teaching [15], which also relies on the small loss trick using the loss obtained by another model to have unbiased estimates. For this reason, we present all the results in Tab. 7.

ProcSim is a method for robust DML on noisy datasets. Nevertheless, for completeness, in Tab. 8, we include the obtained results on clean data side-by-side with state-of-the-art approaches. We present the methods with the same backbone architecture and embedding dimensionality as our current approach, as these are two of the main DML-independent drivers for generalization [68]. ProcSim offers comparable performance to state-of-the-art methods on clean data, although we focus on noisy datasets. In particular, ProcSim slightly improves the recall obtained without per-sample confidence, *i.e.*, MS+PLG [40].

Note that Normalized Mutual Information (NMI) slightly decreases when assigning confidence to samples. However, NMI varies across implementations and is sometimes uninformative [66], so this metric has to be interpreted with caution.

The best method for clean data is S2SD+PLG. S2SD [68] applies feature distillation between the output embeddings and embeddings computed with the so-called target networks, which results in higher-dimensional vectors. However, S2SD results in an objective expressed as a mean of losses for each target network. The fact that the mean is not over samples makes it incompatible with the presented framework.

E. Usage with state-of-the-art backbone

For a fair comparison, we performed all experiments using the standard ResNet-50 backbone [16]. Nonetheless, when trying to get the best results, one can leverage more powerful and expressive backbones using modern architectures such as transformers [71]. Swin transformers [64] are an example of these, and have been successfully applied to the visual retrieval task [67].

Tab. 9 shows the performance of ProcSim with Swin transformers [64] as the backbone model and the same hyperparameters used in the main paper for all the results with the ResNet-50 [16] backbone (see details in Appendix A, where we specify the values for each of the three benchmark datasets). With no fine-tuning, ProcSim outperforms the base MS loss under the presence of noise for the CUB200 [49] and the Cars196 [22] datasets. The difference in performance is monotonously increasing with the noise level and achieves an astounding increment of up to 23% Recall@1 for the Cars196 [22] dataset injected with 50% uniform noise.

Consistently with the results obtained in the paper, the performance on the SOP is somehow more limited. In this case, the base MS loss performs slightly better than ProcSim, although by at most 1.3% of Recall@1. By selecting $\omega = 0$ and $\lambda \to \infty$, ProcSim becomes MS. We could therefore match the performance of the plain MS loss and potentially obtain better results with some fine-tuning. However, we wanted to show the generalization capabilities of our method tailored only to each dataset regardless of the synthetic noise injected and the backbone.

F. Obtaining class hierarchies

Finding class hierarchies is posed as a graph traversal problem and solved by depth-first search. Given natural language class names, we use WordNet to search all their possible meanings (with synsets) and semantic superclasses (with hypernyms). We consider each synset as a graph node and the hypernyms as oriented edges and keep exploring the graph according to the depth-first search algorithm. Among all possible paths in the graph resulting from different meanings of the class name or its superclasses, we select the one with a common hypernym across all dataset classes. Once we find this path, we stop looking for more possible synsets and hypernyms.

Note that class hierarchies are not used during training when applying ProcSim. They are needed only for the semantic noise model proposed in this paper, which aims at showing the robustness capabilities of ProcSim on benchmark datasets corrupted with more realistic noise.

Below, we provide details to obtain the hierarchy of classes for each dataset. We also show visualizations of the obtained class hierarchies. In them, we suppressed the nodes in the graph with a single child for better visualization.

F.1. CUB200

The CUB200 [49] dataset provides natural language class names consisting of bird types, and thus the common hypernym is `bird`. We first preprocess the class names to satisfy the expected input of WordNet [29]. Some classes are not included in WordNet [29], in which case, we manually set the family of the species as a hypernym contained in the word corpora. Fig. 7 depicts the CUB200 [49] hierarchy found using the described procedure.

F.2. Cars196

The Cars196 dataset [22] contains classes whose common hypernym is `car` and have natural language names. However, the class labels contain other information like car type, brand, model, and year. Among all the class descriptors, only the car type is usable in WordNet [29]. Some

Table 9. Recall@1 (%) on the benchmark datasets corrupted with different types and probabilities of noise when Swin transformers [64] are used as backbone model. Best results shown in **bold**. Inside the parentheses, we indicate the boost in performance of ProcSim.

NOISE TYPE →	NONE	SEMANTIC			UNIFORM			
METHODS ↓	-	10%	20%	50%	10%	20%	30%	50%
CUB200 dataset [49]								
MS [50]	87.8	84.7	81.8	77.2	83.7	79.7	72.1	67.6
ProcSim (ours)	**88.4** (+0.6)	**88.4** (+3.7)	**88.5** (+6.7)	**87.8** (+10.6)	**88.1** (+4.4)	**88.2** (+8.5)	**87.1** (+15.0)	**84.7** (+17.1)
CARS196 dataset [22]								
MS [50]	**92.0**	88.9	85.0	71.5	88.9	83.3	78.1	46.7
ProcSim (ours)	90.5 (-1.5)	**89.3** (+0.4)	**88.3** (+3.3)	**85.1** (+13.6)	**89.6** (+0.7)	**87.5** (+4.2)	**84.1** (+6.0)	**69.7** (+23.0)
SOP dataset [46]								
MS [50]	**84.3**	**83.5**	**82.6**	**77.7**	**83.4**	**82.3**	**81.3**	**78.3**
ProcSim (ours)	84.2 (-0.1)	82.3 (-1.2)	82.3 (-0.3)	77.6 (-0.1)	83.0 (-0.4)	82.1 (-0.2)	81.1 (-0.2)	77.5 (-0.8)

brands may have different models of the same car type. Some models can also have different versions released over several years. With this in mind, we first group the classes by year, model, brand, and car type. Then, the car types are fed to WordNet [29] to find the complete class hierarchy, which we show in Fig. 8.

F.3. SOP

Unlike the other datasets, SOP [46] does not contain natural language class names. Instead, the class names consist of numerical identifiers of the product. The only natural language description is in the form of categories. Since training and testing partitions have multiple classes for each category, we can inject semantic noise by only relying on those.

Additional references

[62] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[63] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In *NeurIPS*, 1991. 1

[64] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 6, 7

[65] Timo Milbich, Karsten Roth, Homanga Bharadhwaj, Samarth Sinha, Yoshua Bengio, Björn Ommer, and Joseph Paul Cohen. Diva: Diverse visual feature aggregation for deep metric learning. In *ECCV*, 2020. 5

[66] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *ECCV*, 2020. 6

[67] Xu Ouyang, Tao Zhou, Rene Vidal, and Arnab Dhua. SwinTransFuse: Fusing Swin and Multiscale Transformers for Fine-grained Image Recognition and Retrieval. In *CVPR Workshop on Fine-Grained Visual Categorization*, 2022. 6

[68] Karsten Roth, Timo Milbich, Bjorn Ommer, Joseph Paul Cohen, and Marzyeh Ghassemi. Simultaneous similarity-based self-distillation for deep metric learning. In *ICML*, 2021. 6

[69] Jenny Denise Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. Learning intra-batch connections for deep metric learning. In *ICML*, 2021. 5

[70] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *ECCV*, 2020. 1

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 6

[72] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 3

[73] Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. In *WACV*, March 2020. 5

[74] Wenzhao Zheng, Chengkun Wang, Jiwen Lu, and Jie Zhou. Deep compositional metric learning. In *CVPR*, 2021. 5

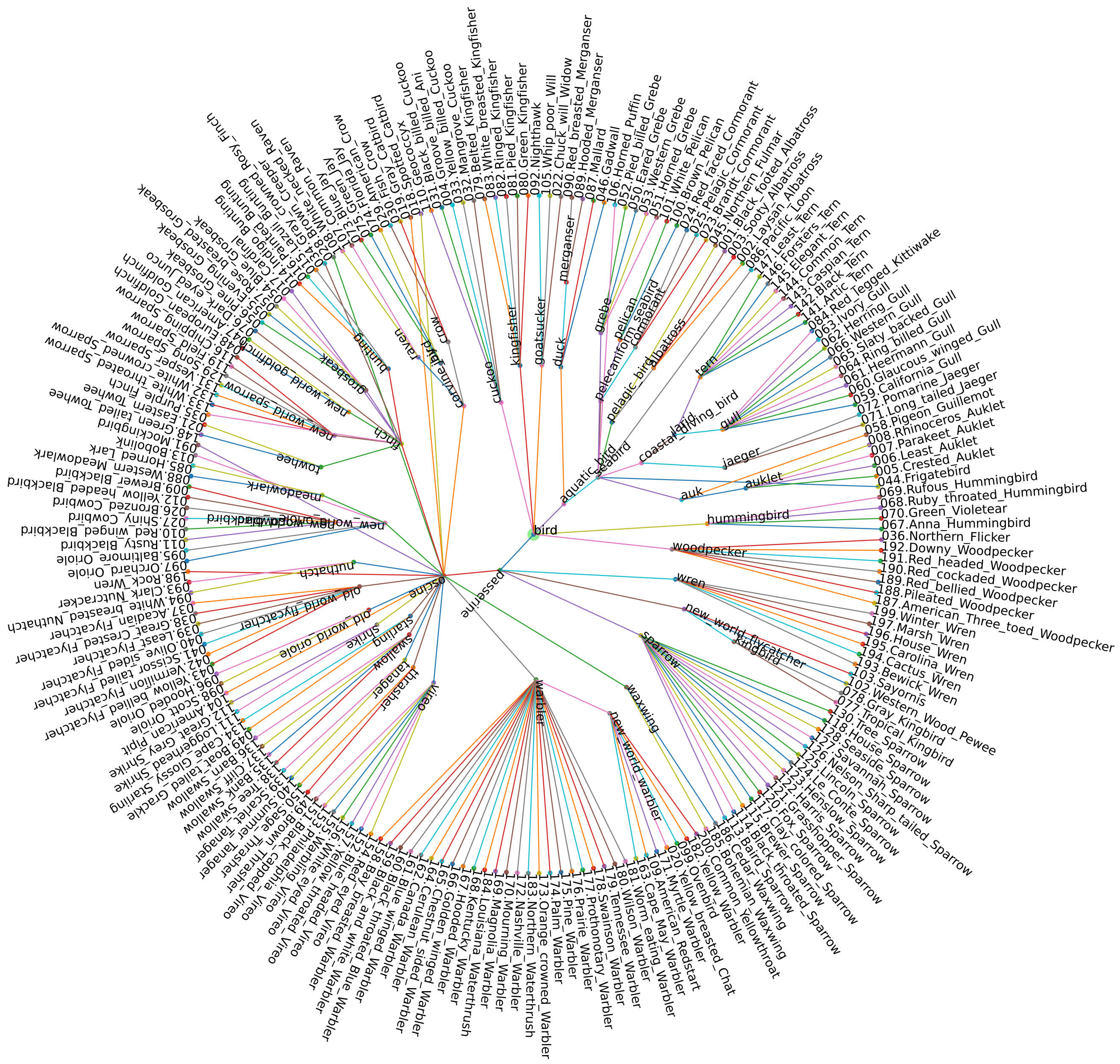


Figure 7. CUB200 [49] hierarchy.

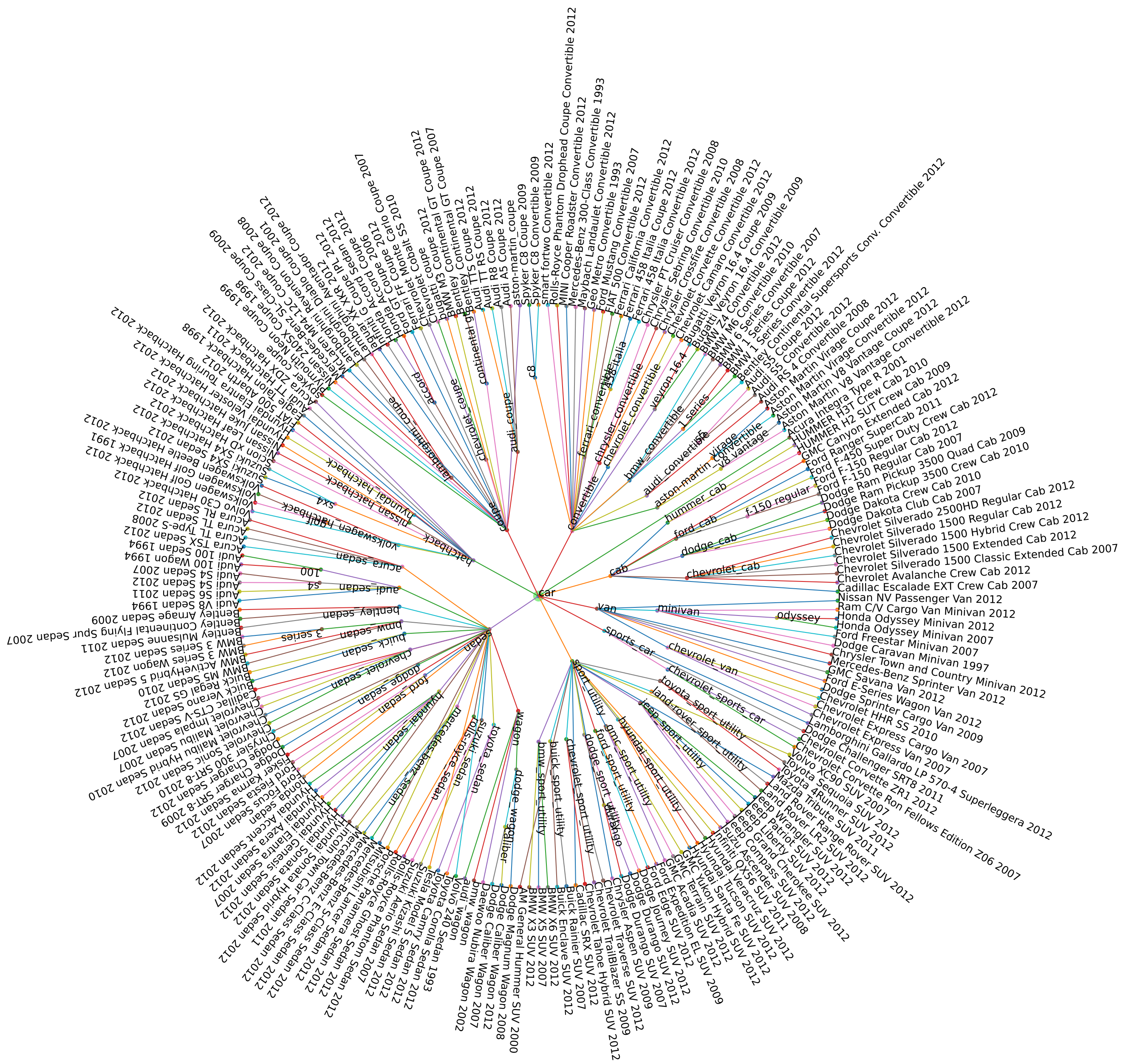


Figure 8. Cars196 [22] hierarchy.