Hypergraph-Induced Semantic Tuplet Loss for Deep Metric Learning
- *Supplementary Material* -

Jongin Lim[1, 2] Sangdoo Yun[3] Seulki Park[1] Jin Young Choi[1]
[1]ASRI, Dept. of ECE., Seoul National University
[2]Samsung Advanced Institute of Technology [3]NAVER AI Lab
{ljin0429,seulki.park,jychoi}@snu.ac.kr sangdoo.yun@navercorp.com

Appendix

In the following pages, we present experimental details (Section A), further analysis of the proposed method (Section B), comparison with state-of-the-art methods under other settings (Section C), and qualitative results (Section D).

A. Details of Experimental Setup

A.1. Datasets

We evaluated the proposed method on three widely used benchmarks for deep metric learning: CUB-200-2011 [26], CARS-196 [13], and Stanford Online Products (SOP) [19]. We split the datasets into training and test sets, according to the standard settings [14, 19]. **CUB-200-2011** contains 11,788 bird images of 200 different classes. We used 5,864 images of its first 100 classes for training and 5,924 images of the remaining 100 classes for testing. **CARS-196** contains 16,185 car images of 196 different classes. We used 8,054 images of its first 98 classes for training and 8,131 images of the remaining 98 classes for testing. **SOP** is a larger dataset than the aforementioned two datasets. SOP contains 120,053 product images of 22,634 classes. We used 59,551 images of its first 11,318 classes for training and 60,502 images of the remaining 11,316 classes for testing. It should be noted that we did not use any annotations (*e.g.*, bounding box or key point annotations) other than class labels for all datasets. The overall statistics of the datasets are summarized in Table 1.

Dataset	Num. of Images		Num. of Classes	
	Train	Test	Train	Test
CUB-200-2011 [26]	5,864	5,924	100	100
CARS-196 [13]	8,054	8,131	98	98
SOP [19]	59,551	60,502	11,318	11,316

Table 1. Dataset statistics.

A.2. Implementation Details

We used PyTorch [20] library for our implementation, and experiments were mainly conducted on a single Nvidia GTX 1080 Ti machine. We followed the *standard evaluation settings* [11, 19, 21, 27] and used constrained experimental settings for fair comparisons with previous works. The embedding network $E(.)$ consists of the backbone network and one fully connected layer. We considered BN-Inception [7] and ResNet-50 [6] pre-trained on ImageNet [1] as our backbone network. On top of the backbone network, a random initialized fully connected layer was attached for dimension reduction where we set the dimension of the output to 512. As in [24, 28], we applied layer normalization without affine parameters to obtain the final embedding. Tables 2 and 3 summarize the full list of implementation details when using BN-Inception and ResNet-50 as the backbone network, respectively. In addition, we will release the code after publication.

Module	Name	CUB-200-2011	CARS-196	SOP
HGNN	Num. layers	2	2	2
	Hidden dimension	512	512	512
	lr-HGNN	5e-4	5e-4	1e-3
$\mathbb{D}$	Initialization	He-normal	He-normal	He-normal
	lr-D	5e-2	1e-1	1e-2
Hyper-parameters	τ	16	24	16
	α	1	0.9	1.6
	λ_s	1	1	1
Training	Batch size	32	32	32
	Learning rate	1e-4	1e-4	1e-4
	Epochs	30	60	60
	Weight decay	1e-4	5e-5	1e-4
	Optimizer	AdamW	AdamW	AdamW
	lr scheduler	Step (5/0.5)	Step (10/0.5)	Step (10/0.5)
	BN freeze	True	True	False
	Warm-up	True	True	True

Table 2. Implementation details of our HIST loss when using **BN-Inception** as the backbone.

Module	Name	CUB-200-2011	CARS-196	SOP
HGNN	Num. layers	2	2	2
	Hidden dimension	512	512	512
	lr-HGNN	6e-4	1e-3	1e-3
$\mathbb{D}$	Initialization	He-normal	He-normal	He-normal
	lr-D	1e-1	1e-1	1e-2
Hyper-parameters	τ	32	32	16
	α	1.1	0.9	2
	λ_s	1	1	1
Training	Batch size	32	32	32
	Learning rate	1.2e-4	1e-4	1e-4
	Epochs	40	50	60
	Weight decay	5e-5	1e-4	1e-4
	Optimizer	Adam	Adam	Adam
	lr scheduler	Step (5/0.5)	Step (10/0.5)	Step (10/0.5)
	BN freeze	True	True	False
	Warm-up	True	True	True

Table 3. Implementation details of our HIST loss when using **ResNet-50** as the backbone.

Hypergraph neural network: For all experiments, 2-layer of HGNN [9] with the dimension of hidden units of 512 was used. The lr-HGNN denotes the initial learning rate for the HGNN. Unlike the embedding network $E(.)$, the HGNN was trained from scratch, so a larger learning rate was applied for faster convergence.

Prototypical distributions $\mathbb{D}$: We assigned mean $\boldsymbol{\mu}_c$ and covariance $\mathbf{Q}_c = \text{diag}(\mathbf{q}_c)$ for each class. To ensure computational stability, we assumed the log variance $\mathbf{v}_c$ instead of $\mathbf{q}_c$, *i.e.*, $\mathbf{v}_c = \log(\mathbf{q}_c)$. Then, $\boldsymbol{\mu}_c \in \mathbb{R}^D$ and $\mathbf{v}_c \in \mathbb{R}^D$, were initialized using He-normal initialization [5]. We also used a higher learning rate (lr-D) for these parameters for faster convergence.

Hyper-parameters: Following the recent methods [11, 24], we applied a large temperature scaling factor τ. We empirically determined α that controls the reflection ratio of negative samples. Besides, we used $\lambda_s = 1$ for all experiments.

Training details: For all experiments, the mini-batch size was set to 32 following [16, 17, 29]. The learning rate and the number of epochs were determined empirically. We also used L_2 regularization on the learnable parameters (weight decay). Following [11], we used AdamW (Adam with decoupled weight decay) [15] optimizer for BN-Inception and Adam [12] optimizer for ResNet-50. During training, the learning rate decreased by a factor of 0.5 for every 5 epochs for CUB-200-2011 and for every 10 epochs for CARS-196 and SOP, respectively. For CUB-200-2011 and CARS-196, BatchNorm (BN) parameters were frozen during training (BN freeze) to reduce over-fitting as in [11, 17]. In addition, we applied warm-up training for training stability as in [11, 24, 28], where only new parameters were trained for the first epoch.

A.3. Input Corruptions

In this section, we describe the detailed setup of the input corruptions used in the experiments presented in Section 4.3 of the manuscript. We used *imgaug* [10], a python library for image augmentation, to apply deformations to images. Specifically, we considered nine input corruptions of four types that were not used for training: additive noises (Gaussian, Salt & Pepper, and uniform noises), dropping pixels (cutout and dropout), affine transformation (rotation and perspective), and degrading image quality (Gaussian blur and JPEG-compression). Figure 1 shows the resulting images for each corruption. In the following, the detailed settings of each corruption are listed. For more details, please refer to *imgaug* [10].

- **Gaussian noise:** We added Gaussian noise to an image, sampled once per pixel from a normal distribution $\mathcal{N}(0, s)$, where s is sampled per image and varies between 0 and 0.1*255.
- **Salt & Pepper:** We replaced 5% of all pixels in an image with salt and pepper noise.
- **Uniform noise:** We added random values between -50 and 50 to images, with each value being sampled once per image and then being the same for all pixels (uniform).
- **Cutout:** We randomly replaced two random rectangle areas of each image with grayish pixels. The size of each rectangle is 20% of the input image size, and the height and width are randomly determined. For more details, please refer to [2].
- **Dropout:** We dropped 0 to 5% of all pixels by converting them to black pixels. Specifically, we applied Coarse Dropout [10] that leads to random rectangular areas being dropped.
- **Rotation:** We rotated images by a random value between -30° and 30°. Empty pixels due to rotation were filled with symmetrical padding.
- **Perspective transform:** We applied random four point perspective transformations to images. Specifically, we applied perspective transformations using a random scale between 0.05 and 0.15 per image, where the scale is roughly a measure of how far the perspective transformation's corner points may be distanced from the image's corner points.
- **Gaussian blur:** We blurred each image with a Gaussian kernel with a random sigma s that was sampled per image and varied between 1 and 3.
- **JPEG-compression:** We removed high frequency components in images via JPEG-compression with a compression strength between 80 and 95 (randomly and uniformly sampled per image). This degrades the quality of images.

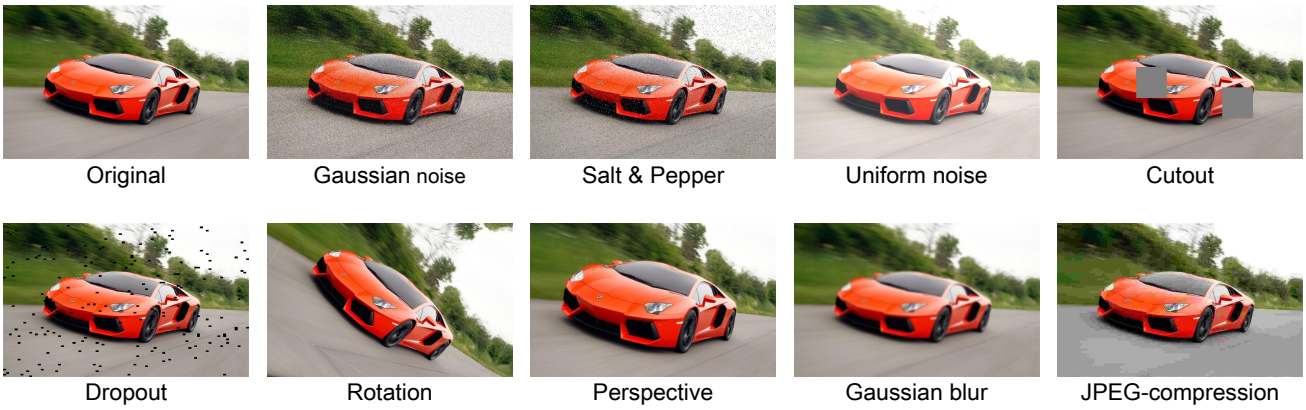


Figure 1. Examples of input corruptions used in the experiment. From left top to right bottom, we visualize the original image and the example results of nine input corruptions of four types: additive noises (Gaussian noise, Salt & Pepper, and uniform noise), dropping pixels (cutout and dropout), affine transformation (rotation and perspective transform), and degrading image quality (Gaussian blur and JPEG-compression). Best viewed when zoomed in.

B. Further Analysis

B.1. Ablation Studies on CUB-200-2011

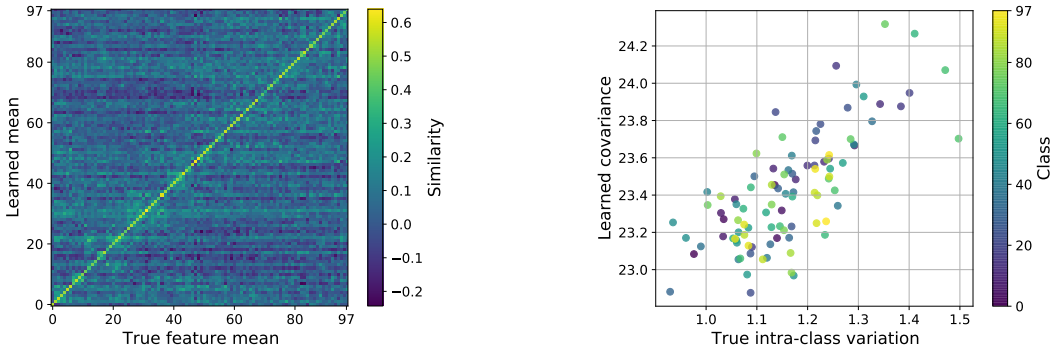
In Table 1 of the manuscript, we showed ablation studies on the CARS-196 dataset. Here, we report the same ablation studies on the CUB-200-2011 dataset. Table 4 shows the retrieval performance (R@1) of ablation models on the CUB-200-2011 dataset. Compared to the baseline and TF-like models, the proposed HIST model showed superior results, which demonstrates the benefits of leveraging higher-order data correlations provided by the proposed hypergraph modeling. H-Pos performed better than the baseline, but not the best, which indicates that exploiting the semantic relations of negative samples further improves performance. Lastly, with $\mathcal{L}_D$, the prototypical distributions better capture the true distribution and improve the quality of semantic tuplets, which provides a large performance gain.

Method	Relations	R@1
Single classification:		
Baseline	-	65.7 $\pm$ 0.3
Graph-based classification:		
TF-like	Transformer [25]-like attention	67.9 $\pm$ 0.3
Hypergraph-based classification:		
H-Pos	Only positive samples	67.4 $\pm$ 0.3
HIST (w.o. $\mathcal{L}_D$)	Semantic tuplets	65.5 $\pm$ 0.2
HIST	Semantic tuplets	**71.4** $\pm$ **0.2**

Table 4. Retrieval performance (R@1) of ablation models on the CUB-200-2011 dataset.

B.2. Prototypical Distributions

Can prototypical distributions capture the true feature distribution? We first investigated whether the learned prototype distributions capture the true feature distribution using the CAR-196 dataset. Figure 2a shows the similarity matrix between the learned means $\boldsymbol{\mu}_c$ (y-axis) and the actual means of the embedded features of each class, *i.e.*, $\frac{1}{N_c}\sum_{y_i=c}\mathbf{z}_i$, (x-axis) for all training classes $c \in \mathcal{C}$. The diagonal elements clearly show high similarities, which demonstrates that the learned means capture the true feature means of each class well. We then analyzed the correlation between the learned covariance matrices and the actual feature variations of each class. In Figure 2b, the Frobenius norm of the learned covariance matrix of each class $\|\mathbf{Q}_c\|_F$ (y-axis) and the average distance between the samples of each class and the learned mean, *i.e.*, $\frac{1}{N_c}\sum_{y_i=c}\|\boldsymbol{\mu}_c - \mathbf{z}_i\|_2^2$, (x-axis) exhibit a positive correlation. This indicates that the learned covariance well characterizes the intra-class variations in the embedding space. These results clearly support our claim that the learned prototypical distributions well characterize the true feature distribution.



(a) (b)

Figure 2. Investigations of the learned prototypical distributions $\mathbb{D}$ on the CARS-196 dataset. (a) shows the similarity matrix between the learned means and the true feature means. (b) shows the scatter plot between the Frobenius norms of the learned covariance and the true intra-class feature variations.

Prototypical distributions vs. Proxies. It should be noted that the proposed prototypical distributions can be viewed as an extended version of the proxies [11, 16, 21, 24, 29], which introduce covariance terms to model intra-class variations. In previous proxy-based methods, all features of the same class are generally represented by a single proxy. Thus, intra-class variations are ignored. While several methods [21, 29] introduce multiple proxies per class to handle intra-class variations, the complexity increases as the number of proxies per class increases. In contrast, we efficiently model intra-class variations by introducing covariance terms, which well characterize the true feature distribution as demonstrated in Figure 2. The better modeling of the true feature distribution helps to improve the quality of our semantic tuples, which leads to improved feature learning. To validate the effectiveness of the prototypical distributions, we designed the ablation model that replaces the prototypical distributions with the proxies (one proxy per class) and compared the performance. Table 5 shows the retrieval performance (R@1) on the CUB-200-2011 and CARS-196 datasets. For both models, we used ResNet-50 as the backbone network and followed the standard evaluation settings. The model with the proposed prototypical distributions showed superior performances for both datasets, which demonstrates the effectiveness of the proposed prototypical distributions.

Method	CUB-200-2011	CARS-196
HIST (with proxies)	69.4 ± 0.4	87.8 ± 0.3
HIST (with prototypical distributions)	$\mathbf{71.4 \pm 0.2}$	$\mathbf{89.6 \pm 0.2}$

Table 5. Comparison of retrieval performance (R@1) when using proxies and prototypical distributions.

B.3. Balanced Mini-batch

Considering that our mini-batch is sampled completely randomly, there might be little communication between samples of the same class. Therefore, we further conducted experiments with the *balanced* mini-batch sampling as in [3,22]. Specifically, each mini-batch was constructed by first randomly sampling 8 classes and then randomly sampling 4 images for each selected class. Table 6 shows the retrieval performance (R@1) comparisons for different sampling schemes. For both models, we used ResNet-50 as the backbone network and followed the standard evaluation settings. Notably, the balanced sampling showed additional performance gain on the CUB-200-2011 dataset. Yet, for simplicity, we used a randomly sampled mini-batch for our experiments.

Mini-batch sampling	CUB-200-2011	CARS-196
Random (32)	71.4 ± 0.2	$\mathbf{89.6 \pm 0.2}$
Balanced (4×8)	$\mathbf{71.7 \pm 0.2}$	89.1 ± 0.3

Table 6. Retrieval performance (R@1) according to mini-batch sampling.

B.4. Different Embedding Dimensions

The dimensionality of the embedding vector is a key factor in deep metric learning that controls the trade-off between performance and computational cost. We investigated the effect of the embedding dimensions on the CARS-196 dataset. We evaluated the retrieval performance (R@1) with embedding dimensions varying from 64 to 1024. We used ResNet-50 as the backbone network and followed the standard evaluation settings. Table 7 summarizes the results. We confirmed that the performance improved as the embedding size increased.

Dimension	64	128	256	512	1024
R@1	80.4 ± 0.3	83.1 ± 0.2	88.1 ± 0.3	89.6 ± 0.2	$\mathbf{89.7 \pm 0.2}$

Table 7. Retrieval performance (R@1) with different embedding dimensions on the CARS-196 dataset.

C. Comparison with State-of-the-art Methods

C.1. MLRC [17] Evaluation

In this work, we followed the *standard evaluation settings* [11, 19, 21, 27] and used constrained experimental settings for fair comparisons with previous deep metric learning methods. Recently, several studies [4, 17] have questioned these conventional experimental settings. Specifically, a paper named *A Metric Learning Reality Check* [17] pointed out the flaws of the existing evaluation settings, including unfair comparisons with a more powerful backbone, uninformative evaluation metrics, and training with test set feedback. Moreover, the authors presented new evaluation settings, which we refer to as *MLRC evaluation settings*, and introduced new evaluation metrics, RP and MAP@R. To improve the credibility of our experimental evaluation, we additionally conducted experiments on the CUB-200-2011 and CARS-196 datasets by strictly following the MLRC evaluation settings (please refer to the original paper [17] for detailed evaluation settings). Tables 8 and 9 summarize the overall results under the MLRC evaluation settings on the CUB-200-2011 and CARS-196 datasets, respectively. In all experiments, our HIST loss achieved state-of-the-art performance.

Method	Concatenated (512-dim)			Seperated (128-dim)		
	P@1	RP	MAP@R	P@1	RP	MAP@R
Contrastive	68.13 ± 0.31	37.24 ± 0.28	26.53 ± 0.29	59.73 ± 0.40	31.98 ± 0.29	21.18 ± 0.28
Triplet	64.24 ± 0.26	34.55 ± 0.24	23.69 ± 0.23	55.76 ± 0.27	29.55 ± 0.16	18.75 ± 0.15
NT-Xent	66.61 ± 0.29	35.96 ± 0.21	25.09 ± 0.22	58.12 ± 0.23	30.81 ± 0.17	19.87 ± 0.16
ProxyNCA	65.69 ± 0.43	35.14 ± 0.26	24.21 ± 0.27	57.88 ± 0.30	30.16 ± 0.22	19.32 ± 0.21
Margin	63.60 ± 0.48	33.94 ± 0.27	23.09 ± 0.27	54.78 ± 0.30	28.86 ± 0.18	18.11 ± 0.17
Margin/class	64.37 ± 0.18	34.59 ± 0.16	23.71 ± 0.16	55.56 ± 0.16	29.32 ± 0.15	18.51 ± 0.13
N. Softmax	65.65 ± 0.30	35.99 ± 0.15	25.25 ± 0.13	58.75 ± 0.19	31.75 ± 0.12	20.96 ± 0.11
CosFace	67.32 ± 0.32	37.49 ± 0.21	26.70 ± 0.23	59.63 ± 0.36	31.99 ± 0.22	21.21 ± 0.22
ArcFace	67.50 ± 0.25	37.31 ± 0.21	26.45 ± 0.20	60.17 ± 0.32	32.37 ± 0.17	21.49 ± 0.16
FastAP	63.17 ± 0.34	34.20 ± 0.20	23.53 ± 0.20	55.58 ± 0.31	29.72 ± 0.16	19.09 ± 0.16
SNR	66.44 ± 0.56	36.56 ± 0.34	25.75 ± 0.36	58.06 ± 0.39	31.21 ± 0.28	20.43 ± 0.28
MS	65.04 ± 0.28	35.40 ± 0.12	24.70 ± 0.13	57.60 ± 0.24	30.84 ± 0.13	20.15 ± 0.14
MS+Miner	67.73 ± 0.18	37.37 ± 0.19	26.52 ± 0.18	59.41 ± 0.30	31.93 ± 0.15	21.01 ± 0.14
SoftTriple	67.27 ± 0.39	37.34 ± 0.19	26.51 ± 0.20	59.94 ± 0.33	32.12 ± 0.14	21.31 ± 0.14
HIST (Ours)	**69.61 ± 0.28**	**38.83 ± 0.12**	**28.21 ± 0.12**	**61.34 ± 0.18**	**33.13 ± 0.15**	**22.30 ± 0.14**

Table 8. Results on the CUB-200-2011 dataset under the MLRC evaluation settings. For all compared methods, the results were quoted from [17]. The best results are marked in **bold**, and the second-best results are underlined.

Method	Concatenated (512-dim)			Seperated (128-dim)		
	P@1	RP	MAP@R	P@1	RP	MAP@R
Contrastive	81.78 ± 0.43	35.11 ± 0.45	24.89 ± 0.50	69.80 ± 0.38	27.78 ± 0.34	17.24 ± 0.35
Triplet	79.13 ± 0.42	33.71 ± 0.45	23.02 ± 0.51	65.68 ± 0.58	26.67 ± 0.36	15.82 ± 0.36
NT-Xent	80.99 ± 0.54	34.96 ± 0.38	24.40 ± 0.41	68.16 ± 0.36	27.66 ± 0.23	16.78 ± 0.24
ProxyNCA	83.56 ± 0.27	35.62 ± 0.28	25.38 ± 0.31	73.46 ± 0.23	28.90 ± 0.22	18.29 ± 0.22
Margin	81.16 ± 0.50	34.82 ± 0.31	24.21 ± 0.34	68.24 ± 0.35	27.25 ± 0.19	16.40 ± 0.20
Margin/class	80.04 ± 0.61	33.78 ± 0.51	23.11 ± 0.55	67.54 ± 0.60	26.68 ± 0.40	15.88 ± 0.39
N. Softmax	83.16 ± 0.25	36.20 ± 0.26	26.00 ± 0.30	72.55 ± 0.18	29.35 ± 0.20	18.73 ± 0.20
CosFace	85.52 ± 0.24	37.32 ± 0.28	27.57 ± 0.30	74.67 ± 0.20	29.01 ± 0.11	18.80 ± 0.12
ArcFace	85.44 ± 0.28	37.02 ± 0.29	27.22 ± 0.30	72.10 ± 0.37	27.29 ± 0.17	17.11 ± 0.18
FastAP	78.45 ± 0.52	33.61 ± 0.54	23.14 ± 0.56	65.08 ± 0.36	26.59 ± 0.36	15.94 ± 0.34
SNR	82.02 ± 0.48	35.22 ± 0.43	25.03 ± 0.48	69.69 ± 0.46	27.55 ± 0.25	17.13 ± 0.26
MS	85.14 ± 0.29	38.09 ± 0.19	28.07 ± 0.22	73.77 ± 0.19	29.92 ± 0.16	19.32 ± 0.18
MS+Miner	83.67 ± 0.34	37.08 ± 0.31	27.01 ± 0.35	71.80 ± 0.22	29.44 ± 0.21	18.86 ± 0.20
SoftTriple	84.49 ± 0.26	37.03 ± 0.21	27.08 ± 0.21	73.69 ± 0.21	29.29 ± 0.16	18.89 ± 0.16
HIST (Ours)	**87.73 ± 0.22**	**39.99 ± 0.20**	**30.46 ± 0.24**	**79.30 ± 0.24**	**32.78 ± 0.17**	**22.34 ± 0.19**

Table 9. Results on the CARS-196 dataset under the MLRC evaluation settings. For all compared methods, the results were quoted from [17]. The best results are marked in **bold**, and the second-best results are underlined.

C.2. Smaller Embedding Dimension

In the manuscript, we compared the performance when the feature dimension is 512, the most common setting. Here, we further evaluated the performance when the feature dimension is 64, which is the second most common setting. We used ResNet-50 as the backbone network and followed the standard evaluation settings. We compared the retrieval performance of the proposed method with those of the state-of-the-art methods on the CUB-200-2011, CARS-196, and SOP datasets. Table 10 summarizes the overall results. For all datasets, the model trained with our HIST loss showed superior performances. This further validates the effectiveness of the proposed method.

Method	CUB-200-2011				CARS-196				SOP			
	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@10^2	R@10^3
LiftedStruct^{64} [19]	43.6	56.6	68.6	79.6	53.0	65.7	76.0	84.3	62.5	80.8	91.9	-
N-pair-mc^{64} [23]	51.0	63.3	74.3	83.2	71.1	79.7	86.5	91.6	67.7	83.8	93.0	97.8
Clustering^{64} [18]	48.2	61.4	71.8	81.9	58.1	70.6	80.3	87.8	67.0	83.7	93.2	-
ProxyNCA^{64} [16]	49.2	61.9	67.9	72.4	73.2	82.4	86.4	88.7	73.7	-	-	-
MS^{64} [27]	57.4	69.8	80.0	87.8	77.3	85.3	90.5	94.2	74.1	87.8	94.7	98.2
SoftTriple^{64} [21]	60.1	71.9	81.2	88.5	78.6	86.6	91.8	95.4	76.3	89.1	95.3	-
ProxyAnchor^{64} [11]	61.7	73.0	81.8	88.8	78.8	87.0	92.2	**95.5**	76.5	89.0	95.1	98.2
ProxyGML^{64} [29]	59.4	70.1	80.4	-	78.9	87.5	91.9	-	76.2	89.4	95.4	-
HIST^{64} (Ours)	**62.5**±0.2	**73.6**±0.2	**83.0**±0.2	**89.6**±0.1	**80.4**±0.3	**87.6**±0.2	**92.4**±0.2	95.4±0.2	**78.9**±0.3	**90.5**±0.2	**95.8**±0.1	**98.5**±0.0

Table 10. Recall@K (%) on CUB-200-2011, CARS-196, and SOP datasets. Superscript denotes the embedding size. For all compared methods, the results were quoted from the original paper. For our method, we reported the average performance with 95% confidence interval evaluated over 10 independent runs. The best results are marked in **bold**, and the second-best results are underlined.

C.3. Larger Image Size

In the standard evaluation settings, the default image size is set to 224×224 in deep metric learning. However, several recent methods [8, 11, 24] utilized larger images with the size of 256×256. The image size has a significant influence on the model performance. For a fair comparison to these methods, we further evaluated the performance using 256×256 images. We used ResNet-50 as the backbone network and followed the standard evaluation settings except for the image size. As shown in Table 11, the proposed method significantly outperformed the other methods by a large margin for all datasets. Notably, the proposed method using 224×224 images already achieved better performance than the other methods using 256×256 images, which further demonstrates the superiority of the proposed method.

Method	CUB-200-2011				CARS-196				SOP			
	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@10^2	R@10^3
†HORDE^{512} [8]	66.3	76.7	84.7	90.6	83.9	90.3	94.1	96.3	80.1	91.3	96.2	98.7
†ProxyNCA++^{512} [24]	69.0	79.8	87.3	92.7	86.5	92.5	95.7	97.7	80.7	92.0	96.7	**98.9**
†ProxyAnchor^{512} [11]	71.1	80.4	87.4	92.5	88.3	93.1	95.7	97.5	80.3	91.4	96.4	98.7
HIST^{512} (Ours)	71.4±0.2	81.1±0.3	88.1±0.2	92.7±0.1	89.6±0.2	93.9±0.1	96.4±0.1	97.8±0.3	81.4±0.2	92.0±0.2	96.7±0.1	**98.9**±0.0
†HIST^{512} (Ours)	**73.0**±0.3	**82.2**±0.2	**88.8**±0.2	**93.2**±0.1	**90.6**±0.2	**94.7**±0.2	**96.9**±0.1	**98.2**±0.1	**81.5**±0.2	**92.2**±0.2	**96.8**±0.1	**98.9**±0.1

Table 11. Recall@K (%) on CUB-200-2011, CARS-196, and SOP datasets. Superscript denotes the embedding size, and "†" indicates that the model uses a larger input image (256×256). For all compared methods, the results were quoted from the original paper. For our method, we reported the average performance with 95% confidence interval evaluated over 10 independent runs. The best results are marked in **bold**, and the second-best results are underlined.

D. Qualitative Results

D.1. Visualization of Feature Activation Maps

In Section 4.3 of the manuscript, we investigated the feature activation maps from the last convolutional layer of the learned embedding network for test images and showed two examples. To demonstrate the consistency of our results, we present more results on CUB-200-2011 in Figure 3 and CARS-196 in Figure 4. As can be seen, the proposed method better attended to the object region than the baseline. This observation implies that our HIST loss would make the embedding network capture important semantics from the image.

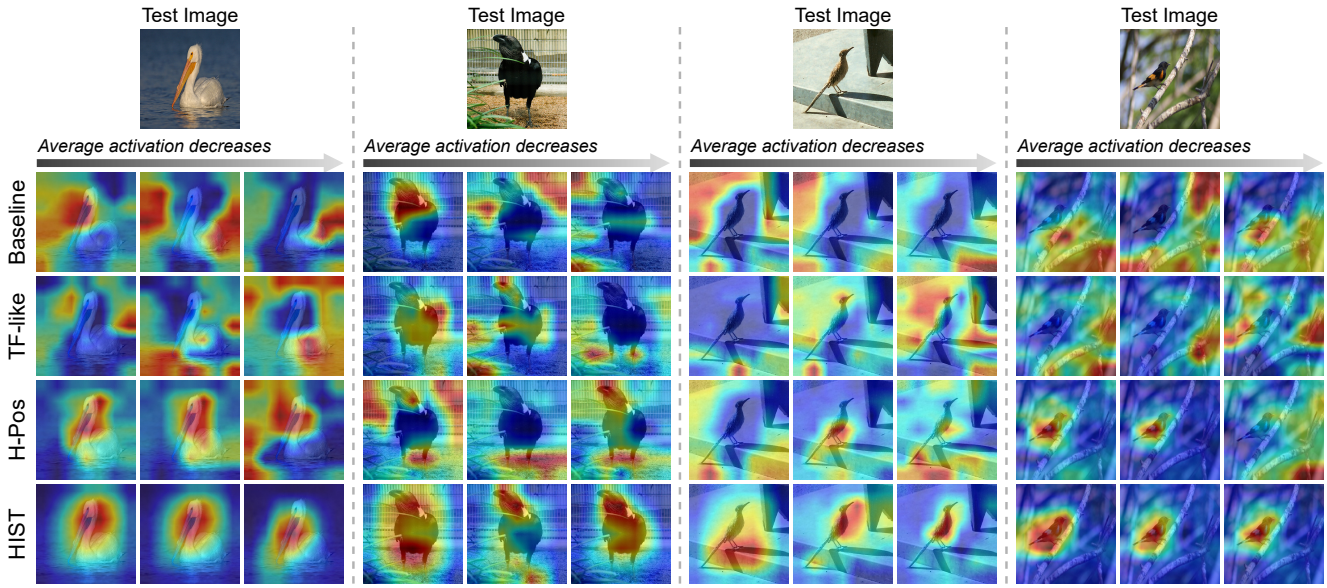


Figure 3. Visualization of three channels of the last feature activation maps which have the maximal average activation values on the CUB-200-2011 dataset.

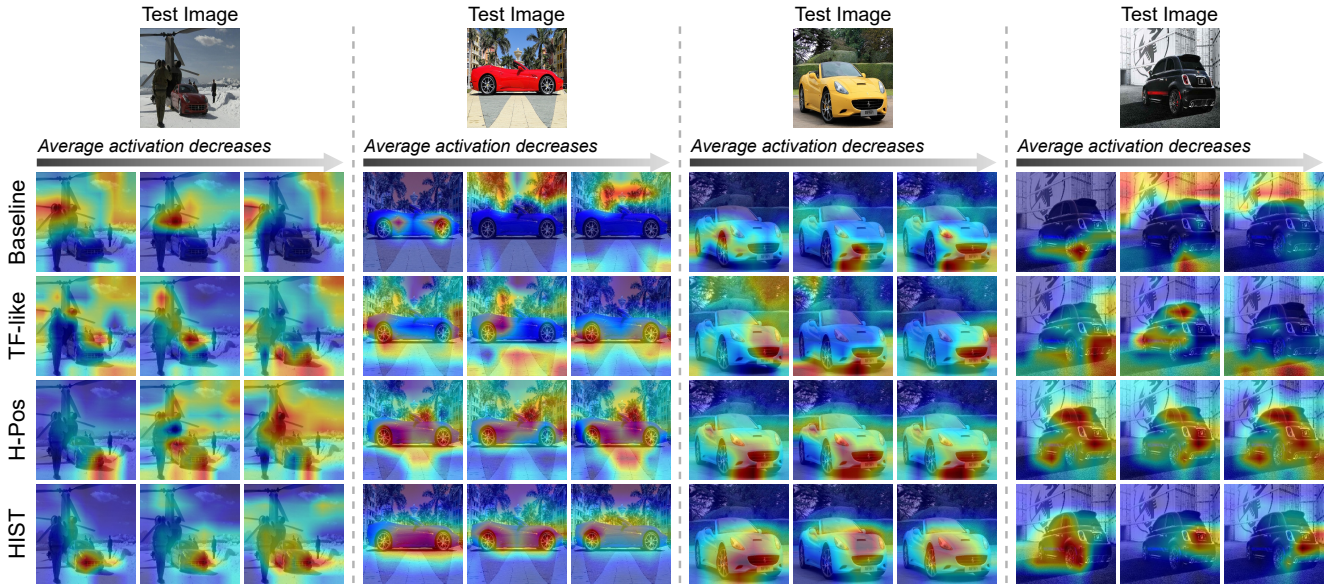


Figure 4. Visualization of three channels of the last feature activation maps which have the maximal average activation values on the CARS-196 dataset.

D.2. Retrieval Results

In this section, we present qualitative retrieval results on the CUB-200-2011, CARS-196, and SOP datasets. Figure 5, 6, and 7 show retrieval results on CUB-200-2011, CARS-196, and SOP datasets, respectively. Specifically, we visualize the top-4 nearest images along with the query image. The green box indicates that the retrieved image is of the same class as the query image, and the red box indicates that the image is of a different class from the query image. We can observe that the proposed method clearly finds images that are visually similar to the query images no matter that the retrieval result is correct or incorrect. The overall results clearly demonstrate the effectiveness of the proposed method.

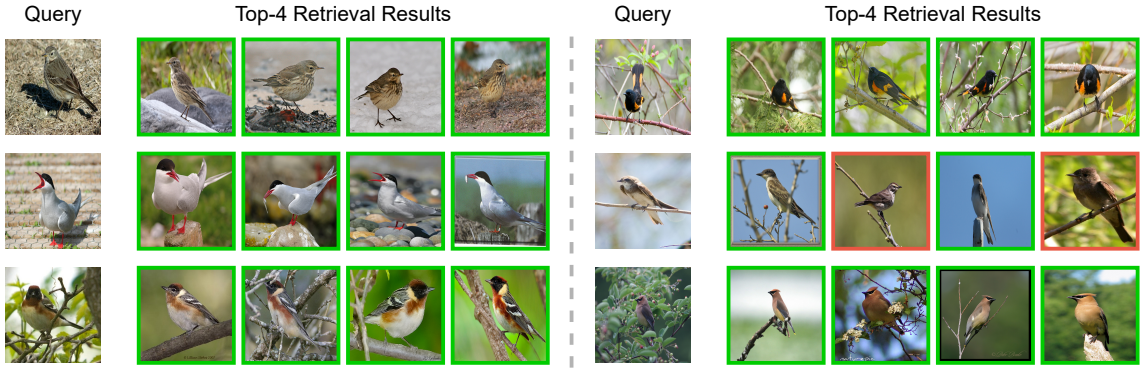


Figure 5. Retrieval results on the CUB-200-2011 dataset.

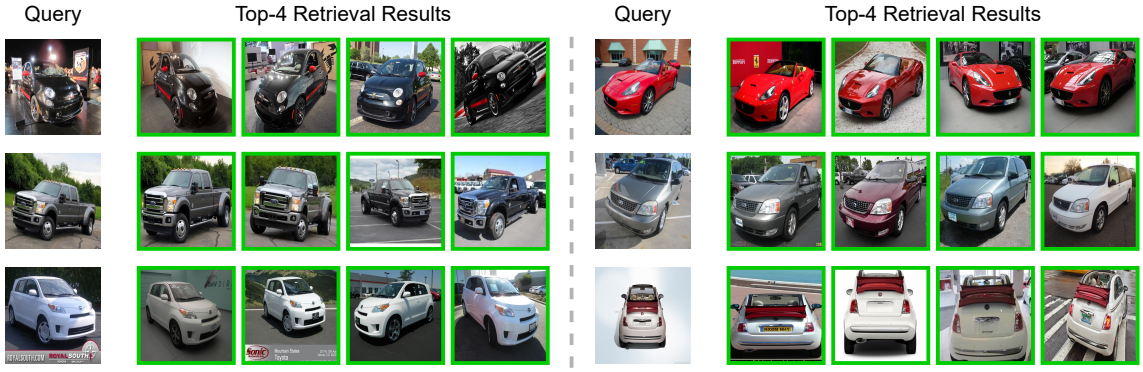


Figure 6. Retrieval results on the CARS-196 dataset.

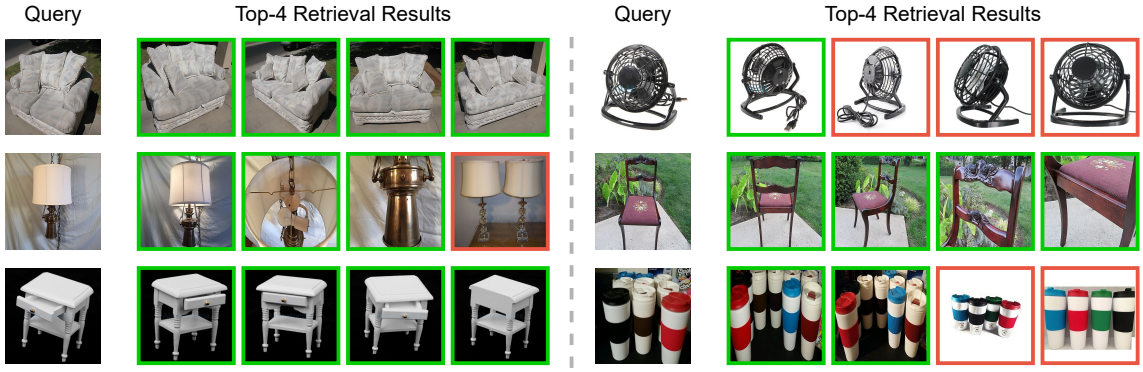


Figure 7. Retrieval results on the SOP dataset.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 3

[3] Ismail Elezi, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, and Laura Leal-Taixé. The group loss for deep metric learning. In *European Conference on Computer Vision*, pages 277–294. Springer, 2020. 5

[4] Istvan Fehervari, Avinash Ravichandran, and Srikar Appalaraju. Unbiased evaluation of deep metric learning algorithms. *arXiv preprint arXiv:1911.12528*, 2019. 6

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1

[8] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6539–6548, 2019. 7

[9] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *IJCAI*, pages 2635–2641, 2019. 2

[10] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020. 3

[11] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3238–3247, 2020. 1, 2, 5, 6, 7

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[13] 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. 1

[14] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016. 1

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[16] 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, 5, 7

[17] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020. 2, 6

[18] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017. 7

[19] 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, 6, 7

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019. 1

[21] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6450–6458, 2019. 1, 5, 6, 7

[22] Jenny Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. Learning intra-batch connections for deep metric learning. *arXiv preprint arXiv:2102.07753*, 2021. 5

[23] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016. 7

[24] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*. Springer, 2020. 1, 2, 5, 7

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4

[26] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 1

[27] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 1, 6, 7

[28] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018. 1, 2

[29] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is more: A deep graph metric learning perspective using fewer proxies. In *NeurIPS*, 2020. 2, 5, 7