HIER: Metric Learning Beyond Class Labels via Hierarchical Regularization —Supplementary Material—

Sungyeon Kim[1] Boseung Jeong[1] Suha Kwak[1,2]
Dept. of CSE, POSTECH[1] Graduate School of AI, POSTECH[2]
{sungyeon.kim, boseung01, suha.kwak}@postech.ac.kr
http://cvlab.postech.ac.kr/research/HIER

A. Appendix

This supplementary material provides further analyses, additional experimental results, and implementation details that are left out from the main paper due to the space limit.

A.1. Effect of Deploying Semantic Hierarchy

In this section, we investigate the effect of deploying semantic hierarchies by utilizing proxies. In particular, we quantitatively compare our method with SoftTriple loss [6] which can model intra-class variance of data by utilizing multiple proxies per class. Table 2 shows the performance of two proxy-based methods and the number of proxies they used. In SoftTriple loss, 10 proxies are assigned per class, and our method uses 512 hierarchical proxies in addition to proxies for proxy anchor loss.

The result shows that SoftTriple loss lags significantly in performance despite using a much larger amount of proxies compared to our method. The main difference between these methods is how they handle proxies. SoftTriple assigns multiple proxies per class, so their proxies can only represent sub-classes of data and model intra-class variance. On the other hand, proxies in our method can represent sub-classes or super-classes as well as predefined classes, thus allowing more flexibility in modeling more complex relations of data beyond intra-class variance.

A.2. Embedding Space Visualization

We further visualize the embedding space at the beginning of training, such as Epoch 1, 3, 5, 7, and 10. The results of visualization presented in Figure 1 show that the earlier embedding space does not construct a hierarchical structure between the embedding vectors and hierarchical proxies, while this structure is gradually constructed as the training epoch grows. Specifically, as training progresses, the hierarchical proxies have ancestor-descendant relations between sample or other proxies, where these proxies can be regarded as predefined classes, sub-classes, and super-classes. As a consequence, our method can discover the latent semantic hierarchy of training data, which provides rich and granular supervision beyond human-labeled classes.

Hyperparameters	CUB	Cars	SOP	In-Shop
total epochs	50	50	150	150
warm-up epochs	1	1	5	5
LR of the last layer	$\times 1$	$\times 1$	$\times 10^2$	$\times 10^2$
weight decay	$1e{-2}$	$1e{-2}$	$1e{-4}$	$1e{-4}$

Table 1. Additional details of hyperparameters for network optimization. LR denotes the learning rate, and its value denotes how many times higher than the original learning rate.

A.3. More Qualitative Results

We further verify the effect of HIER in a qualitative perspective. To this end, we present the qualitative results of our method, compared to those of a model optimized by only a metric learning loss, $\mathcal{L}_{\text{ML}}$ in Eq. (9) of the main paper. We take proxy anchor loss [3] as the metric learning loss. Figure 2 presents the qualitative results for the four public benchmark datasets, CUB [9], Cars [4], SOP [7] and In-Shop [5]. These results demonstrate that our method is robust against small inter-class variance (CUB and SOP), viewpoint variation and distinct color (Cars), and large intra-class variations and viewpoint changes (In-Shop) by discovering and deploying a latent semantic hierarchy of data, while the proxy anchor still suffers from those problems.

We note that all the results in the figure are obtained from the fully unseen class samples. It implies that it allows the proposed hierarchical regularizer to discover the latent semantic hierarchy of data with no additional annotation for the hierarchy; our method allows the embedding space to be generalized well.

A.4. Additional Implementation Details

Since each dataset that we utilize to evaluate our method has a different characteristic, we set different hyperparame-

	CUB			Cars			SOP			In-Shop		
Methods	#Proxies	R@1	R@2	#Proxies	R@1	R@2	#Proxies	R@1	R@10	#Proxies	R@1	R@10
SoftTriple [6]	**1,000**	72.7	82.7	**980**	83.2	90.2	**113,180**	80.9	91.3	**39,970**	88.5	97.3
PA + HIER (ours)	612	**75.2**	**84.2**	610	**85.1**	**91.2**	11,830	**82.5**	**92.7**	4,509	**91.0**	**98.0**

Table 2. Comparison between methods using proxies in terms of performance and the number of proxies on the four benchmark datasets. In these experiments, the backbone network is initialized by weights of DeiT [8].

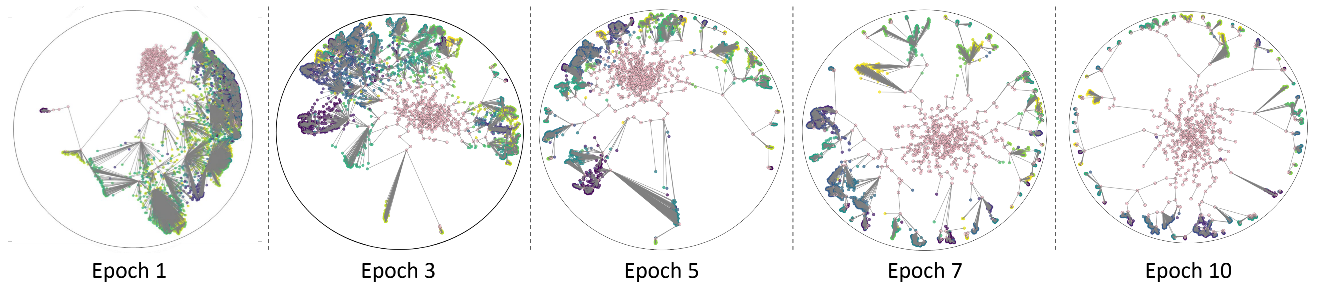

Figure 1. UMAP visualizations of our embedding space learned on the train split of the Cars dataset at different epochs. Pink points indicate hierarchical proxies, and other colors represent distinct classes. The gray line indicates the ancestor-descendant relation between the hierarchical proxy and data points.

ters for network optimization according to each dataset. The summary of the settings of hyperparameter for four datasets are presented in Table 1. For all backbone network variants (*i.e.*, ViT-S [2], DeiT-S [8], and DINO [1]), our model is trained with 50 epochs on CUB [9] and Cars [4], and 150 epochs on SOP [7] and In-Shop [5]. As a warm-up strategy, the last layer which consists of a linear layer followed by the exponential mapping layer is only trained, while the pretrained backbone network is not updated; the warm-up strategy is applied for 1 epoch on CUB and Cars, and 5 epochs on SOP and In-Shop. On the other hand, we use a high learning rate for the last layer (*i.e.*, embedding layer) by scaling 10^2 times for SOP and In-Shop. Furthermore, we set the weight decay factor as $1\mathrm{e}{-2}$ for CUB and Cars, and $1\mathrm{e}{-4}$ for SOP and In-Shop.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. International Conference on Learning Representations (ICLR)*, 2021. 2

[3] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[4] 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, 2

[5] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[6] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[7] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[8] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proc. International Conference on Machine Learning (ICML)*, 2021. 2

[9] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 1, 2

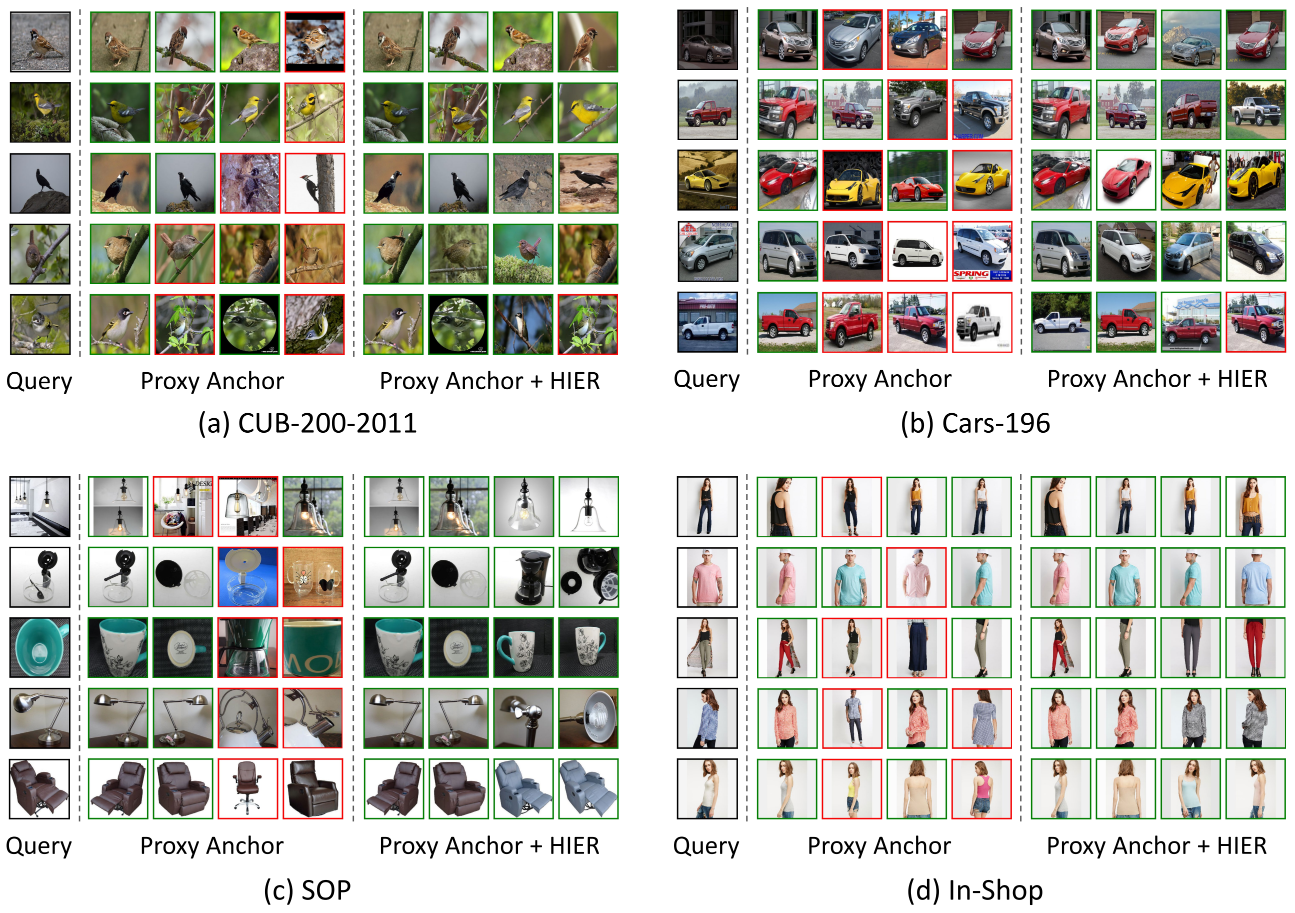

Figure 2. Qualitative results of ours and proxy anchor on the four public benchmark datasets, CUB (a), Cars (b), SOP (c), and In-Shop (d). Queries and the top 4 retrieval results of our method are presented. The true and false matches are colored in green and red, respectively.