Embedding Expansion: Augmentation in Embedding Space for Deep Metric Learning

Supplementary Material

A. Introduction

This supplementary material provides more details of the proposed embedding expansion (EE). First, we compare the proposed method with mixup augmentation techniques and compare the generation methods between EE and mixup. Then, we show the visualization of the embedding space to see the effect of the proposed method. Finally, we investigate the impact of network capacity to see if the proposed method works for different sizes of models.

B. Comparison with MixUp

Early works of mixup [10, 7] propose a data augmentation method by combining two input samples, where the ground truth label of the combined sample is given by the mixture of one-hot labels. By doing so, it improves the generalization of the neural network by regularizing the network to behave linearly in-between training samples. While those mixup methods work in the input space, manifold mixup [8] performs linear combinations of hidden representations of training samples in the representation space. It also improves the generalization of the neural network by perturbing the hidden representations, similar to dropout [6], batch normalization [4], and information bottleneck [1]. In the representative input mixup [10, 7], generating virtual feature-target vectors $(\tilde{x}, \tilde{y})$ is formulated as:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \text{(i)}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \text{(ii)}$$

where (x_i, y_i) and (x_j, y_j) are two feature-target vectors from the training data, $\lambda \sim Beta(\alpha, \alpha)$ for $\alpha \in (0, \infty)$, and $\lambda \in [0, 1]$.

The proposed embedding expansion has similarity with the mixup techniques, where both methods generate virtual feature vectors by combining two original feature vectors for augmentation. However, both methods have major differences in four points: (i) The proposed embedding expansion is for pair-based metric learning losses, whereas the mixup is for softmax loss and its variants. The mixup can not be used with pair-based metric learning losses because most of the pair-based metric learning losses require obvious class labels and can not exploit the mixture of one-hot labels. (ii) The proposed method generates synthetic points in-between a pair from the same class, which are internally dividing points into $n+1$ equal parts, while the mixup uses mixing coefficient λ sampling from beta distribution and generates virtual feature vectors in-between a pair from the different class. (iii) The proposed method exploits the output embedding feature points from a network, while the mixup techniques use feature vectors from the input or hidden representation of a network. (iv) After gener-

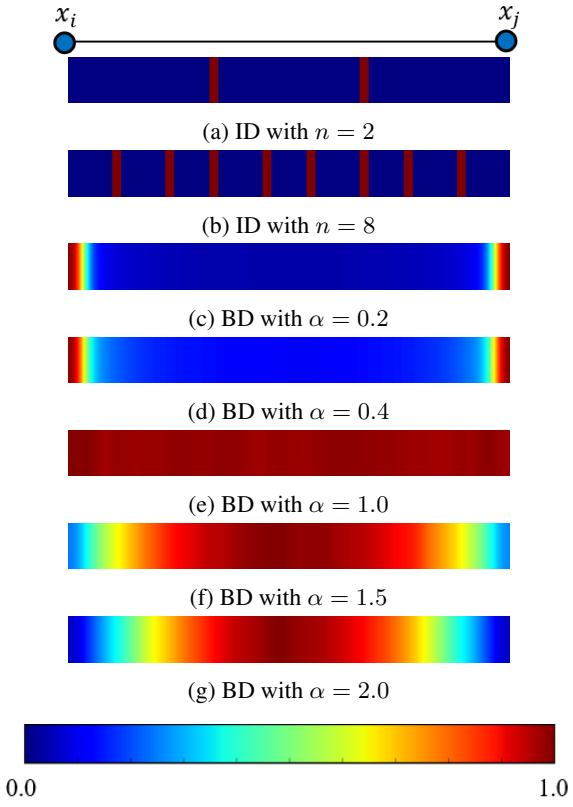


Figure A. A visualization of the locational generation ratio between an original pair (x_i and x_j) from the same class with two different generation methods: the proposed internally dividing (ID) points into $n+1$ equal parts and beta distribution (BD) with α parameter.

Method	n	α	R@1
Baseline	0	-	60.3
EE (ID)	2	-	**71.6**
EE (BD)	2	0.2	66.7
		0.4	59.6
		1.0	56.5
		1.5	55.1
		2.0	53.8
EE (ID)	8	-	**71.2**
EE (BD)	8	0.2	61.5
		0.4	58.1
		1.0	54.3
		1.5	54.5
		2.0	53.9

Table A. Performance (%) comparison among baseline, EE (ID), and EE (BD) with HPHN triplet trained on CARS196. We generate n synthetic points by using the proposed internally dividing points into $n+1$ equal parts for EE (ID) and beta distribution with α parameter for EE (BD).

ating synthetic points, the proposed method performs hard negative pair mining to select the most informative feature points among original and synthetic points.

B.1. Generation with Beta Distribution

The proposed EE generates synthetic points between a pair, which are internally dividing points into $n+1$ equal parts. The synthetic points will be generated on the deterministic locations, as illustrated in Figure Aa and Ab. On the other hand, it is possible to generate synthetic points by using the beta distribution as Equation i of mixup. This will generates synthetic points on the stochastic locations. Smaller α values generates synthetic points nearby the original points (Figure Ac and Ad) and larger α values generates synthetic points around middle of the pair (Figure Af and Ag), while $\alpha = 1.0$ is equal to the uniform distribution (Figure Ae).

To compare these two generation methods, we conduct an experiment by generating n synthetic points with these generation methods and use the same hard negative pair mining as proposed EE. We use triplet loss with hard positive and hard negative (HPHN) mining [3, 9], trained with CARS196 dataset. As shown in Table A, methods of EE (BD) with $\alpha = 0.2$ outperform the baseline model, while larger α decreases the performance. It indicates that the stochastic generation between a pair can be distractive for the training, except for the synthetic points which are close and similar to the original points. Meanwhile, the proposed EE (ID) shows better performance than the baseline and the EE (BD), which shows that the deterministic generation is more stable and effective.

C. Visualization of Embedding Space

In order to see the process of clustering during training, we visualize the embedding space of certain training epochs with the Barnes-Hut t-SNE [5]. We use HPHN triplet loss in Figure B and its combination of EE in Figure C, trained with CARS196 dataset. For each model, we visualize the embedding of the train data with different colors for different classes, and the joint embedding of the train and test data to highlight where the test data is embedded compared to train data.

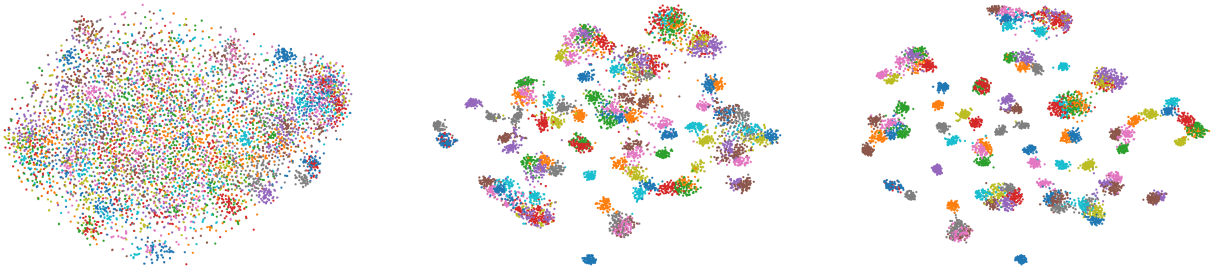
At the beginning of the training, the train and the test set of both triplet and EE + triplet are scattered without forming any discriminative clusters (Figure Ba, Bd, Ca, and Cd). In the middle of the training at 1000 epoch, the train set of EE + triplet starts having clusters (Figure Cb) and the test set are less scattered than the 10 epoch (Figure Ce), while the train set of triplet also starts having clusters with less inter-class variation than EE + triplet (Figure Bb). At the end of the training at 3000 epoch, the train set of EE + triplet has more discriminative clusters with larger inter-class variation, compared to the triplet embedding (Figure Bc and Cc). The test set of EE + triplet are less spread out and forming some clusters compared to the triplet embedding (Figure Cf and Bf). Overall, the combination of triplet loss and the proposed method has shown a better clustering ability than the sole triplet loss. Entire visualization of the training process can be found in the supplementary video[1].

D. Impact of Network Capacity

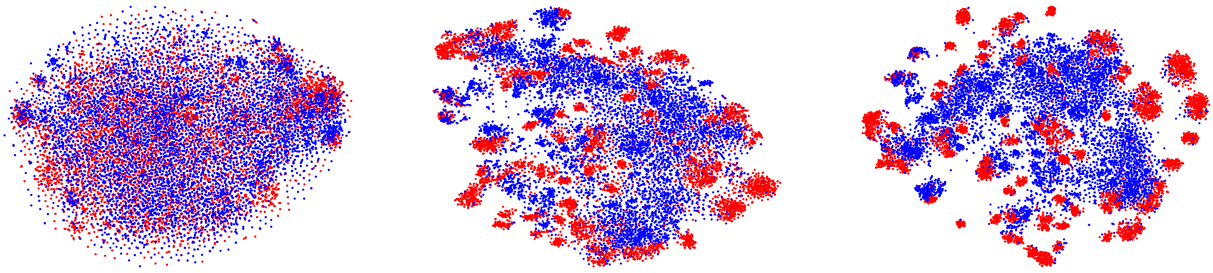
In order to see the impact of network capacity on the proposed method, we conduct an experiment by differentiating the network capacity and the number of synthetic points. We used one of the most generally used ResNet50_v1 [2] and its smaller capacity variants (ResNet18_v1 and ResNet34_v1). Moreover, we compare the proposed method with the hard triplet generation (HTG) [11] method on the same network capacity of ResNet18_v1. Throughout the experiment, we use HPHN triplet and its combination with the proposed method on CUB200-2011 (CUB200), CARS196, and stanford online products (SOP) datasets.

As shown in the Table B, the proposed method achieves around 1% to 3% of performance boost for every network and dataset. We observe that there are the best number of synthetic points n for each dataset, such as $n = 8$ for CUB200, $n = 4$ for CARS196, and $n = 2$ for SOP. In comparison with HTG, even though it uses a combination of no-bias softmax and triplet loss, and a generative adversarial network for sample generation, the proposed method outperforms for every dataset.

[1] https://youtu.be/5msMSXyQZ5U

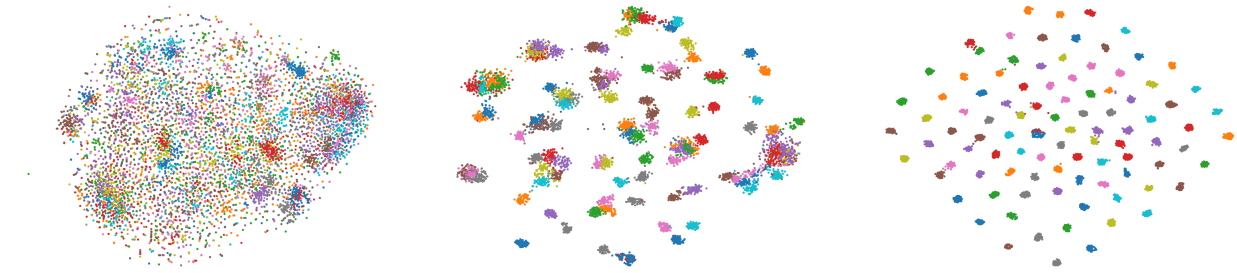


(a) Train, 10 epoch (b) Train, 1000 epoch (c) Train, 3000 epoch

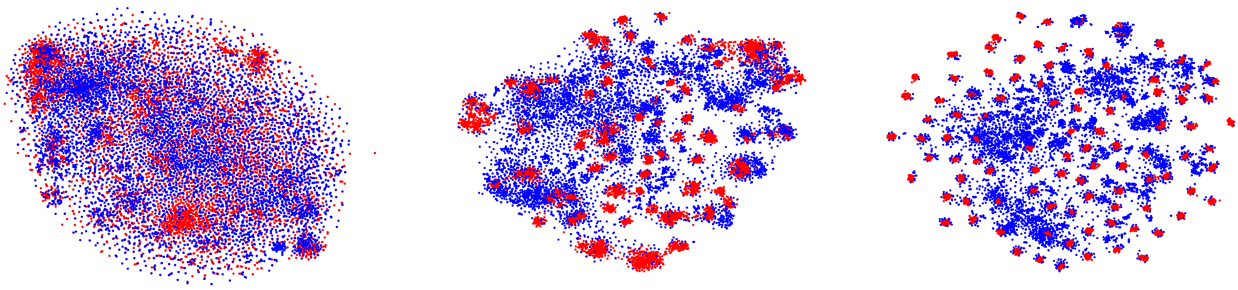


(d) Train + test, 10 epoch (e) Train + test, 1000 epoch (f) Train + test, 3000 epoch

Figure B. A t-SNE visualization of triplet loss with CARS196 dataset. (a), (b), and (c) are the embedding of the train data, while (d), (e), and (f) are the joint embedding of the train (red) and the test (blue) data at each epoch.



(a) Train, 10 epoch (b) Train, 1000 epoch (c) Train, 3000 epoch



(d) Train + test, 10 epoch (e) Train + test, 1000 epoch (f) Train + test, 3000 epoch

Figure C. A t-SNE visualization of EE + triplet loss with CARS196 dataset. (a), (b), and (c) are the embedding of the train data, while (d), (e), and (f) are the joint embedding of the train (red) and the test (blue) data at each epoch.

Method	Network	n	CUB200				CARS196				SOP		
			R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@100
HTG (Soft.+Tri.)†	ResNet18_v1*	-	59.5	71.8	81.3	88.2	76.5	84.7	90.4	94	-	-	-
EE + Triplet‡	ResNet18_v1	0	58.6	70.7	80.6	88.1	83.5	90.3	94.6	97.1	74.7	88.5	95.2
		2	59.3	70.7	80.8	88.1	84.5	90.8	94.5	97.0	**76.9**	**89.4**	**95.3**
		4	59.6	70.0	79.8	87.4	84.5	90.8	94.7	97.1	76.8	89.2	95.2
		8	**60.2**	**71.9**	**81.5**	**88.4**	**85.4**	**91.4**	**94.9**	**97.2**	76.1	88.8	94.9
	ResNet34_v1	0	60.3	72.9	82.4	89.3	84.9	90.1	94.6	97.0	75.6	89.1	95.5
		2	61.1	73.3	83.0	89.0	85.0	91.2	95.0	97.1	**78.2**	**90.3**	**95.8**
		4	61.9	73.7	83.2	89.4	**85.5**	**91.6**	**95.4**	**97.3**	77.8	89.7	95.5
		8	**62.7**	**74.7**	**83.9**	**89.8**	85.1	91.1	94.5	96.9	77.9	90.0	95.7
	ResNet50_v1	0	63.0	73.9	83.1	89.4	87.3	93.0	96.1	98.0	81.2	92.0	96.6
		2	63.5	74.9	84.3	89.7	88.2	93.2	96.2	97.9	**82.5**	**92.7**	**96.9**
		4	63.7	75.0	83.5	89.8	**88.4**	**93.6**	**96.3**	**98.1**	82.3	92.4	96.8
		8	**64.6**	**75.4**	**83.9**	**90.9**	88.3	93.3	96.1	98.0	82.0	92.4	96.8

Table B. Retrieval performance (%) of different network capacity, where $n = 0$ are baseline models. † denotes HTG method with no-bias softmax loss and triplet loss, ‡ denotes the HPHN triplet, and * is specifically modified ResNet18_v1.

References

[1] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016. i

[2] 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. ii

[3] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. ii

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. i

[5] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. ii

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. i

[7] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5486–5494, 2018. i

[8] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Aaron Courville, Ioannis Mitliagkas, and Yoshua Bengio. Manifold mixup: Learning better representations by interpolating hidden states. 2018. i

[9] Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. *arXiv preprint arXiv:1904.04370*, 2019. ii

[10] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. i

[11] Yiru Zhao, Zhongming Jin, Guo-jun Qi, Hongtao Lu, and Xian-sheng Hua. An adversarial approach to hard triplet generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 501–517, 2018. ii