Cross-Domain Ensemble Distillation for Domain Generalization

Kyungmoon Lee[1,2] Sungyeon Kim[2] Suha Kwak[2]

[1]NALBI Inc, Seoul, Korea [2]POSTECH, Pohang, Korea

This supplementary material presents additional experimental results, further analyses, and implementation details, all of which are omitted from the main paper due to the space limit. In Sec. A, additional experimental results of our method are provided. Sec. B and Sec. C present further analyses of the proposed components, respectively. Lastly, implementation details are in Sec D.

A Additional experimental results

A.1 Comparison to existing self-KD methods.

To further assess the impact of the ensemble distillation, we also compare XDED with state-of-the-art self-knowledge distillation (self-KD) methods in spite of sharing a similar methodology. Furthermore considering that our method does not explicitly depend on domain labels, we investigate the impact of XDED on the standard benchmark datasets for the image recognition task, which assume the independent and identically distributed (i.i.d.) condition. As shown in Table a1, XDED outperforms not only the vanilla method but also other self-KD methods for all datasets. Considering that XDED is not dedicated to the case under the i.i.d. condition whereas other methods are dedicated to the case under the i.i.d. condition, these experimental results support our view that XDED can provide more meaningful supervisory signals thanks to the use of ensemble which encodes the complementary knowledge from the data with the same label.

A.2 Results on input deformations

To further assess the quality of the robustness to unseen domains, we measure the changes in test accuracy as the domain gap becomes larger. We simulate larger domain gaps by increasing the degree of input deformations. Specifically, we applied multiple augmentations defined in RandAugment [2] to images of the unseen target domain (*i.e.*, Cartoon of PACS), and gradually increased the number of deformations[1]. Similar to the result of table 9 in our main paper, Fig. a1 shows that our method outperforms baseline methods against pixel-level deformations on unseen domains.

[1] For a fair comparison, we use the same operations for each setting.

Table a1: Accuracy (%) on general image recognition benchmarks.

Methods	CIFAR-10	CIFAR-100
ResNet-18	94.6	75.2
CS-KD [11]	94.6	78.0
BYOT [12]	95.1	78.6
XDED	**95.3**	**78.7**

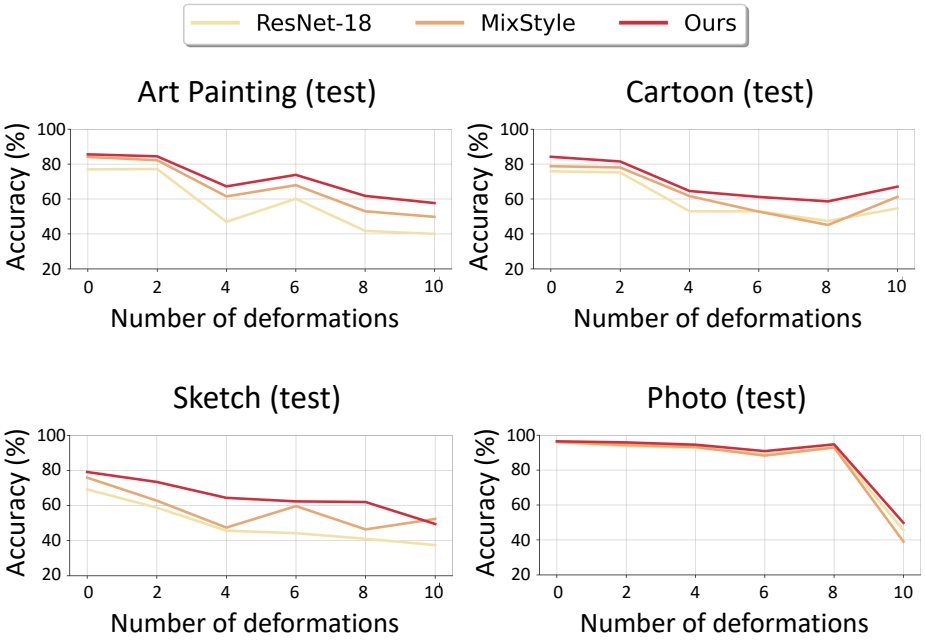

Fig. a1: Test accuracy (%) on target domains with input deformations. All models are trained under the multi-source domain generalization setting.

A.3 Learning acceleration

Our method also enables faster convergence of the learning process. In generalization on PACS, our effectiveness of learning acceleration is demonstrated. As shown in Fig. a2, our method reaches highest performance with less epochs (*i.e.*, 24 epochs) compared to both vanilla method (40 epochs) and MixStyle [14] (39 epochs). Although the vanilla method requires many iterations to address the domain gaps between source domains, our method accelerates the learning process via promoting flat minima which produce lower errors in not only source domains but also the target domain. Since MixStyle [14], as an augmentation method, synthesizes novel styles via mixing statistics at the feature level, it is inherently limited to requiring as many epochs as possible to address many augmented styles for its best performance. Also, in the context of the actual training

Table a2: Performance boosts over training complexity.

Methods	Millisec / iter	Accuracy boost (%)
ResNet-18	34	-
MixStyle [14]	35	4.2
Ours	37	6.9

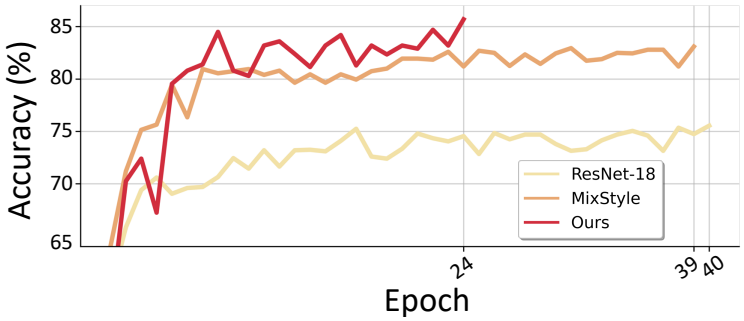

Fig. a2: Multi-source domain generalization accuracy (%) versus the number of epochs on the unseen target domain, Art Painting of PACS. Note that all models were trained on the rest three domains of PACS (*i.e.*, Cartoon, Sketch, and Photo) and all methods were trained with the same hyperparameter setting (*e.g.*, batch size and learning rate) on a single RTX 2080 GPU.

time in total, compared to MixStyle, we remark that our method significantly reduces the training complexity. In detail, our method increases training time per iteration by 5% but requires only 60% of training epochs and achieves about 1.6 times more performance boost averaged over every target domain of the PACS dataset. The results are summarized in Table a2.

A.4 Impact of hyperparameters

We empirically investigated the effect of the two hyperparameters of XDED: λ and τ. We examine accuracy of multi-source domain generalization on Sketch of the PACS dataset by varying the values of $\lambda \in \{0.5, 1.0, 2.5, 5.0, 10.0\}$ and $\tau \in \{1.0, 2.0, 4.0, 6.0, 8.0\}$. The results are summarized in Fig. a3. First, the performance boost is the highest when τ is increased from 1.0 to 2.0. However, if τ exceeds 2.0, the performance boost is somewhat reduced. Next, λ also affects the performance. The performance boost commonly increases with the value of λ. However, the degree of boost is also reduced when τ is 1.0. To sum it up, the accuracy of our method is generally high and stable when λ and τ are greater than 5.0 and 2.0, and we argue that this result supports our view that our method does not strongly depend on the specific hyperparameter setting.

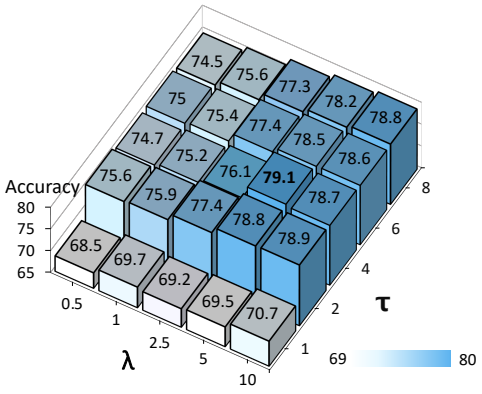

Fig. a3: Multi-source domain generalization accuracy (%) versus hyperparameters λ and τ on Sketch of the PACS dataset.

Table a3: Generalization results on the target domain (Clipart of OfficeHome) and required overhead during training.

Methods	Vanilla	SAM [3]	XDED
Accuracy (%) (↑)	49.4	<u>51.9</u>	**55.2**
Time / iter (ms) (↓)	**32**	72	<u>33</u>

B Additional analysis results on XDED

In this section, we provide further analyses of XDED. Also, we provide additional empirical results on the flatness of local minima (Fig. a5).

B.1 Convergence into flat minima at a low cost

Since there are several ways for convergence into flat minima, we believe that the required overhead to achieve that goal should be considered as another criterion of valuation for a practical usage. Therefore, according to this point of view, we compare XDED with SAM [3] which has been proposed to optimize loss sharpness more directly. As shown in Table. a3, XDED shows superior robustness on the unseen domain over SAM. Furthermore, XDED requires negligible overhead while SAM requires the training time more than 2 times longer than the vanilla method and XDED. This is because SAM must additionally compute gradient approximation for the loss sharpness to be optimized, which exposes a trade-off between training complexity and the optimization of the loss sharpness. On the other hand, SAM's performance is inferior to that of XDED, since it directly promotes a flat minima but cannot learn domain-invariant features, unlike our method. To sum it up, we argue that XDED offers more appealing option for

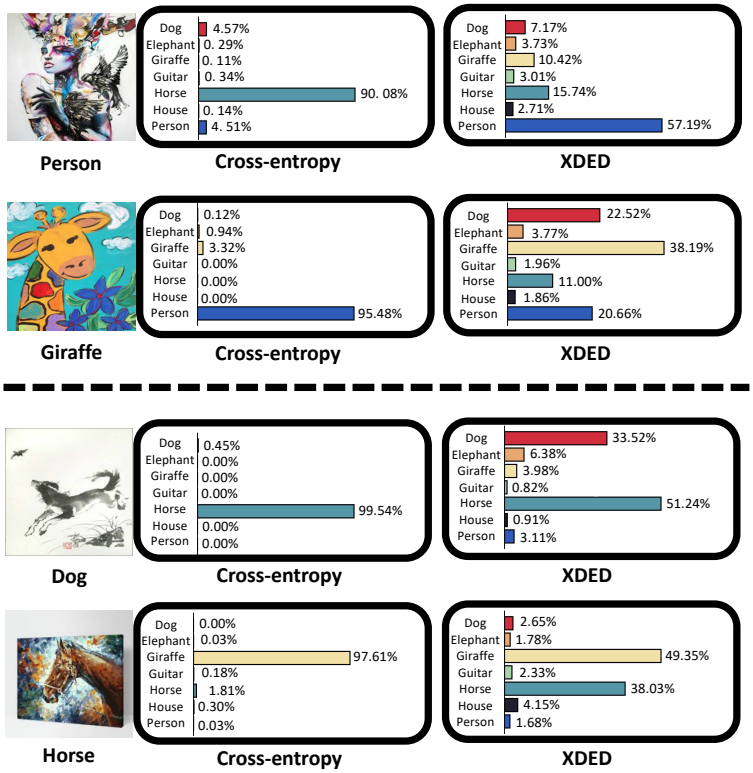

Fig. a4: Prediction scores for misclassified samples from the vanilla method (*i.e.*, Cross-entropy). **Top**: Well-classified results from XDED, **Bottom**: Misclassified results from XDED.

domain generalization in the context of both performance improvement and required overhead.

B.2 Ensemble matters for generalization ability

To examine how XDED affects the predictions of the learned model in detail, we investigate the softmax scores from the model which is trained on the source domains (*i.e.*, Cartoon, Sketch and Photo of PACS) and evaluated on the target domain (*i.e.*, Art Painting). Specifically, for the misclassified samples from the vanilla method, we assess the differences in the softmax scores from the vanilla method and XDED. First, as shown in Fig. a4 (Top), XDED can lead to accurate predictions on the samples which the vanilla method confidently misclassifies. For the giraffe sample, the model learned with XDED not only predicts the ground truth successfully but also puts relatively high probability on the similar class (*i.e.*, Dog) while lowering the probabilities of the irrelevant classes (*i.e.*, Guitar and House). Similarly, even in its failure cases, XDED encourages the model

Table a4: Comparison with other ensemble methods which are dedicated to domain generalization. Extra θ and Domain ID denote whether each method requires additional modules and domain labels are explicitly bound to be given, respectively. Accuracy denotes the average accuracy of each method over PACS and Office-Home.

Methods	Extra θ	Domain ID	Accuracy
DAEL [13]	✓	✓	74.7
DSON [10]	✓	✓	74.0
Ours	✗	✗	76.9

to predict with consideration of inter-class relations (*i.e.*, increasing scores for similar classes while suppressing those for irrelevant classes) as shown in Fig. a4 (Bottom). By exploiting the ensemble incorporating information of multiple domains, XDED encourages the model to consider richer inter-class relations which may be exploited on the target domain and lead to meaningful predictions even in failure cases. Note that UniStyle is excluded from the results discussed above.

B.3 Comparison with other ensemble methods.

Lastly, we compare XDED with other domain generalization methods which adopt ensemble techniques. The comparison analysis is summarized in Table a4. We remark that the proposed method does not require any additional module and is also free from domain labels (Note that our method, thus, seamlessly extends to the single-source domain generalization setting, which is discussed in Sec.4.1 of our main paper.). Nevertheless, our method clearly outperforms other methods in the context of generalization ability. In contrast, DAEL [13] and DSON [10] show inferior results even though they explicitly exploit domain labels and multiple domain-specific modules (*i.e.*, classifiers and BN layers [5], respectively.). DAEL is the most similar to our method but clearly different in the sense that DAEL exploits the predictions of an expert as supervisory signals for non-experts on the same instance, but our method constructs the ensemble using multiple instances and exploits it as supervision to model itself. In other words, by learning domain-invariant information through the ensemble of meaningful knowledge contained in different domains, better performance could be achieved even with limited resources.

C Additional analysis results on UniStyle

C.1 Where to apply UniStyle

We remark that UniStyle can be applied after arbitrary intermediate layers of the backbone (*i.e.*, ResNet [4]) as a plug-and-play module. Therefore, to investigate the impact of where UniStyle is applied, we evaluate the generalization performance in image classification and person re-ID while varying the locations

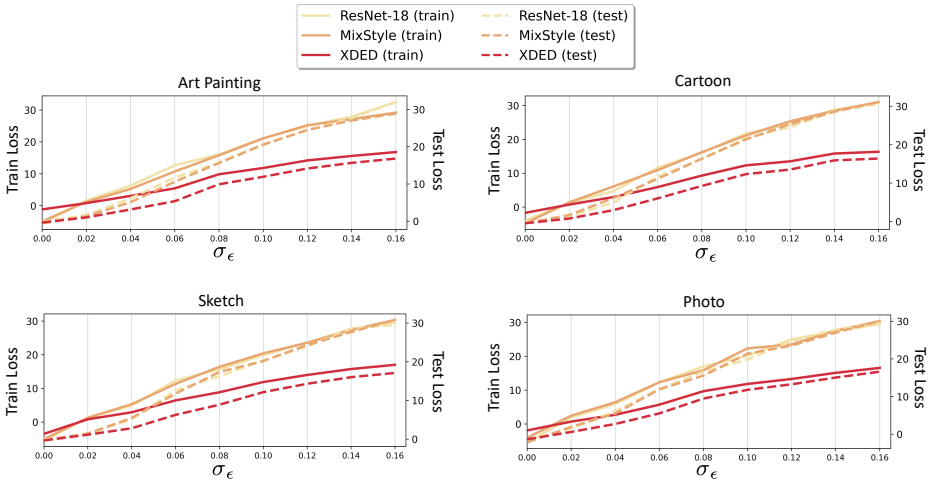

Fig. a5: Train/Test losses versus the weight perturbation while varying the standard deviation of the added Gaussian noise. Note that train losses are calculated over the remaining three source domains for each target domain and the loss values are log-scaled.

Table a5: Ablation study on where to apply UniStyle in the ResNet architecture.

(a) Image classification on PACS

Model	Accuracy (%)
ResNet-18	79.5
+ MixStyle (RES1)	80.1
+ UniStyle (RES1)	81.5
+ MixStyle (RES12)	81.6
+ UniStyle (RES12)	**82.9**
+ MixStyle (RES123)	82.8
+ UniStyle (RES123)	82.4
+ MixStyle (RES1234)	75.6
+ UniStyle (RES1234)	12.8

(b) Person re-ID from Market1501→Duke

Model	mAP (%)
ResNet-50	19.3
+ MixStyle (RES1)	22.6
+ UniStyle (RES1)	23.3
+ MixStyle (RES12)	23.8
+ UniStyle (RES12)	24.1
+ MixStyle (RES123)	22.0
+ UniStyle (RES123)	**26.2**
+ MixStyle (RES1234)	10.2
+ UniStyle (RES1234)	0.2

of where the operation is applied. For a baseline, MixStyle [14] is adopted and compared with our UniStyle since they share the commonality of being applied to multiple intermediate feature maps. For brevity, let RES# denote the indexes of residual blocks where the specified operation is applied (*e.g.*, RES12 means the operation is applied after both the first and second residual blocks). As shown in Table. a5, we observe that UniStyle and MixStyle have a similar trend in both tasks. First, since early layers are known to capture low-level features such as texture or edge information, it is pertinent for both UniStyle and MixStyle to be applied after early layers to remove style bias and synthesize novel styles,

respectively. However, UniStyle achieves the best performances on both tasks, which indicates the importance of removing and unifying style bias rather than augmenting novel styles. Next, on the contrary, both operations lead critical performance drop when incorporated with the last layer, RES4, since late layers are known to address semantic information (*i.e.*, their statistics would be highly correlated with target labels). In detail, MixStyle perturbs the statistics by interpolating those of two different instances that may have different labels, whereas UniStyle normalizes the feature map of all samples regardless of their labels, resulting in a more critical performance drop. Note that XDED is excluded in results of Table. a5.

C.2 Comparison with other methods

Even though UniStyle is not the first that addresses feature-level statistics (*i.e.*, style information), we take notice of its appealing trade-off between performance boost and training complexity and compare it with other related methods. The results are summarized in Table a6. First, UniStyle outperforms SagNet [9] and MixStyle [14], all of which are dedicated to domain generalization. This indicates that directly removing and unifying the style bias is more effective for generalization ability than augmenting feature statistics or learning style-agnostic features. Next, although BIN [8] and SRM [6] adaptively recalibrate feature statistics and achieve state-of-the-art performance in image recognition tasks, they show inferior results on unseen domains. Last but not least, Deep Whitening Transformation (DWT) [1] which is the most similar to UniStyle shows the second-best performance. DWT has been proposed to encourage the feature extractor to naturally encode the whitened feature by enforcing consistency regularization between the covariance matrix of the feature and identity matrix. Nevertheless, as a constraint that is too strong, DWT weakens feature discrimination and excessively throws away content information, resulting in performance that lags behind UniStyle.

D Implementation details

For the task of domain generalization in image classification, we train the models using the sgd optimizer with the cosine learning decay [7] and initial learning rate of 10^{-3}. They are learned for 100 epochs. For batch construction, we use the batch size of 64, and sample 16 instances per class for the proposed XDED. For the task of person re-ID, we also train the models using the sgd optimizer with initial learning rate of 0.05.

Table a6: Comparison between UniStyle and other related methods. Extra θ and Extra $\mathcal{L}$ denote whether each method requires additional modules and loss function, respectively, which both increase training complexity and resources. Accuracy denotes the multi-source domain generalization accuracy (%) of each method on the target domain, Cartoon of the PACS dataset.

Methods	Extra θ	Extra $\mathcal{L}$	Accuracy
ResNet-18	✗	✗	75.9
Domain Generalization			
SagNet [9]	✓	✓	77.6
MixStyle [14]	✗	✗	78.8
Style Transfer			
BIN [8]	✓	✗	78.7
SRM [6]	✓	✗	78.9
DWT [1]	✗	✓	<u>79.0</u>
UniStyle	✗	✗	**79.2**

References

1. Cho, W., Choi, S., Park, D.K., Shin, I., Choo, J.: Image-to-image translation via group-wise deep whitening-and-coloring transformation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 8, 9
2. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (2020) 1
3. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: Proc. International Conference on Learning Representations (ICLR) (2021) 4
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 6
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. International Conference on Machine Learning (ICML) (2015) 6
6. Lee, H., Kim, H.E., Nam, H.: Srm: A style-based recalibration module for convolutional neural networks. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 8, 9
7. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 8
8. Nam, H., Kim, H.E.: Batch-instance normalization for adaptively style-invariant neural networks. In: Proc. Neural Information Processing Systems (NeurIPS) (2018) 8, 9
9. Nam, H., Lee, H., Park, J., Yoon, W., Yoo, D.: Reducing domain gap by reducing style bias. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 8, 9
10. Seo, S., Suh, Y., Kim, D., Kim, G., Han, J., Han, B.: Learning to optimize domain specific normalization for domain generalization. In: Proc. European Conference on Computer Vision (ECCV) (2020) 6
11. Yun, S., Park, J., Lee, K., Shin, J.: Regularizing class-wise predictions via self-knowledge distillation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 2
12. Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., Ma, K.: Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: Proc. IEEE International Conference on Computer Vision (ICCV) (2019) 2
13. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain adaptive ensemble learning. IEEE Transactions on Image Processing (TIP) (2021) 6
14. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain generalization with mixstyle. In: Proc. International Conference on Learning Representations (ICLR) (2021) 2, 3, 7, 8, 9