1. Experimental Details

We designed different task network architectures and training strategies for different data sets.

In the Digits dataset, we use a simple task model according to [10]. The network architecture is shown in Tab. 1. We applied the ReLU activation layer for all the convolution layers and fully connected layers. The fc5 and fc6 are two output layers that are connected to fc4. The softmax layer is applied for fc5. And the output vector of fc6 is normalized to lie on the unit hypersphere.

Layer	ksize	stride	pad	# filters	Data shape
input					(3, 32, 32)
conv1	5	1	0	64	(64,28,28)
pool1	2	2			(64,14,14)
conv2	5	1	0	128	(128,10,10)
pool2	2	2			(128,5,5)
fc3					(1024,)
fc4					(1024,)
fc5(out1)					(10,)
fc6(out2)					(128,)

Table 1. Configuration of the task network architecture used in Digits dataset.

In CIFAR10-C[3] dataset, the task model is a 16 layers Wide Residual Network(WRN) [15] and the width is 4. The first layer is a convolution layer with 16 kernels and the kernel size is 3. Then there are 3 groups of convolution layers. Each group consists of 2 residual blocks and each block consists of 2 convolution layers. The number of channels in the 3 groups is 64, 128 and 256 respectively. An average pooling layer with the 8×8 kernel is used after the third group. Finally, we apply two fully connected layers to get the prediction results and the projection vectors. All convolution layers are followed by the ReLU activation layers and the Batch Normalize layers.

In SYNTHIA[11], we use the FCN [8] as the task model. ResNet-50 is applied as the backbone [1]. First, the features are extracted by the ResNet-50. Then the coarse segmentation result is predicted with a 1×1 convolutional layer. A transposed convolution layer is used to upsample the coarse segmentation result to the input image size. And a fully connected layer is applied after the ResNet-50 to get projection vectors.

The domain expansion network G in all the experiments is similar. G can be a variety of structures depending on related downstream tasks, such as AutoEncoder [6], HRNet [12], spatial transform network(STN) [4] or a combination of these networks. In our experiment, we mainly use the Autoencoder with AdaIN [5] as the generator. G consists of the encoder G_E, the AdaIN and the decoder G_D. First, the images go through the encoder to get the features. Then the mean and variance of the features are randomly adjusted by the AdaIN layer. Finally, a new image is generated through the decoder.

[1]https://pytorch.org/vision/stable/models.html?highlight=fcn_resnet50 #torchvision.models.segmentation.fcn_resnet50

2. Comparison on CIFAR10-C

We train all the models on the CIFAR10 train set, validate the models on the CIFAR10 test set, and evaluate the models on the CIFAR10-C. We show the experimental results across different types of corruptions with the 5th level severity in Tab. 2. Our approach has higher average accuracy than other approaches. In some corruption types, the RandAugment[2] approach performs better than us. However, it is important to note that there is no manual data augmentation in our approach, and our approach can be used together with RandAugment.

3. Visualization of the feature space

The main idea of PDEN is to improve the generalization of the model in the unseen domain by learning the domain invariant feature representation. Fig.1 illustrates the difference in feature space between PDEN and the baseline models. For better visualization, we keep all the model output 2-d features. Rows 1 and 2 correspond to the baseline model and the PDEN. Columns 1, 2 and 3 correspond to different dataset.

Column 1: The MNIST distribution in feature space of the baseline model and the PDEN is different. For the baseline model, each class is distributed in the feature space in a spindle shape. All samples are far from the decision boundary. For PDEN, the distribution of each class is fan-shaped in the feature space. In contrast, more samples are close to the decision boundary in the feature space of PDEN.

Column 2: These two sub-figure represent the distribution of samples(generated by PDEN) in the feature space of the baseline and PDEN models. For the baseline model, the generated samples of different categories are mixed with each other to such an extent that they cannot be distinguished. This indicates that the samples generated by PDEN are hard samples for the baseline model. For PDEN, the hard samples can be perfectly distinguished.

Column 3: We take MNIST_M as the target domain. In the feature space of the baseline model, most samples from MNIST_M are mixed with each other. In the feature space of PDEN, fewer samples are mixed with each other. This illustrate that PDEN can focus on domain-invariant representations, so PDEN achieves better generalization on unseen domains.

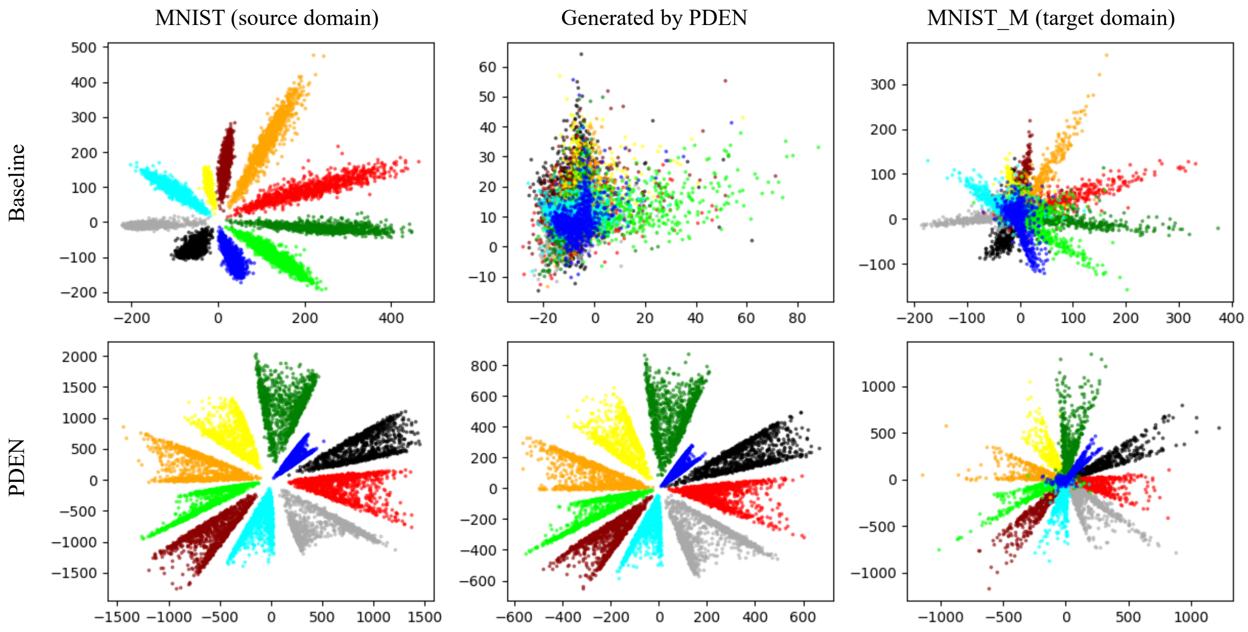

Figure 1. Visualization of different domains in the feature space. **Row 1**: the feature space of the baseline model. **Row 2**: the feature space of the PDEN model. **Column 1**: the distribution of the MNIST (source domain). **Column 2**: the distribution of the samples generated by the generator in PDEN. **Column3**: the distributions of MNSIT_M (target domain). Different colors correspond to different classes.

	Weather			Blur					Noise			
	Fog	Snow	Frost	Zoom	Defocus	Glass	Gaussian	Motion	Speckle	Shot	Impulse	Gaussian
ERM[7]	65.92	74.36	61.57	59.97	53.71	49.44	30.74	63.81	41.31	35.41	25.65	29.01
CCSA[9]	66.94	74.55	61.49	61.96	56.11	48.46	32.22	64.73	40.12	33.79	24.56	27.85
d-SNE[14]	65.99	75.46	62.25	58.47	53.71	50.48	33.06	63.0	45.30	39.93	27.95	34.02
GUD[13]	68.29	76.75	69.94	62.95	56.41	53.45	38.33	63.93	38.45	36.87	22.26	32.43
MADA[10]	69.36	80.59	76.66	68.04	61.18	61.59	47.34	64.23	60.88	60.58	45.18	56.88
AA[1]	84.61	81.04	72.32	83.94	84.38	52.29	76.26	77.36	52.14	45.40	52.54	36.77
RA[2]	**85.99**	80.13	74.97	**88.60**	**89.33**	57.70	**87.88**	**79.34**	60.50	56.03	55.64	49.68
PDEN	69.64	**81.81**	**84.50**	83.73	82.15	**60.13**	79.31	76.72	**79.31**	**81.28**	**66.79**	**81.06**

	Digital							
	Jpeg	Pixelate	Spatter	Elastic	Brightness	Saturate	Contrast	Avg.
ERM[7]	69.90	41.07	75.36	72.40	91.25	89.09	36.87	56.15
CCSA[9]	69.68	40.94	77.91	72.36	91.00	89.42	35.83	56.31
d-SNE[14]	70.20	38.46	73.40	73.33	90.90	89.27	36.28	56.96
GUD[13]	74.22	53.34	80.27	74.64	89.91	82.91	31.55	58.26
MADA[10]	77.14	52.25	80.62	75.61	90.78	87.62	29.71	65.59
AA[1]	73.65	36.12	89.13	73.79	**94.54**	**93.79**	91.31	71.13
RA[2]	74.92	37.36	**90.42**	**75.96**	93.90	93.17	**92.06**	74.93
PDEN	**85.24**	**70.82**	79.38	75.05	90.98	88.44	55.59	**77.47**

Table 2. Full version of Tab. 3 in the main paper. The experimental result on CIFAR10-C. The model is trained on the clean data of CIFAR10 and evaluate on CIFAR10-C. We compared the accuracy of 19 types of corruption(only 12 corruptions are shown in the table) at level 5(the severest) in different methods.

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 2

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 1, 2

[3] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 1

[4] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 1

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[7] Vladimir Koltchinskii. *Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems: Ecole d'Eté de Probabilités de Saint-Flour XXXVIII-2008*, volume 2033. Springer Science & Business Media, 2011. 2

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[9] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017. 2

[10] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. 2020. 1, 2

[11] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[12] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5693–5703, 2019. 1

[13] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5334–5344. Curran Associates, Inc., 2018. 2

[14] Xiang Xu, Xiong Zhou, Ragav Venkatesan, Gurumurthy Swaminathan, and Orchid Majumder. d-sne: Domain adaptation using stochastic neighborhood embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2497–2506, 2019. 2

[15] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1