Uncertainty-guided Model Generalization to Unseen Domains Supplementary Materials

Fengchun Qiao
University of Delaware
fengchun@udel.edu

Xi Peng
University of Delaware
xipeng@udel.edu

1. Architecture Design and Setup

We provide more experimental details on the five datasets: *Digits* [20], *CIFAR-10-C* [6], *SYTHIA* [16], *Amazon Reviews* [1], and *Google Commands* [21]. In learnable label mixup, we use Gaussian parameters of feature perturbations from the first layer. We choose specific backbone models, and design different auxiliary models as well as training strategies according to characteristics of each dataset.

In *Digits* [20], the backbone model is *conv-pool-conv-pool-fc-fc-softmax*. There are two 5×5 convolutional layers with 64 and 128 channels respectively. Each convolutional layer is followed by a max pooling layer with the size of 2×2. The size of the two Fully-Connected (FC) layers is 1024 and the size of the softmax layer is 10. We inject perturbations to latent features of the two convolutional layers. The detailed architecture is presented in Fig. 1 (a). We employ Adam [8] for optimization with batch size of 32. We train for total 10K iterations with learning rate of 10^{-4}.

In *CIFAR-10-C* [6], we evaluate our method on two backbones: AllConvNet (AllConv) [18] and Wide Residual Network (WRN) [22] with 40 layers and the width of 2. In AllConv [18], the model starts with three 3×3 convolutional layers with 96 channels. Each layer is followed by batch normalization (BN) [7] and GELU. They convert the original image with three channels to feature maps of 96 channels. Then, the features go though three 3×3 convolutional layers with 192 channels. After that, the features are fed into two 1×1 convolutional layers with 192 channels and an average pooling layer with the size of 8×8. Finally, a softmax layer with the size of 10 is used for classification. In WRN [22]. The first layer is a 3×3 convolutional layer. It converts the original image with three channels to feature maps of 16 channels. Then the features go through three blocks of 3×3 convolutional layers. Each block consists of six basic blocks and each basic block is composed of two convolutional layers with the same number of channels. And their channels are {32, 64, 128} respectively. Each layer is followed by batch normalization (BN) [7]. An average pooling layer with the size of 8×8 is appended to the output of the third block. Finally, a softmax layer with the size of 10 is used for prediction. In both AllConv [18] and WRN [22], we only inject perturbations to the latent features of the first convolutional layer. We also tried to inject perturbations in the next few layers or blocks, however, we found the performance degraded severely mainly due to its large effect on the semantic feature, *i.e.*, outputs before the activation layer. The detailed architecture with backbone of WRN is shown in Fig. 1 (b). Following the training procedure in [22], we use SGD with Nesterov momentum and set the batch size to 128. The initial learning rate is 0.1 with a linear decay and the number of epochs is 200.

In *SYTHIA* [16], we use FCN-32s [11] with the backbone of ResNet-50 [5]. The model consists of a feature extractor and a classifier. We use ResNet-50 [5] as the feature extractor, which is composed of a 7×7 convolutional layer with 64 channels and four convolutional blocks. The classifier consists of a 3×3 convolutional layer with 512 channels, a 1×1 convolutional layer with 14 channels, and a bilinear layer used to up-sample the coarse outputs to the original size. We use Adam with the learning rate $\alpha = 0.0001$. We set the batch size to 8 and the number of epochs to 50.

In *Amazon Reviews* [1], reviews are assigned binary labels - 0 if the rating of the product is up to 3 stars, and 1 if the rating is 4 or 5 stars. The extracted features are fed into two FC layers with the size of 50. A softmax layer with the size of two is used to classify the sentiment of reviews into "positive" or "negative". All models are trained using Adam [8] optimizer with learning rate of 10^{-4} and batch size of 32 for 1000 iterations. In *Google Commands* [21], the mel-spectrogram features are fed into LeNet [10] as one-channel input. The original image is fed into two 5×5 convolutional layers with the channels of 6 and 16, respectively. Next, the features go through two FC layers with the size of 120 and 84, respectively. Finally, a softmax layer with the size of 30 is leveraged to predict the spoken word. Models are trained using Adam [8] with learning rate 10^{-4} and batch size of 128 for 30 epoches. For the corrupted test sets, the range of "amplitude change" is (0.7, 1.1). The maximum scales of "pitch change", "background noise", and "stretch" are

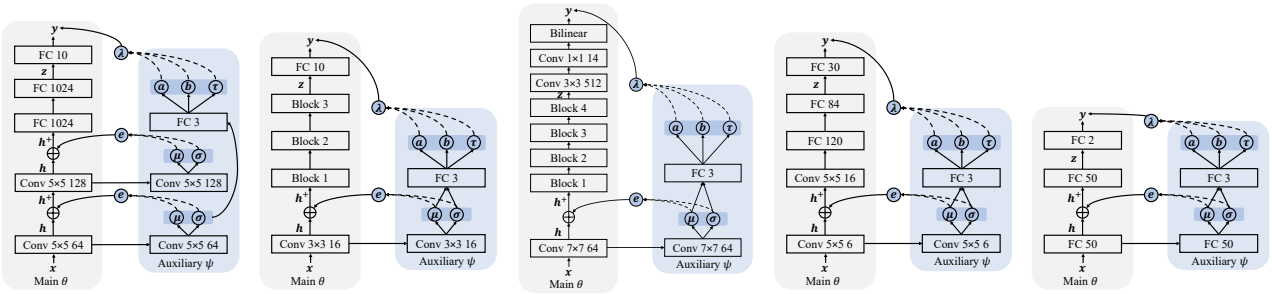

Figure 1: Architectures of main and auxiliary models. **From left to right:** (a) *Digits* [20]; (b) *CIFAR-10-C* [6]; (c) *SYTHIA* [16]; (d) *Google Commands* [21], and (e) *Amazon Reviews* [1].

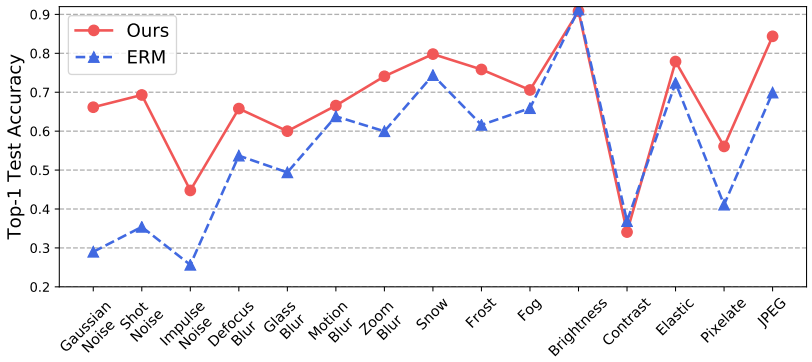

Figure 2: Classification accuracy on fifteen corruptions of *CIFAR-10-C* using the backbone of WRN (40-2). Following Fig. 3, the accuracy of each corruption with the highest level of severity is presented. Our method achieves 20% improvements on corruptions of noise.

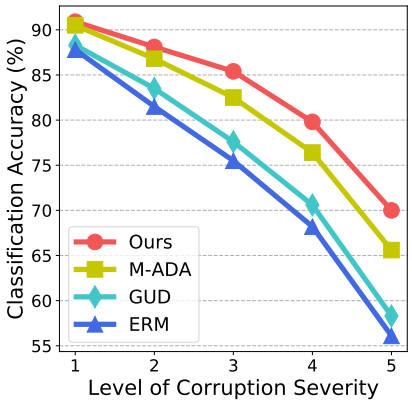

Figure 3: Classification accuracy (%) on five levels of corruption severity. Our method has the smallest degradation under the highest level of corruption severity.

0.2, 0.45, and 0.2, respectively. The maximum shift of "time shift" is 8. In the experiments of *Amazon Reviews* [1] and *Google Commands* [21], feature perturbations are appended to the first layer. The detailed architectures used for *Google Commands* [21] and *Amazon Reviews* [1] are presented in Fig. 1 (c) and (d), respectively.

2. Additional Results

2.1. Image Classification

1) Classification accuracy on *CIFAR-10-C* [6]. We train all models on clean data, *i.e.*, CIFAR-10, and test them on corruption data, *i.e., CIFAR-10-C*. In this case, there are totally 15 unseen testing domains. We compare our method with the other three methods for domain generalization: ERM [9], GUD [20], and M-ADA [15]. The classification results on corruptions across five levels of severity are shown in Fig. 3. As seen, our method outperforms other methods across all levels of corruption severity. Specifically, the gap between M-ADA [15] (previous SOTA) and our method gets larger with the level of severity increasing. Fig. 2 shows more detailed comparison of all corruptions at the highest level of severity. As seen, our method achieves substantial gains across a wide variety of corruptions, with a small drop of performance in only two corruption types: brightness and contrast. Especially, accuracy is significantly improved by 20% on corruptions of noise. Results demonstrate its strong generalization capability on severe corruptions.

Method	$\lvert\mathcal{T}\rvert$	U → M	M → S	S → M	Avg.
DIRT-T [19]		-	54.50	**99.40**	-
SE [3]	All	**98.07**	13.96	99.18	70.40
SBADA [17]		97.60	**61.08**	76.14	78.27
FADA [12]	7	91.50	47.00	87.20	75.23
CCSA [13]	10	95.71	37.63	94.57	75.97
Ours	7	92.97	58.12	89.30	80.13
	10	93.16	59.77	91.67	**81.53**

Table 1: Few-shot domain adaptation accuracy (%) on *MNIST(M), USPS(U), and SVHN(S)*. $\lvert\mathcal{T}\rvert$ denotes the number of target samples (per class) used during model training.

2) Few-shot domain adaptation. We conduct three few-shot domain adaption tasks: *USPS(U)→MNIST(M)*,

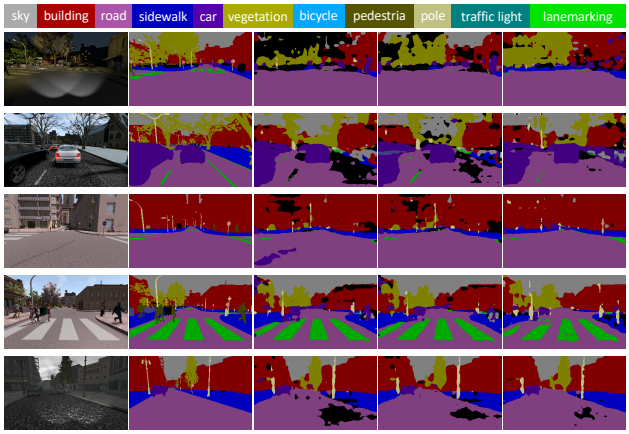

Figure 4: Examples of semantic segmentation on *SYTHIA* [16]. **From left to right:** (a) images from unseen domains; (b) ground truth; (c) results of ERM [9]; (d) results of M-ADA [15]; and (e) results of our method. Best viewed in color and zoom in for details.

MNIST(M)→SVHN(S), and *SVHN(S)→MNIST(M)*. Results of the three tasks are shown in Tab. 1. As seen, the result on the hardest task (*M→S*) is even competitive to that of SBADA [17] which requires all images of the target domain during training. Specifically, our method achieves the best performance on the average of the three tasks. Note that, when the target domain changes, our method only needs to fine-tune the pre-trained model with a few samples within a small number of iterations, while other methods have to train entirely new models.

2.2. Semantic Segmentation

In the experiment on *SYTHIA* [16], *Highway* is the source domain, and *New York-like City* together with *Old European Town* are unseen domains. Visual comparison on *SYTHIA* [16] is shown in Fig. 4. Results demonstrate that our model can better generalize to the changes of locations, weather, and time.

2.3. Text Classification

We train the models on one source domain, and evaluate them on the other three domains. Tab. 2 shows the results of text classification on *Amazon Reviews* [1] . We found that our method outperform previous ones on all the three unseen domains when the source domain is "books" or "kitchen". Specially, our method outperforms ERM [9] by 3.5% on "books → electronics". We observe that there is a little drop in accuracy on "dvd → electronics" and "electronics → dvd". One possible reason is that "electronics" and "dvd" may share a similar distribution. And our method creates large distribution shift, degrading the performance on them.

2.4. Ablation Study

We study the effect of two important hyper-parameters of our model: the number of Monte-Carlo (MC) samples (K) and the coefficient of constraint (β). We report the average accuracy on the four unseen domains (*MNIST-M* [4], *SVHN* [14], *SYN* [4], and *USPS* [2]). We present the classification results under different hyper-parameters in Fig. 5.

1) Number of MC samples (K). The classification accuracy on *Digits* [20] with different K is shown in Fig. 5 (a). We notice that the average accuracy gradually increases from $K = 1$ to $K = 15$ and remains stable when $K = 20$.

2) Coefficient of constraint (β). The constraint is used to make adversarial domain augmentation satisfy the worst-case constraint. Results on *Digits* [20] with different β is presented in Fig. 5 (b). As seen, the accuracy falls dramatically when $\beta = 10$, because large β may severely limit the domain transportation and create domain augmentations similar to the source.

References

[1] Minmin Chen, Zhixiang Xu, Kilian Q Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *International Conference on Machine Learning*, pages 1627–1634, 2012.

[2] John S Denker, WR Gardner, Hans Peter Graf, Donnie Henderson, Richard E Howard, W Hubbard, Lawrence D Jackel, Henry S Baird, and Isabelle Guyon. Neural network recognizer for hand-written zip code digits. In *Annual Conference on Neural Information Processing Systems*, pages 323–331, 1989.

[3] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*, 2018.

[4] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[6] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations*, 2019.

[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, 2015.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *arXiv:1412.6980 [cs.LG]*, 2014.

[9] 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.

	books			dvd			kitchen			electronics		
Method	d	k	e	b	k	e	b	d	e	b	d	k
ERM [9]	78.7	74.6	63.6	78.5	82.1	**75.2**	75.4	76.0	81.2	69.4	**74.8**	83.9
GUD [20]	79.1	75.6	64.7	78.1	82.0	74.6	74.9	76.7	81.6	68.9	74.2	84.4
M-ADA [15]	79.4	76.1	65.3	78.8	82.6	74.3	75.2	77.3	82.3	69.0	73.7	84.8
Ours	**80.2**	**76.8**	**67.1**	**80.1**	**83.5**	75.0	**76.1**	**78.2**	**83.5**	**70.2**	74.5	**85.7**

Table 2: Text classification accuracy (%) on *Amazon Reviews* [1]. The models are trained on only one text domain and evaluated on other unseen text domains. Our method outperforms others in all settings except "$dvd \rightarrow electronics$" and "$electronics \rightarrow dvd$". The possible reason is that "dvd" and "$electronics$" may share a similar distribution while our method creates large distribution shift.

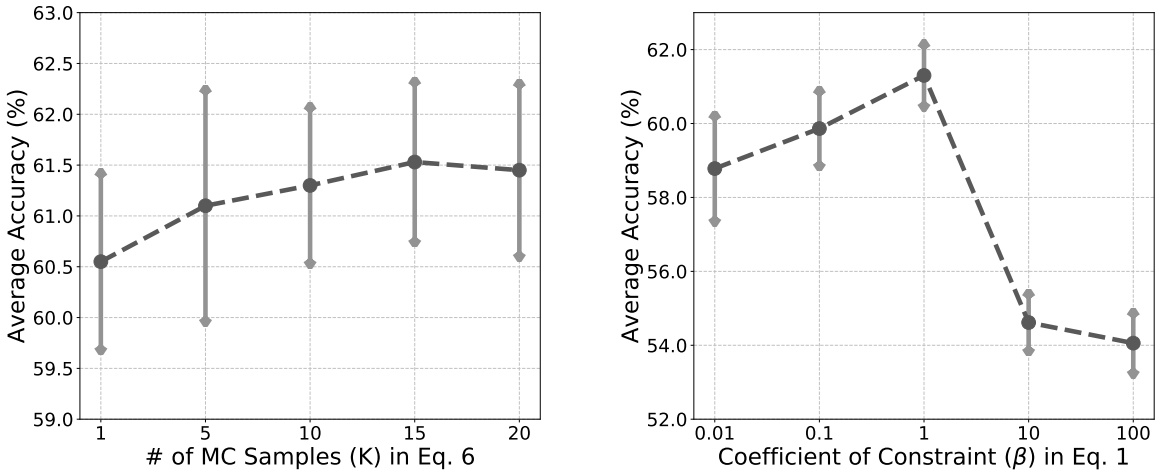

Figure 5: Ablation study on hyper-parameters K and β. The average accuracy on the four unseen domains (*MNIST-M* [4], *SVHN* [14], *SYN* [4], and *USPS* [2]) is presented. We set $K = 15$ and $\beta = 1$ according to the best classification accuracy.

[10] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 1998.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[12] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Annual Conference on Neural Information Processing Systems*, pages 6670–6680, 2017.

[13] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified Deep Supervised Domain Adaptation and Generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5715–5725, 2017.

[14] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[15] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020.

[16] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[17] Paolo Russo, Fabio M Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8099–8108, 2018.

[18] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 901–909, 2016.

[19] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon.

A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[20] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Annual Conference on Neural Information Processing Systems*, pages 5334–5344, 2018.

[21] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[22] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 2016.