SimDE: A Simple Domain Expansion Approach for Single-source Domain Generalization: Supplementary

Qinwei Xu[1] Ruipeng Zhang[1] Yi-Yan Wu[3] Ya Zhang[1,2 ✉] Ning Liu[1] Yanfeng Wang[1,2]

[1] Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China
[2] Shanghai AI Laboratory, Shanghai, China
[3] Communications Research Centre, Ottawa, Ontario, Canada

{qinweixu, zhangrp, ya_zhang, ningliu, wangyanfeng}@sjtu.edu.cn, yiyan.wu@ieee.org

A. Implementation Details

Here we elaborate the implementation details of the network architectures and training configurations of our method on different datasets.

For Digits, we employ the LeNet [4] for the task model as [6, 7], and the parameters of the convolution layers are shared across dual task models. We use a single fully-connected layer with hidden size 128 for the projector. We resize all the images to 32×32, and covert them from grayscale to RGB by duplicating their channels. We set the batch size as 32. Both the generators and the task models are optimized by Adam with learning rate 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Follow [6], we set the training steps E_0, E_1 and E_2 in Algorithm 1 to be 20, 500 and 3000 respectively. The loss weight λ_{sty} and λ_{con} are set to be 0.5 and 1.0 respectively.

For CIFAR10-C, the network configurations for the task models are the same as the Digits dataset except that the backbone network is WideResNet [9] with 16 layers and widen factor 4. We resize all the images to 32×32 and set the batch size to 128. For the task models, we use the SGD optimizer with momentum 0.9 and weight decay 0.0005, and the initial learning rate is 0.1 which is decayed by the cosine annealing scheduler. For the generators, we use the Adam optimizer with learning rate fixed as 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set E_0 to be 9, E_1 to be 30 epochs and E_2 to be 10 epochs. The loss weight λ_{sty} and λ_{con} are both set to be 0.5.

For PACS, we employ the ImageNet-pretrained ResNet18 [3] as the backbone as [8]. We resize all the images to 224×224 and use batch size 32. We use the SGD optimizer with momentum 0.9 and weight decay 0.0005 to train the task models and the generators. The learning rate for the task models is 0.004 decayed by 0.1 at 80% of the total epochs, and for the generators is fixed as 0.001. We set E_0 to be 5, E_1 to be 5 epochs and E_2 to be 20 epochs. The loss weight λ_{sty} and λ_{con} are both set to be 0.5.

For DomainNet, we employ the ImageNet-pretrained ResNet18 [3] as the backbone as [1]. We resize all the images to 224×224 and use batch size 64. We use the SGD optimizer with momentum 0.9 and weight decay 0.0005 to train the task models and the generators. The learning rate for the task models is 0.004 decayed by 0.1 at 80% of the total epochs, and for the generators is fixed as 0.001. We set E_0 to be 3, E_1 to be 5 epochs and E_2 to be 10 epochs. The loss weight λ_{sty} and λ_{con} are both set to be 3.0.

For all the datasets, we instantiate the generator as an encoder-decoder structure with an AdaIN module inserted at the bottleneck layer. Follow [6], both the encoders and decoders are composed of two convolution layers with kernel size 3×3 , stride 1 and padding size 1. All the convolution layers are followed by the ReLU activation except a Sigmoid for the last layer. A random Gaussian noise is mapped to be the scaling and shifting parameter for AdaIN with a single fully connected layer.

B. Details of Different Domain-invariant Regularizations

In addition to the contrastive loss, different domain-invariant regularizations can be incorporated into the SimDE framework. Here we provide the details of the instantiations involved in Table 5.

MSE loss: By using the MSE loss, we directly minimize the Euclidean distance between the features from different domains. Let the features of the original sample x_i, generated sample $\widehat{x}_i^1$ from generator G_1 and $\widehat{x}_i^2$ from generator G_2 to be f_i, $\widehat{f}_i^1$ and $\widehat{f}_i^2$ respectively, the loss is formulated as follows:

$$\mathcal{L}_{mse} = -\frac{1}{2N} \sum_i \left((f_i - \widehat{f}_i^1)^2 + (f_i - \widehat{f}_i^2)^2 \right) \quad (1)$$

JSD loss: By using the JSD loss, we minimize the Jensen–Shannon Divergence between the original source distributions and the generated source distributions. Let the output probability of x_i, $\widehat{x}_i^1$ and $\widehat{x}_i^2$ from task model M_1 to be p_i^1, $\widehat{p}_i^{11}$ and $\widehat{p}_i^{21}$ respectively, and the output probability p_i^2, $\widehat{p}_i^{21}$ and $\widehat{p}_i^{22}$ are defined correspondingly. The JSD loss is formulated as follows:

$$\mathcal{L}_{jsd} = -\frac{1}{3N}\sum\nolimits_i \Big(\mathrm{KL}(p_i^1||\widetilde{p}_i^1) + \mathrm{KL}(\widehat{p}_i^{11}||\widetilde{p}_i^1) + \mathrm{KL}(\widehat{p}_i^{21}||\widetilde{p}_i^1) + \mathrm{KL}(p_i^2||\widetilde{p}_i^2) + \mathrm{KL}(\widehat{p}_i^{12}||\widetilde{p}_i^2) + \mathrm{KL}(\widehat{p}_i^{22}||\widetilde{p}_i^2)\Big) \tag{2}$$

where $\widetilde{p}_i^1 = (p_i^1 + \widehat{p}_i^{11} + \widehat{p}_i^{21})/3$ and $\widetilde{p}_i^2 = (p_i^2 + \widehat{p}_i^{12} + \widehat{p}_i^{22})/3$.

MMD loss: Bu using the MMD loss, we minimize the Maximum Mean Discrepancy [2] between the feature distributions of different domains in the Reproducing kernel Hilbert space. The loss is formulated as follows:

$$\mathcal{L}_{mmd} = \frac{1}{2}\Bigg(\Big|\Big|\frac{1}{N}\sum_{i=1}^{N}\varphi(f_i) - \frac{1}{N}\sum_{i=1}^{N}\varphi(\widehat{f}_i^1)\Big|\Big|_{\mathcal{H}}^2 + \Big|\Big|\frac{1}{N}\sum_{i=1}^{N}\varphi(f_i) - \frac{1}{N}\sum_{i=1}^{N}\varphi(\widehat{f}_i^2)\Big|\Big|_{\mathcal{H}}^2\Bigg) \tag{3}$$

where $\varphi(\cdot)$ is the mapping function and $k(\cdot,\cdot)$ is the kernel function induced by $\varphi(\cdot)$. Here we use the RBF kernel, *i.e.*, $k(f, f') = \varphi(f)^T\varphi(f') = \exp(-\frac{1}{2\sigma}||f - f'||^2)$, where σ is the bandwidth parameter.

Meta learning: Follow [7], we can implement the domain-invariant regularization in the manner of meta learning. Specifically, we choose the original source domain $\mathcal{S}$ as the meta-train set, and the generated domains $\mathcal{S}_1$, $\mathcal{S}_2$ from G_1, G_2 as the meta-test set. During meta-training, we compute the updated parameters $\hat{\theta}$ with one step by minimizing the task loss on the original source domain:

$$\hat{\theta} \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_{ce}(\theta;\mathcal{S}) \tag{4}$$

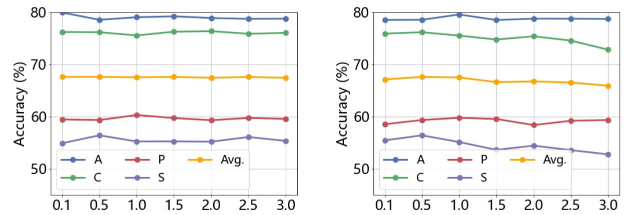
where η is the learning rate. Then during meta-testing, we update θ by the gradient calculated from a combined loss on the original and generated domains with $\hat{\theta}$:

$$\theta \leftarrow \theta - \eta\nabla_\theta\Big(\mathcal{L}_{ce}(\theta;\mathcal{S}) + \frac{1}{2}\sum_{i=1}^{2}\mathcal{L}_{ce}(\hat{\theta};\mathcal{S}_i)\Big) \tag{5}$$

According to [5], by using such a meta-learning strategy, we are implicitly matching the gradients between the original source domain and the generated domains.

C. Sensitivity of the Loss Weights

We conduct sensitivity analysis about the loss balancing weight λ_{sty} and λ_{con} on the PACS dataset. The initial value of λ_{sty} and λ_{con} is both set to 0.5. The results are plotted in Figure 1. For the style divergence loss weight λ_{sty}, although accuracy of different source domains fluctuates sightly with respect to different values of λ_{sty}, the averaged performance is rather stable. Therefore, our method is not very sensitive to the specific values of λ_{sty}. For the contrastive loss weight λ_{con}, the accuracy of larger values drops when the source domain is cartoon and sketch, suggesting that an overly strong domain-invariant regularization would impede model learning. Nevertheless, the overall performance of our method is stable within the range $\lambda_{con} \in [0.1, 2.0]$. In general, we suggest $(\lambda_{sty}, \lambda_{con}) = (0.5, 0.5)$ to be a good starting point.



(a) λ_{sty} (b) λ_{con}

Figure 1. Sensitivity of (a) λ_{sty} and (b) λ_{con} on PACS dataset. "A", "C", "P" and "S" represents the corresponding source domain where the models are trained on, and averaged accuracy on the remaining target domains are reported.

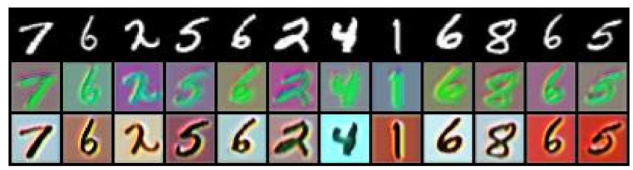


Figure 2. Visualization of the generated images. The 1st, 2nd, 3rd row are original images, generated images from generator G_1 and generated images from G_2 respectively. Data: Digits.

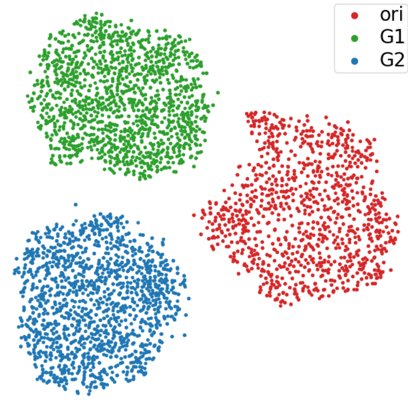


Figure 3. T-SNE visualization of the distributions of original samples and generated sample from G_1 and G_2. Data: Digits.

D. Visualization of the Generations

We visualize the original images, the generated images from generator G_1 and generated images from generator G_2

in Figure 2. It is clear that our method can generate domain-shifted samples by manipulating the superficial statistics of the original images without altering the semantics. Moreover, the dual generators in our method can generate samples from different angles. As shown in Figure 2, generated images from G_1 are mainly composed of green blur digits and gloomy backgrounds, while those from G_2 are composed of black digits and brighter backgrounds. We also plot the sample distribution of the original images and the generated images from G_1 and G_2 with t-SNE in Figure 3. It is clear that samples from both generators form distinct distributions from the original ones, and also differs from each other.

References

[1] Ilke Cugu, Massimiliano Mancini, Yanbei Chen, and Zeynep Akata. Attention consistency on visual corruptions for single-source domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4165–4174, 2022. 1

[2] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 2

[3] 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. 1

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[5] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[6] Lei Li, Ke Gao, Juan Cao, Ziyao Huang, Yepeng Weng, Xiaoyue Mi, Zhengze Yu, Xiaoya Li, and Boyang Xia. Progressive domain expansion network for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 224–233, 2021. 1

[7] 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. 1, 2

[8] Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 834–843, 2021. 1

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