Supplementary Materials for GCISG: Guided Causal Invariant Learning for Improved Syn-to-real Generalization

Gilhyun Nam[1], Gyeongjae Choi[1], and Kyungmin Lee[2]

[1] Agency for Defense Development (ADD), Daejeon, Korea
{ngh707, def6488}@gmail.com
[2] Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
kyungmnlee@kaist.ac.kr

A Implementation Details

A.1 Default Setting

For GCISG, we set balancing hyperparameter λ_G and λ_{CI} to 1. We set λ_{CI} to 0.1 to stabilize training when only causal invariance loss $\mathcal{L}_{\text{CI}}$ was applied to the framework for ablation study.

$$\mathcal{L} = \mathcal{L}_{\text{Task}}(x, y) + \lambda_G \mathcal{L}_G(x) + \lambda_{CI} \mathcal{L}_{\text{CI}}(x). \tag{1}$$

We adopt all pre-trained model weights of ResNet [8] from the PyTorch official repository[3]. For all experiments in this paper, we use SGD optimizer of momentum 0.9 and weight decay 0.0005, constant learning rate scheduling. Our implementation is based on PyTorch and PytorchLightning [7].

A.2 Semantic Segmentation

We use DeeplabV3 [1] with ResNet-50 backbone for our semantic segmentation model which is the same as CSG. It is worth noting that CSG corrected their semantic segmentation model to DeeplabV3 despite they mentioned DeepLabV2 in their original paper[4].

When training, we initialize the backbone model with ImageNet pre-trained weights, while randomizing parameters of the segmentation head. We apply color jittering of altering brightness, contrast, saturation, and hue of training images in range (0.6, 1.4) and use multi-scale training of range (0.75, 1.25).

[3] https://pytorch.org/vision/stable/models.html
[4] https://github.com/NVlabs/CSG/issues/3

Table 1. Effect of freezing Batch Normalization layers during training. * denotes run by us.

Method	BN Unfrozen	BN Frozen
CSG	63.2*	**64.1**
GCISG (ours)	**67.5**	62.8

Table 2. Effect of RandAugment magnitude factor M.

M	w/ $\mathcal{L}_G$	w/o $\mathcal{L}_G$
0	65.7	51.4
3	**67.8**	**58.8**
6	67.5	58.6
9	67.2	57.8
12	67.0	56.8

A.3 Object Detection

We use Faster R-CNN [11] for our base object detector and ResNet-101 with FPN [10] as a backbone for object detection experiments, following similar studies that also conducted experiments on Sim10k to Cityscapes object detection task [14,4].

When training, we initialize the backbone model with ImageNet pre-trained weights, while randomizing parameters of FPN and detection head. We follow the default hyperparameter settings of Faster R-CNN in PyTorch official implementation, and additionally apply color jittering of altering brightness, contrast, saturation, and hue of training images in range (0.6, 1.4).

We re-implement CSG for comparing object detection results, since CSG does not provide object detection experiments. Unlike original CSG implementation[5], we change poly learning rate scheduler to constant learning rate scheduler, and drop layer-wise learning rate scaling factor, which we have found that they do not affect the experiment results, to match hyperparameter settings of CSG with GCISG.

B Additional Ablation Studies

B.1 Batch Normalization

We conduct a study on the effect of freezing weights of Batch Normalization (BN) layers in the student network during training. We compare the result with CSG [2] which also follows a similar architecture. As shown in Table 1, our method shows a significant performance drop of around 4.7% when BN layers are frozen during training. In contrast, CSG shows a slight performance gain of around 0.9% when BN layers are frozen.

B.2 Magnitude of Augmentation

We conduct an ablation study on the effect of augmentation magnitude in syn-to-real generalization by adjusting the global magnitude control factor M of

[5] https://github.com/nvlabs/csg

Table 3. The average precision (AP) of Sim10k to Cityscapes object detection with various methods, according to target object size. * denotes run by us.

Method	AP_{large}	AP_{medium}	AP_{small}
Baseline*	45.2	17.8	3.6
CSG*	49.2	21.8	4.5
GCISG (ours)	**52.9**	**23.5**	**5.0**

RandAugment [6]. We also conduct the ablation study on removing guidance loss to identify the effect of the magnitude of augmentation on the generalization while causal invariant learning. In Table 2, we observe that moderate level ($M = 3$) of augmentation magnitude can improve generalization performance regardless of the guidance loss. However, in GCISG, we set global magnitude control factor M to 6 for a fair comparison with CSG.

B.3 Object Detection Results According to Object Size

In Table 3, we report the object detection results on Sim10k to Cityscapes according to target object size range following the object detection evaluation metric of COCO[6]. We observe that GCISG outperforms other competing methods consistently on every target object size range.

C Detailed Explanations for Evaluation Metrics

C.1 Match Rate

In section 4, we introduce match rate $\mathcal{M}$ to measure the style-invariance of a classifier C. Let D_{val} be a validation dataset and $\mathcal{A}$ be a photometric image transformation operator that alters the style of an image. For each validation data $x \in D_{\text{val}}$, we generate a transformed counterpart $\mathcal{A}(x)$. Then the match rate $\mathcal{M}(C, D_{\text{val}}, \mathcal{A})$ is defined by the fraction of consistent prediction over the total number of validation samples:

$$\mathcal{M}(C, D_{\text{val}}, \mathcal{A}) = \frac{|D_{\text{con}}|}{|D_{\text{val}}|}, \tag{2}$$

where $D_{\text{con}} = \{x \in D_{\text{val}} : C(x) = C(\mathcal{A}(x))\}$. The photometric transform $\mathcal{A}$ is composed of Gaussian blurring with $\sigma \in [0.15, 1.3]$ and color jittering with parameter range of [0.6, 1.4] for brightness, contrast, saturation, and hue. Similarly, RobustNet [5] applied photometric transform consisting of random Gaussian blurring and color jittering to simulate style shift.

[6] https://cocodataset.org/#detection-eval

C.2 CKA similarity

The centered kernel alignment (CKA) similarity is a statistical measure that is invariant to the linear and orthogonal transformation as well as isotropic scaling. Thus it can effectively quantify the similarity between the two neural representations [9]. We introduced CKA similarity to measure the similarity between our synthetic-training model and real-guidance model in section 4. Remark that the evaluation of the similarities of neural networks with CKA similarity has also been proposed in various machine learning tasks such as knowledge distillation [13,15].

Let $X \in \mathbb{R}^{n \times p_1}$ denote a matrix of activations of p_1 neurons for n samples, and $Y \in \mathbb{R}^{n \times p_2}$ denote a matrix of activations of p_2 neurons for the same n samples. For kernel k and l, let $K_{ij} = k(X_i, X_j)$ and $L_{ij} = l(Y_i, Y_j)$. Then the empirical estimator of a *Hilbert-Schmidt Independence Criterion* (HSIC) is defined by following:

$$HSIC(K, L) = \frac{1}{(n-1)^2}\mathrm{tr}(KHLH), \tag{3}$$

where $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. Then the CKA similarity is defined by following:

$$\mathrm{CKA}(K, L) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K)\mathrm{HSIC}(L, L)}}. \tag{4}$$

Note that the linear kernel $k(X_i, X_j) = X_i^\top X_j$ and RBF kernel $k(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right)$ is widely used for computation of CKA similarity. In the main paper, we report the CKA similarity with the linear kernel. We implemented CKA similarity computation from [7].

D Visualization of Attention on VisDA-17

In Fig. 1 and Fig. 2, we present visualized attention maps on validation images using Grad-CAM [12]. In Fig. 1, we compare attention maps of various pooling methods that were introduced in this paper. Remark that the model trained with self-attention pooling better captures the semantic object than other pooling methods. Also, in Fig. 2, we compare attention maps according to training methods: Oracle on ImageNet (freeze backbone and fine-tuning), baseline (training without any regularization), and our method GCISG. We observe that even though the backbone is frozen, the Oracle method cannot capture the semantic region of an image. The baseline method tends to capture a partial semantic feature of an image, since the synthetic data contain nuisance style variables that hinder learning the semantics of natural image data. On the other hand, our method GCISG can effectively spot the semantic objects in an image.

[7] `https://github.com/jayroxis/CKA-similarity`

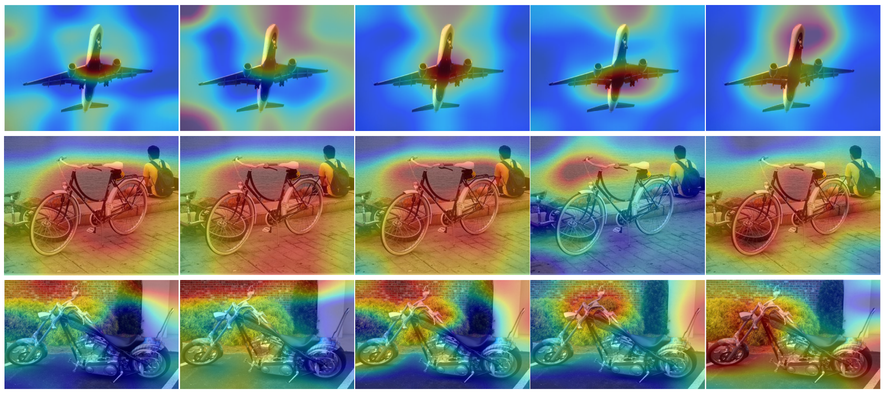

Fig. 1. Visualized attention of feature according to pooling methods. From left to right, columns correspond to NP, GAP, CP, SP, SAP. The red area corresponds to high score for class.

E Additional Information of Experiment Results

In Table 1 and 6 of main paper, we reported results of other methods referring to ASG [3], CSG [2] and RobustNet [5].

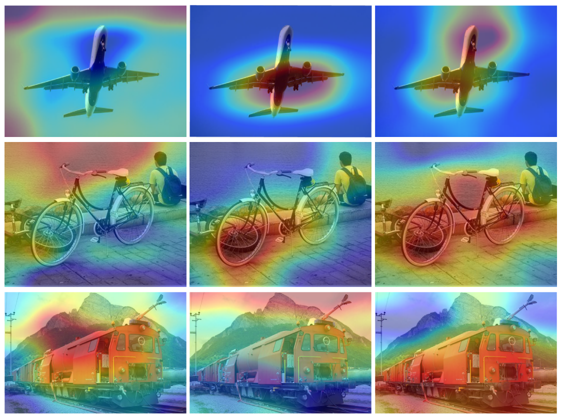

Fig. 2. Visualized attention of feature according to train methods. From left to right, columns correspond to Oracle on ImageNet, Baseline, GCISG. The red area corresponds to high score for class.

References

1. Chen, L., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587 (2017)
2. Chen, W., Yu, Z., Mello, S., Liu, S., Alvarez, J.M., Wang, Z., Anandkumar, A.: Contrastive syn-to-real generalization. In: ICLR (2021)
3. Chen, W., Yu, Z., Wang, Z., Anandkumar, A.: Automated synthetic-to-real generalization. In: International Conference on Machine Learning. pp. 1746–1756. PMLR (2020)
4. Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L.: Domain adaptive faster r-cnn for object detection in the wild. In: CVPR (2018)
5. Choi, S., Jung, S., Yun, H., Kim, J.T., Kim, S., Choo, J.: Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In: CVPR (2021)
6. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: CVPR (2020)
7. Falcon, W., The PyTorch Lightning team: PyTorch Lightning (2019), https://github.com/PyTorchLightning/pytorch-lightning
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
9. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: ICML (2019)
10. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. arXiv:1612.03144 (2016)
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: TPAMI (2017)
12. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: ICCV (2017)
13. Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A.A., Wilson, A.G.: Does knowledge distillation really work? In: NIPS (2021)
14. Wang, X., Huang, T.E., Liu, B., Yu, F., Wang, X., Gonzalez, J.E., Darrell, T.: Robust object detection via instance-level temporal cycle confusion. In: ICCV (2021)
15. Xu, G., Liu, Z., Li, X., Loy, C.C.: Knowledge distillation meets self-supervision. In: ECCV (2020)