DRANet: Disentangling Representation and Adaptation Networks for Unsupervised Cross-Domain Adaptation (Supplementary Material)

Seunghun Lee
DGIST
lsh5688@dgist.ac.kr

Sunghyun Cho
POSTECH CSE & GSAI
s.cho@postech.ac.kr

Sunghoon Im*
DGIST
sunghoonim@dgist.ac.kr

1. Motivation for Separator Design

In this section, we provide an additional discussion on the motivation of our network design with a separator with a comparison of different network structures. Traditional domain adaptation methods [2, 3] learn encoders for each source and target images to be mapped into domain invariant space as illustrated in Fig. 1**-(a). The individual characteristics of each domain can be disregarded on the domain invariant space, so it can degrade the task discrimination. Note that, during the training process, they impose losses to minimize task discrimination and maximize domain discrimination.

While the researches on domain adaptation focus on the domain invariance, a recent study on style transfer [11] attempts to disentangle image representation into content and style components and transfer the style. We observe that the representation disentanglement can bring great advantages to the domain adaptation, such as (1) preserving each domain's characteristics, (2) improving adaptation performance, and (3) transferring domains multi-directionally with a single network. However, the work [11] separates the representations linearly in the complex non-linear manifold space in Fig. 1-(b). As demonstrated in an experiment in Fig. 1, their simple linear separation scheme often fails to disentangle the content and style, and consequently, their approach often produces incorrectly adapted results.

To overcome the limitations of the linear separation while taking the advantage of representation disentanglement, in our work, we propose a Disentangling Representation and Adaptation Network. *DRANet* contains a separator composed of two convolutional layers as non-linear function and domain-specific scale parameters as illustrated in Fig. 1-(c). Our intuition behind the network design is that different domains may have different distributions for their contents and styles, which cannot be effectively handled by a linear separation using a single encoder. Thus, to handle such difference, our network adopts the non-linear separation and domain-specific scale parameters that is dedicated to handle such inter-domain difference. As the proposed separator provides domain-adaptive separation of the content and style, our encoder and decoder can handle images from different domains in a unified way and still achieve the state-of-the-art domain adaptation results.

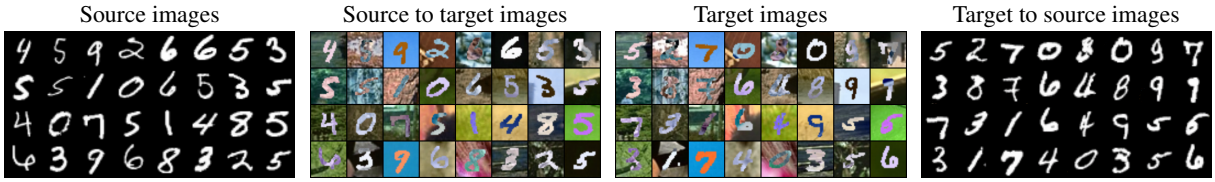
We compare the results of our method in Fig. 1-(a), (c), and that of a linear separation approach in Fig. 1-(b), (d). The first and third columns in the figure depict source and target domain images, respectively, while the second and fourth columns show domain-adapted results. If the content and style features were well-separated, then the images in the second column should have the contents of the source domain images and the style of the target domain images. Similarly, the images in the fourth column should have the contents of the target domain images and the style of the source domain images. However, as shown in Fig. 1-(b), the images in the fourth column have the same contents with the source domain images, which indicates that the method failed to separate the content and style. Also, as shown in Fig. 1-(d), the failure of separating the content and style can cause mode collapse, which is an undesirable behavior of GAN-based methods. As shown in this experiment, without careful hyperparameter adjustment, the training of the model with simple assumption of [11] easily goes to wrong.

2. Implementation Details

We summarize our training algorithm in Algorithm 1. We first learn all discriminator $D \in \mathcal{D}$ to maximize the adversarial loss, where $\mathcal{D}$ is the set of discriminators. Then, we learn the encoder E, separator S, and generator G to minimize the reconstruction, consistency, adversarial and perceptual loss functions. For the balance of learning, we run two iterations of learning for E, S, and G while running one iteration for D.

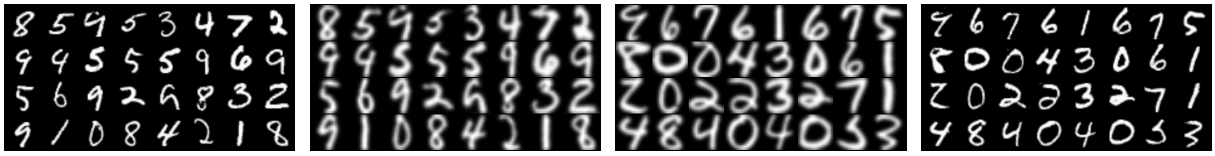
The trained network produces the domain adapted images $I_{X \to Y}$ transferred from source domain X to target do-

*Corresponding author.
**Blue number indicates figure in main paper.

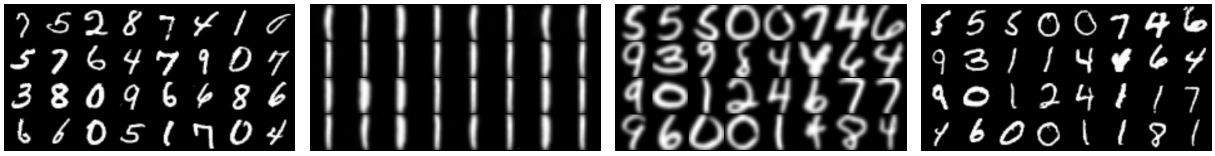


(a) Results of our network between MNIST and MNIST-M

(b) Results of a linear separator between MNIST and MNIST-M



(c) Results of our network between MNIST and USPS



(d) Results of a linear separator between MNIST and USPS

Figure 1. Domain adaptation results of (a), (c) our network illustrated in Fig. 1-(c) and (b), (d) a linear separator illustrated in Fig. 1-(b). **Note that, the contents shown in (b), (d) are not preserved for many cases.**

main Y, which will be used to train the task network T. We train the network, either a classification or a segmentation network in this paper:

$$\min_{T} \mathcal{L}_{Task}^{X \to Y}(I_X, I_{X \to Y}, \mathcal{Y}_X), \tag{1}$$

where $\mathcal{Y}_X$ is the ground-truth labels corresponding to source images I_X. Note that, we do not use the ground-truth labels of target domain Y to train the task network, and after training, we evaluate the performance of the task network using the target domain test sets. We use a typical softmax cross-entropy loss as the task loss for both classification and segmentation. To further improve the performance of a task network, we train it every single iteration of *DRANet* training. We use the Adam optimizer with learning rate 1e-3 for learning all networks in *DRANet*. For learning the classification network, we use the stochastic gradient descent (SGD) optimizer with learning rate 5e-4 and momentum 0.9. For learning the segmentation network, we use the SGD optimizer with learning rate 2.5e-4 and momentum 0.9.

We use batches of 32 samples resized to 64×64 for the digit adaptation, and batches of 2 samples resized to 512×1024 for the driving scene adaptation. Our network architecture is illustrated in Fig. 2. For stable training, we apply spectral normalization [6] to all layers in the generator and discriminator except the residual blocks. We find that batch instance normalization [7] is more effective than batch normalization [4] and instance normalization [10] in our framework. We use an ImageNet-pretrained VGG-19 network [9] as the perceptual network, and compute the content perceptual loss at layer relu4_2 and the style perceptual losses at layers relu1_1, relu2_1, relu3_1, and relu4_1. We set the domain-specific scale parameters of the same size as features $\mathcal{F}$, and perform element-wise multiplication for scaling.

Algorithm 1 Algorithm for training *DRANet*

Input: I_X, I_Y
// Source and target domain images
Output: $I_{X \to Y}, I_{Y \to Y}, I'_X, I'_Y$
// Domain transferred and reconstructed images
for $k = 1 \ldots K$ **do**
  $\theta_D^* = \text{argmax}_{\theta_D} \mathcal{L}_{GAN}$, where $D \in \mathcal{D}$
  // Update every D
  for $i = 1$ to 2 **do**
    $\theta_E^*, \theta_S^*, \theta_G^* =$
    $\text{argmin}_{\theta_E, \theta_S, \theta_G} (\mathcal{L}_{Rec} + \mathcal{L}_{Con} + \mathcal{L}_{GAN} + \mathcal{L}_{Per})$
    // Update every E, S, G
  end for
end for

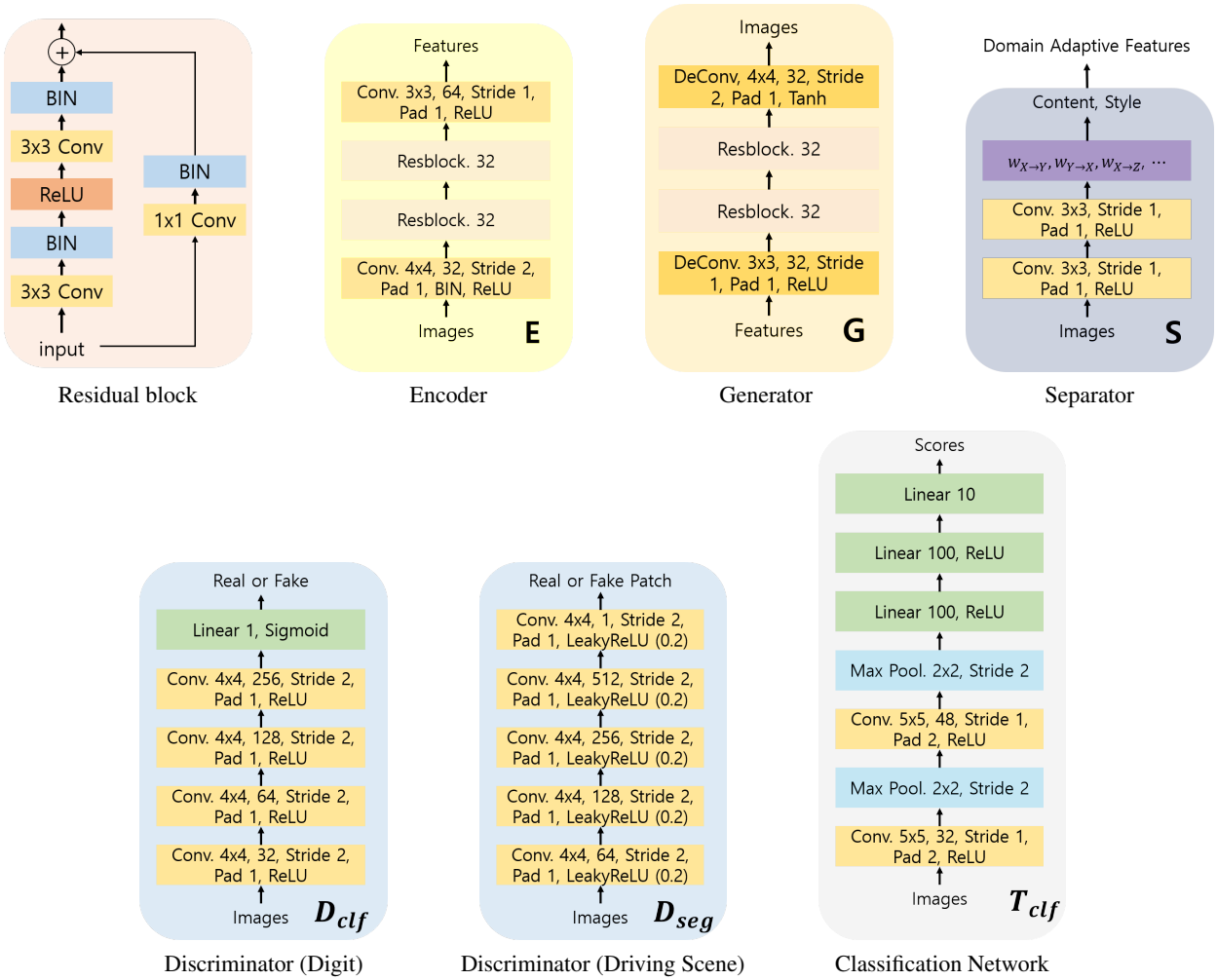


Figure 2. Our network details including the encoder E, separator S, generator G, discriminators D, and task network T. BIN and Resblock are batch instance normalization layer and residual block, respectively. D_{Seg} has same structure with PatchGAN discriminator [5].

3. Effect of Content-Adaptive Domain Transfer (CADT)

Fig. 3 presents zoom-in views of Fig. 8 in the main paper for better comparison. Specifically, the first row of the figure shows a source image from the GTA5 dataset [8], its transformed images with CADT and without CADT, respectively, and a randomly selected target image from the CityScapes dataset [1]. The second and third rows show zoom-in views of their corresponding images on the first row. As described in Fig. 8 in the main paper, Fig. 3-(b) is obtained using CADT at inference time using four target images while Fig. 3-(c) is obtained using the target image in Fig. 3-(d) without CADT. As shown in Fig. 3-(b), the model with CADT successfully generates a visually pleasing result without artifacts. On the other hand, Fig. 3-(c) shows ghosting artifacts caused by the target image in Fig. 3-(d) that is semantically distant from the source image. This result indicates that the style feature from Fig. 3-(d) has content-related information. We found that CADT is especially effective in training our networks. Specifically, it helps the networks to learn better separation of the content and style of an image, and eventually leads to better domain adaptation results as shown in Fig. 3.

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. 3

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1

(a) Source image (GTA5) (b) Result with CADT (c) Result without CADT (d) Target image (Cityscapes)

Figure 3. Comparison on image synthesis using Content-Adaptive Domain Transfer (CADT) and normal Domain Transfer (DT).

[3] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning (ICML)*, pages 1989–1998. PMLR, 2018. 1

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning (ICML)*, 2015. 2

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017. 3

[6] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2018. 2

[7] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2558–2567, 2018. 2

[8] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 102–118. Springer, 2016. 3

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015. 2

[10] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2

[11] Rui Zhang, Sheng Tang, Yu Li, Junbo Guo, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Style separation and synthesis via generative adversarial networks. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 183–191, 2018. 1