Adversarial Style Augmentation for Domain Generalized Urban-Scene Segmentation (Supplementary Material)

Zhun Zhong[1]*† **Yuyang Zhao**[2]* **Gim Hee Lee**[2] **Nicu Sebe**[1]

[1] Department of Information Engineering and Computer Science, University of Trento

[2] Department of Computer Science, National University of Singapore

A Details of Datasets in Domain Generalized Semantic Segmentation

For the synthetic-to-real domain generalization (DG), we use one of the synthetic datasets (GTAV [12] or SYNTHIA [13]) as the source domain and evaluate the model performance on three real-world datasets (CityScapes [2], BDD-100K [16], and Mapillary [11]).

Synthetic datasets. GTAV [12] contains 24,966 images with the size of 1914×1052. It is splited into 12,403, 6,382, and 6,181 images for training, validating, and testing. SYNTHIA [13] contains 9,400 images of 960×720, where 6,580 images are used for training.

Real-world datasets. We use the validation sets of the three real-world datasets for evaluation. CityScapes [2] contains 500 validation images of 2048×1024, collected primarily in Germany. BDD-100K [16] and Mapillary [11] contain 1,000 validation images of 1280×720 and 2,000 validation images of 1920×1080, respectively.

B Details of Single Domain Generalization in Image Classification

Digits includes five domains (MNIST [8], SVHN [10], MNIST-M [4], SYN [4], and USPS [7]) of 10 classes. We use MNIST as the source domain and evaluate the model performance on the other 4 domains. Following ADA [15], we use the ConvNet architecture [8] as the model and use Adam optimizer with learning rate 10^{-4} for optimization. The overall training iteration is set to 10,000 with a batch size of 32. We set the learning rate of AdvStyle to 20,000[2].

PACS [9] contains four domains (Artpaint, Cartoon, Sketch, and Photo) of 7 classes. For evaluation, we select one of them as the source domain and the other domains as the target domains. Following RSC [6], we use the ResNet18 [5] pretrained on ImageNet [3] as the backbone and add a fully-connected layer as the classification head. We train the model by SGD optimizer. The learning rate is initially set to 0.004 and divided by 10 after 24 epochs. The model is trained for 30 epochs in total with a batch size of 128. The learning rate of AdvStyle is set to 3.

Baseline. The baseline model is the vanilla empirical risk minimization (ERM) [14], which directly uses the source domain to train the model with classification loss.

C Position of AdvStyle

We inject AdvStyle at different positions (0-4) to verify the effectiveness of image level augmentation. 0-th indicates the image level. 1st-4th indicate the outputs of 1st-4th layer of ResNet, respectively.

*Equal contribution. † Corresponding author.

[2] Due to the absent of batch normalization layer, the gradient is very small on the style feature. Therefore, we set a large learning rate for AdvStyle.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Results are shown in Table 1. We can find that injecting AdvStyle at 0th-2nd positions clearly improves the performance and the best result is achieved by applying at 0-th position. Moreover, applying AdvStyle at a deep layer (*e.g.*, 3rd or 4th) fails to improve or even hurts the performance, since more semantic content will be captured instead of styles as the layer deepens.

Table 1: Impact of injecting AdvStyle at different positions.

Position	N/A	0	1	2	3	4
Mean	27.42	**37.39**	34.76	31.15	27.44	26.92

D Comparison of Adversarial Augmentations

AdvPixel [15] is a state-of-the-art method for domain generalized image classification. The main difference between AdvPixel and AdvStyle is that AdvPixel learns pixel-wise adversarial example while AdvStyle learns style-wise adversarial example. The model needs to produce the per-pixel predictions in semantic segmentation. In such a context, AdvPixel may distort the semantic content of original pixels during the pixel-wise adversarial learning. Instead, AdvStyle varies the style feature of the image while retaining the semantic content of most pixels. Therefore, AdvStyle can well guarantee the pixel-wise semantic consistency, making it more suitable for augmenting samples of segmentation. As shown in Table 2, both AdvStyle and AdvPixel improve the performance, while AdvStyle outperforms AdvPixel by 3.42% in mean mIoU. More interestingly, AdvPixel can serve to enhance AdvStyle. We randomly select one adversarial augmentation from AdvPixel and AdvStyle at each iteration. The performance yields an improvement of 0.81% in mean mIoU. The above results verify the effectiveness of adversarial augmentations and the superiority of AdvStyle.

Table 2: Comparison of adversarial augmentations. Source: GTAV; Backbone: ResNet-50. CJ: Color Jittering, GB: Gaussian Blur, AP: AdvPixel, Ours: AdvStyle.

CJ	GB	AP	Ours	CityScapes	BDD	Mapillary	Mean
✓	✓	-	-	28.95	25.14	28.18	27.42
✓	✓	✓	-	35.42	33.28	33.23	33.97
✓	✓	-	✓	39.62	35.54	**37.00**	37.39
✓	✓	✓	✓	**40.65**	**37.16**	36.77	**38.20**

E Variants of AdvStyle

In this section, we investigate two variants of AdvStyle, which can further demonstrate the versatility of AdvStyle. Also, we hope to provide some inspirations for future work.

AdvStyle in local patches. AdvStyle can be applied to not only the whole image but also the local patches. Specifically, we split each image into 4 patches evenly (top left, top right, bottom left, and bottom right), and regard the channel-wise mean and standard deviation of each patch as learnable parameters (four 6-dim features). Then the model is trained in the same way as AdvStyle. As shown in the Table 3, AdvStyle-Patches can further improve the performance on BDD and Mapillary. However, the mean improvement over all domains is marginal.

AdvStyle in LAB color space. AdvStyle is applied to RGB space in this paper, but it can also be applied to other color space, *e.g.*, LAB color space. To verify this, we first convert the RGB-sample to the counterpart LAB-sample and obtain the learnable mean and standard deviation. Then, we reconvert the LAB-sample to RGB-sample for adversarial learning and model optimization. This manner enables us to implement AdvStyle in the LAB space as well as to use the ImageNet-pretrained parameters. As shown in the Table 3, LAB-based AdvStyle (AdvStyle-LAB) also significantly improves the performance on unseen domains but achieves lower results than RGB-based AdvStyle on two of the three benchmarks. On the other hand, converting between RGB and LAB will increase the training time due to the extra computation costs.

Table 3: Results of AdvStyle variants. The backbone is ResNet-50.

Methods (GTAV$\rightarrow$)	CityScapes	BDD	Mapillary	Mean
Baseline [1]	28.95	25.14	28.18	27.42
AdvStyle-LAB	37.09	32.89	37.13	35.70
AdvStyle-Patches	39.50	**36.37**	**37.42**	**37.76**
AdvStyle	**39.62**	35.54	37.00	37.39

F Quantitative Understanding of AdvStyle

To demonstrate the effectiveness of AdvStyle in narrowing the domain shift, we provide the quantitative analysis on the distribution of different datasets. Specifically, we computed the histograms of pixel values of four datasets (GTAV [12], CityScapes [2], BDD-100K [16], Mapillary [11]) and the AdvStyle-augmented dataset of GTAV which is generated by 4 epochs. The bin size is set to 8. For each dataset, the histograms of RGB channels are normalized by L1-norm and re-scaled ($\times$) by $\#$bins, and then are concatenated as the histogram feature. We estimate the distribution distance between two datasets by computing the KL-distance between their histogram features. Results are reported in Table 4. We can observe that the AdvStyle-augmented dataset has a smaller distance to real datasets, verifying that AdvStyle can narrow the gap between synthetic and real data.

Table 4: Comparison of KL-distance between different datasets.

Source	CityScapes $\downarrow$	BDD $\downarrow$	Mapillary $\downarrow$	Mean $\downarrow$
GTAV	0.5867	0.3421	0.3211	0.4166
Adv-GTAV	**0.5587**	**0.3217**	**0.3058**	**0.3954**

G Algorithm and Pytorch-Like Pseudo-Code

The training procedure and Pytorch-like pseudo-code are shown in Alg. 1 and Fig. 1, respectively.

Algorithm 1: The training procedure of AdvStyle.

Inputs: labeled source domain $\mathcal{S}$, segmentation model $\mathcal{F}$ parameterized by θ, batch size $N_{\ b}$, total training iterations max_iter, adversarial learning rate $_\gamma$, and model learning rate α.
Outputs: Optimized model $\mathcal{F}$ parameterized with θ.

1: **for** i in max_iter **do**
2: Sample mini-batch $\mathcal{X}$ with N_b images;
3: // Stage 1: Adversarial Style Learning.
4: Compute channel-wise mean μ, standard deviation σ and normalized images $\bar{\mathcal{X}}$ with Eq. 2;
5: Initialize adversarial style feature: $\mu^+ \leftarrow \mu$, $\sigma^+ \leftarrow \sigma$;
6: Compute adversarial segmentation loss $-\mathcal{L}_{\text{seg}}$;
7: Optimize μ^+ and σ^+ with Eq. 3;
8: // Stage 2: Robust Model Training.
9: Generate adversarial images $\mathcal{X}^+$ with $\bar{\mathcal{X}}$, μ^+ and σ^+ by Eq. 4;
10: Compute the overall training loss $\mathcal{L}_{\text{seg}}(\theta; \mathcal{X}) + \mathcal{L}_{\text{seg}}(\theta; \mathcal{X}^+)$ by Eq. 5;
11: Optimize the segmentation model $\mathcal{F}$: $\theta \leftarrow \theta - \alpha \nabla_\theta \left(\mathcal{L}_{\text{seg}}(\theta; \mathcal{X}) + \mathcal{L}_{\text{seg}}(\theta; \mathcal{X}^+)\right)$;
12: **end for**
13: **Return** $\mathcal{F}$ parameterized with θ.

H More Visualizations

Segmentation Results. In Fig. 2, Fig. 3, and Fig. 4, we provide more segmentation results for the baseline and "baseline+AdvStyle".

Examples of AdvStyle. In Fig. 5, we illustrate more examples generated by AdvStyle.

I Limitations

The main limitation of AdvStyle lies in the increase of training time. The computational cost of AdvStyle is almost double of that of the baseline since one more forward-backward process is required to generate style-adversarial examples. Another limitation is that despite generating hard examples, AdvStyle cannot address severe environmental change in practice, *e.g.*, rainy and snowy weather, since such conditions cannot be represent purely by style features. Those conditions, *e.g.*, rain, snow and fog, can be added to source samples and adversarial-augmented samples manually to alleviate the problem.

````
import torch

def AdvStyle(input, gt, net, optim, adv_lr):
    ```
    Args:
            input: source images
            gt: ground-truth labels
            net: segmentation network
            optim: optimizer of net
            adv_lr: learning rate of AdvStyle
    ’’’
    ### Adversarial Style Learning

    # Get style feature and normalized image
    B = input.size(0)
    mu = input.mean(dim=[2, 3], keepdim=True)
    var = input.var(dim=[2, 3], keepdim=True)
    sig = (var + 1e-5).sqrt()
    mu, sig = mu.detach(), sig.detach()
    input_normed = (input - mu) / sig
    input_normed = input_normed.detach().clone()

    # Set learnable style feature and adv optimizer
    adv_mu, adv_sig = mu, sig
    adv_mu.requires_grad_(True)
    adv_sig.requires_grad_(True)
    adv_optim = torch.optim.SGD(params=[adv_mu, adv_sig], lr=adv_lr, momentum=0, weight_decay=0)

    # Optimize adversarial style feature
    adv_optim.zero_grad()
    adv_input = input_normed * adv_sig+ adv_mu
    adv_output = net(adv_input)
    adv_loss = torch.nn.functional.cross_entropy(adv_output, gt)
    (- adv_loss).backward()
    adv_optim.step()

    ### Robust Model Training
    net.train()
    optim.zero_grad()
    adv_input = input_normed * adv_sig + adv_mu
    inputs = torch.cat((input, adv_input), dim=0)
    gt = torch.cat((gt, gt), dim=0)
    outputs = net(inputs)
    loss = F.cross_entropy(outputs, gt)
    loss.backward()
    optim.step()
````

Figure 1: The Pytorch-like pseudo-code of AdvStyle.

References

[1] Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, 2021.

[2] 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 *CVPR*, 2016.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
````

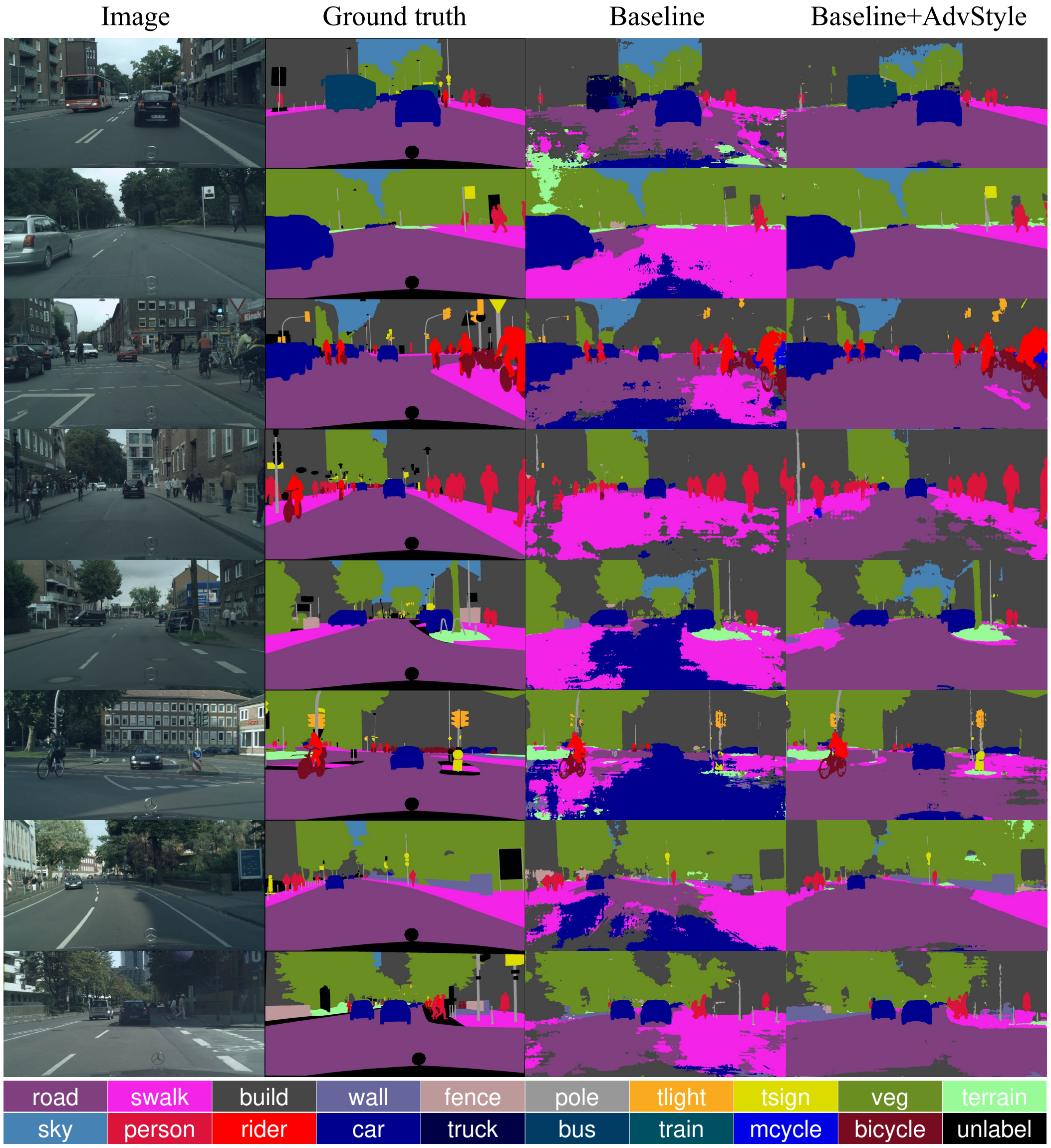

Figure 2: Segmentation results on CityScapes. Source: GTAV; Backbone: ResNet-50.

[4] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] Zeyi Huang, Haohan Wang, Eric P. Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

[7] Jonathan J. Hull. A database for handwritten text recognition research. *TPAMI*, 1994.

[8] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[9] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

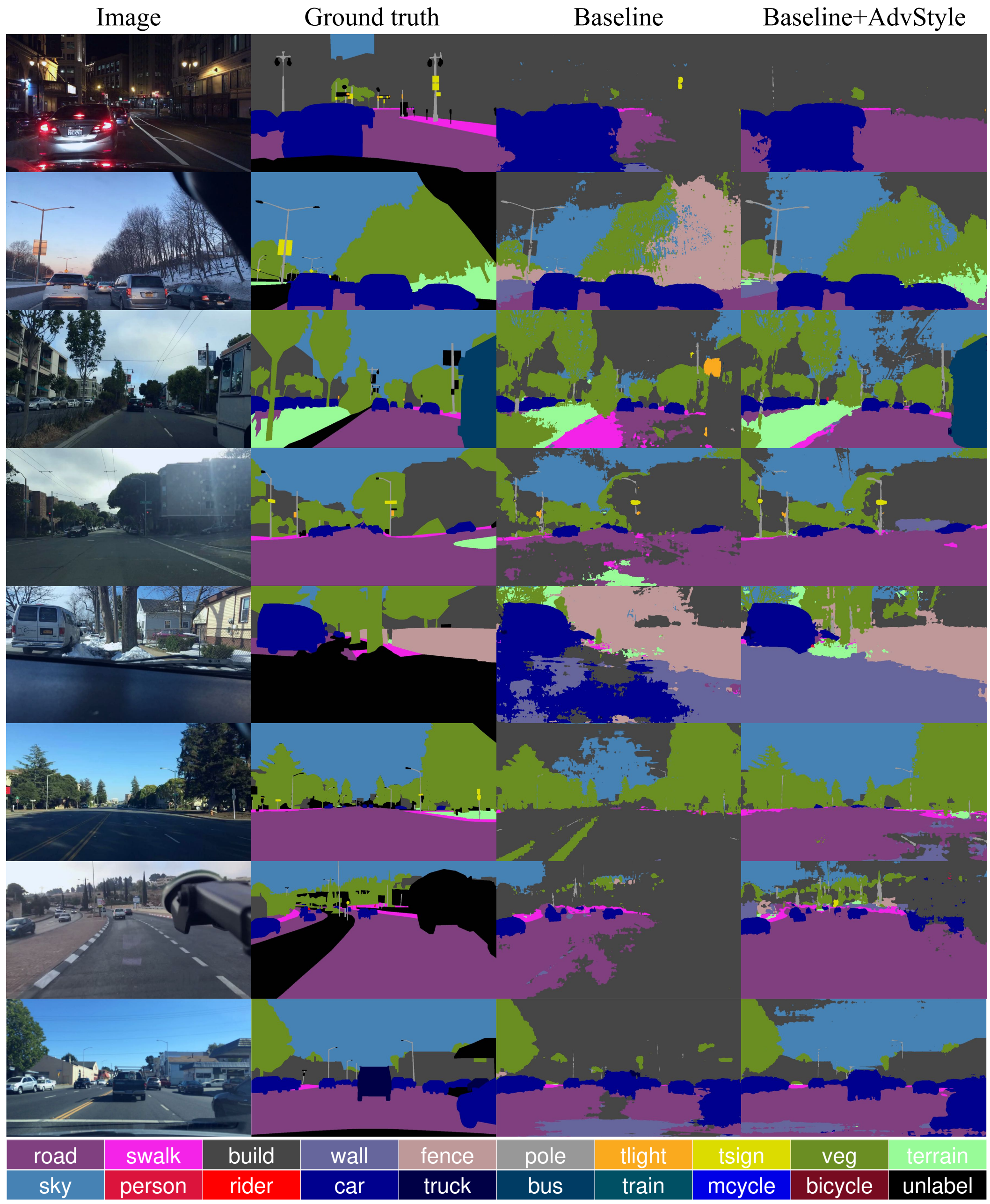

Figure 3: Segmentation results on BDD-100K. Source: GTAV; Backbone: ResNet-50.

[10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop*, 2011.

[11] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.

[12] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[13] 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 *CVPR*, 2016.

[14] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

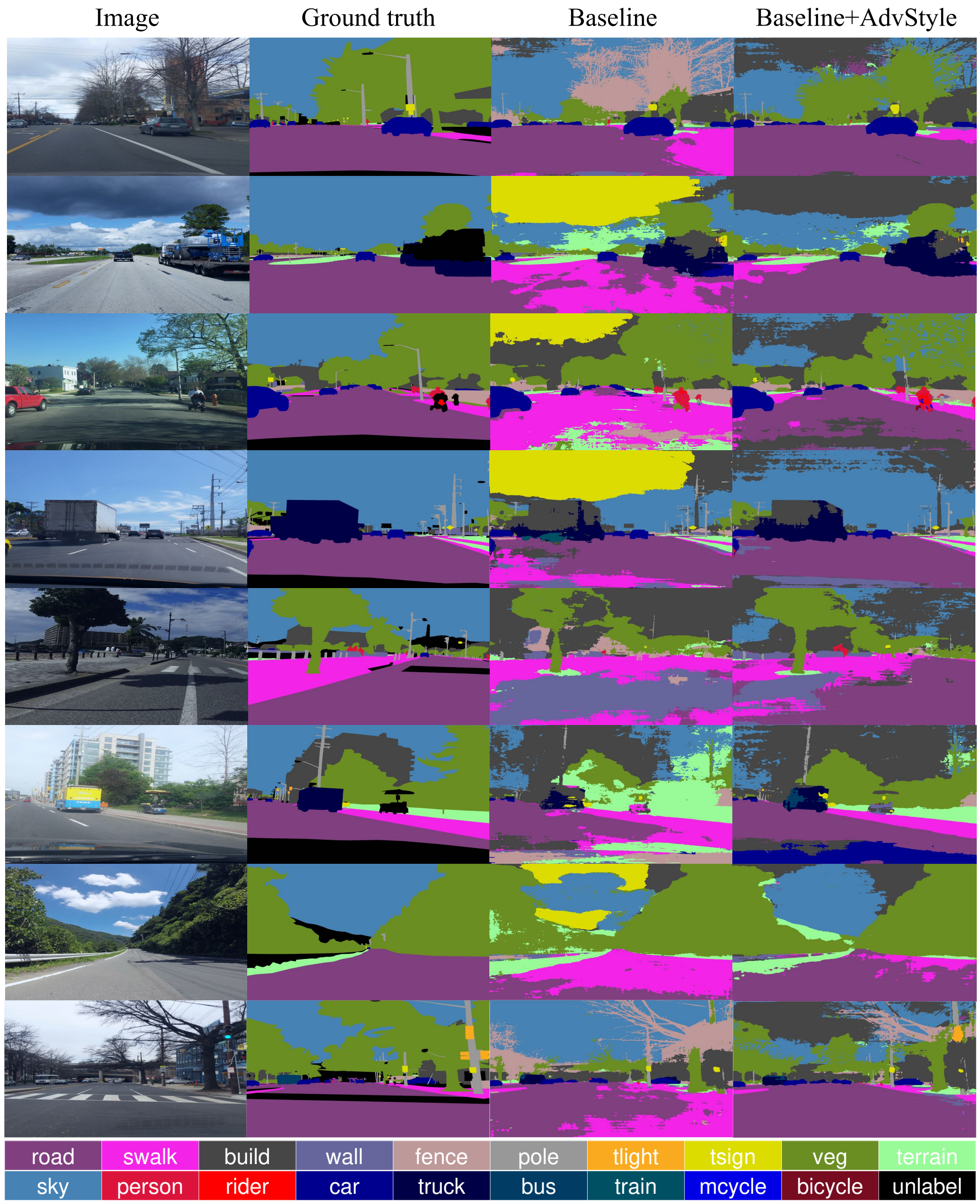

Figure 4: Segmentation results on Mapillary. Source: GTAV; Backbone: ResNet-50.

[15] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018.

[16] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020.

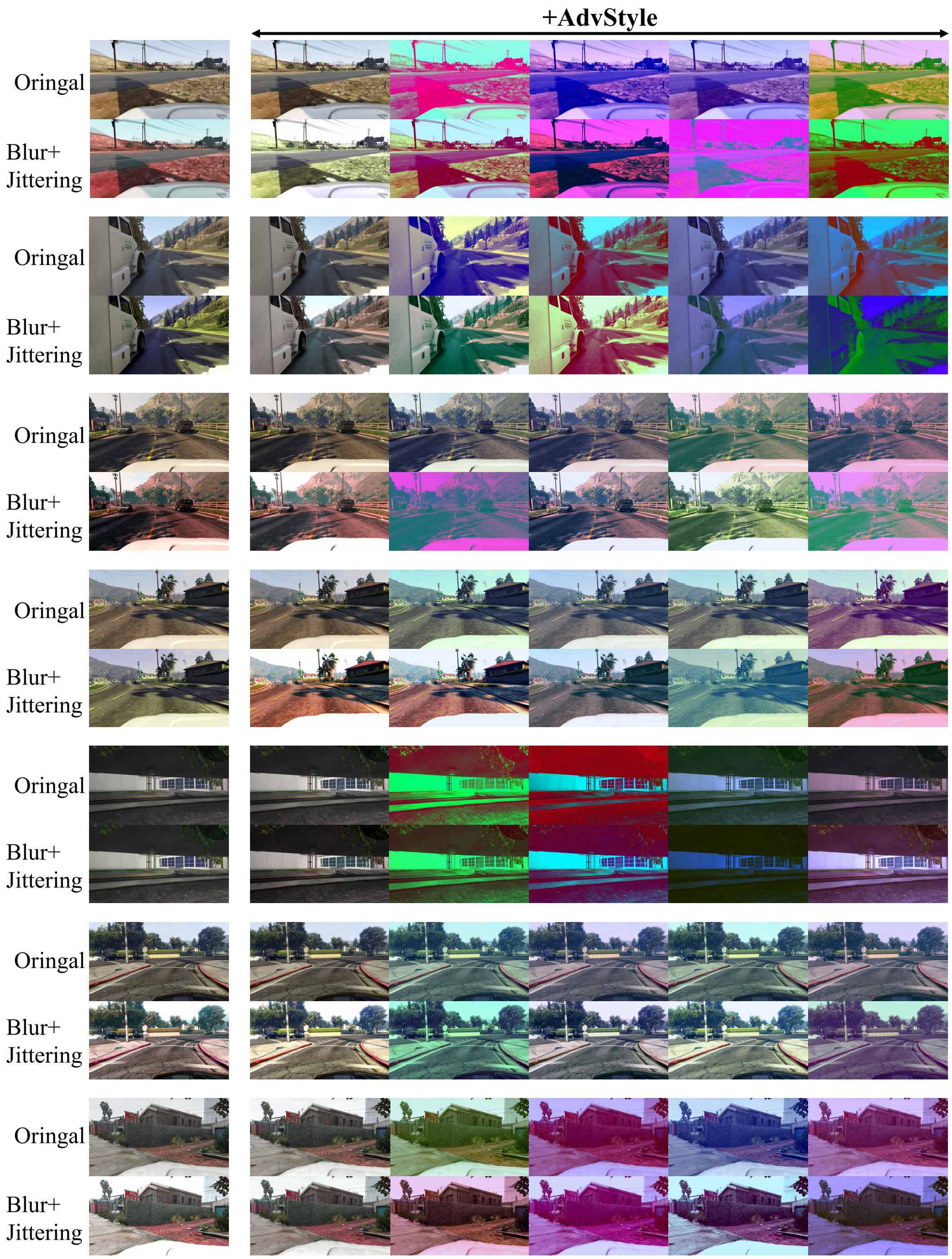

Figure 5: Examples of adversarial style augmentation. Source: GTAV; Backbone: ResNet-50.