NANYANG TECHNOLOGICAL UNIVERSITY

AI6121 Project: Image Translation and UDA

YU YUE, G2202151A
JU XILAI, G2202544B
LUO HAO, G2202279H

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2022

Contents

Abstract

Image-to-image translation (I2I) technology is an efficient approach to group the pixels in an image. With the help of I2I technology, tasks such as style transfer and semantic segmentation can be achieved by training a neural network on the specific datasets that well labeled. However, in the real training, labeling is very time-consuming and very limited labeling datasets are accessible. Therefore, an approach called unpaired image-to-image translation has been proposed, such as CycleGAN. In this project, we adopt a traditional image-to-image model CycleGAN [1] to implement semantic segmentation trained on GTA5 dataset [2].

Besides, in the real world, to train a self-driving car's vision system by semantic segmentation, due to some privacy rules, the typical approach is to obtain street views from electronic games like GTA5 and then train the model on these images, and then test the model on the real images. In this project, we apply the trained model based on GTA5 to Cityscapes datatset [3]. However, because of the domain gap between GTA5 and Cityscapes, the generalization ability of the CycleGAN model trained only on GTA5 is not ideal enough.

To narrow the gap, We then implement input-space unsupervised domain adaptation (UDA) via training another CycleGAN model to achieve style transfer between GTA5 style and Cityscapes style. After that, two approaches are designed to test the input-space UDA with CycleGAN: first, we train the third CycleGAN with translated GTA5 dataset generated by UDA model, and test on Cityscapes dataset; second, we directly translate Cityscapes dataset to GTA5 style by UDA model, and then generate semantic segmentation maps by the first CycleGAN trained only on GTA5 dataset. The report describes the details of the implementation and compares the performances before and after practising UDA via input-space alignment. All the code can be download from github repository.

Keywords: semantic segmentation, image-to-image translation, CycleGAN, input-space unsupervised domain adaptation.

Chapter 1

CycleGAN

1.1 Introduction

Image-to-image translation aims to translate a set of images from one style to another by learning the mappings between two sets of images via neural network technologies. However, paired training datasets are rare for many tasks such as semantic segmentation for street view images due to the huge workload of annotating. Therefore, in the paper *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*, Zhu et al. proposed CycleGAN for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. To be specific, CycleGAN learns the mapping $G : X \rightarrow Y$ by using an adversarial loss, such that the distribution of $G(X)$ is indistinguishable from the distribution of Y. Besides, an inverse mapping $F : Y \rightarrow X$ is also learnt by cycle consistency loss to enforce $F(G(X)) \approx X$ and vice versa, as shown in the figure 1.1.

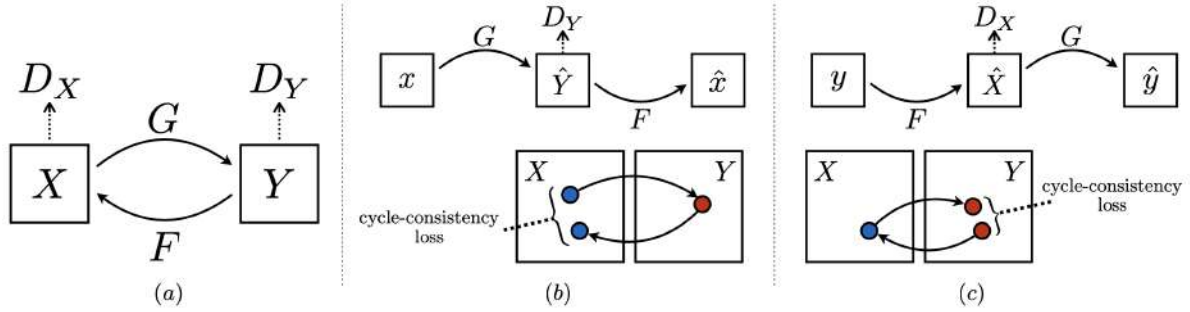

Figure 1.1: (a) CycleGAN contains two mapping functions G and F, and associated adversarial discriminators D_Y and D_X. (b)-(c) cycle-consistency loss is proposed such that: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. [1]

The proposed model CycleGAN is proved to be efficient as an unpaired image-to-image translation approach to achieve applications such as style transfer, input-space UDA and semantic segmentation.

1.2 Formulation

Given two datasets $X = \{x_i\}_{i=1}^{N}$ and $Y = \{y_j\}_{j=1}^{M}$, we denote $x_i \sim p_{data}(x)$ and $y_j \sim p_{data}(y)$ as the distributions of the two datasets. Let's say the mapping between two domains X and Y denote as $G : X \rightarrow Y$ and $F : Y \rightarrow X$. To learn the mapping functions G and F, two adversarial discriminators D_Y and D_X are jointly trained to distinguish the difference between generated images and the original image sets. To be specific, D_Y distinguishes $\{G(x)\}$ from $Y = \{y\}$ and D_X distinguishes $\{F(y)\}$ from $X = \{x\}$. Two types of losses, *adversarial loss* and *cycle consistency loss* are proposed respectively.

1.2.1 Adversarial Loss

To train the generator of the model G learns the mapping from the distribution X to Y, the adversarial loss is applied:

$$\mathscr{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))],$$

in which G tries to generate images $G(x)$ that similar to those from the domain Y, and D_Y tries to distinguish those $G(x)$ generated from the real images from the domain Y. Therefore, G aims to minimize the objective against an adversary D which tries to maximize it, i.e., $\min_G \max_{D_Y} \mathscr{L}_{GAN}(G, D_Y, X, Y)$. Similar to the mapping function F:

$$\mathscr{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))],$$

in which F tries to generate images $F(y)$ that similar to those from the domain X, and D_X tries to distinguish those $F(y)$ generated from the real images from the domain X. Theoretically, adversarial training is able to learn mappings G and F that produce outputs identically distributed as target domains Y and X, respectively.

1.2.2 Cycle Consistency Loss

As the author discussed in the paper, only with the adversarial loss, a network may map the same set of input images to any random permutation of images in the target domain, which means it cannot guarantee the learned mapping function can map an individual input x_i to a desired output y_i. Therefore, cycle consistency loss is proposed. Briefly speaking, as shown in figure 1.1 (b) and (c), $F(G(x)) \approx x$ and $G(F(y)) \approx y$. The cycle consistency loss:

$$\mathscr{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[|F(G(x)) - x|_1] + \mathbb{E}_{y \sim p_{data}(y)}[|G(F(y)) - y|_1].$$

The author suggests using L1 norm didn't improve the performance, and the experiments performed by author shows the behavior induced by the cycle consistency loss in the figure 1.2.

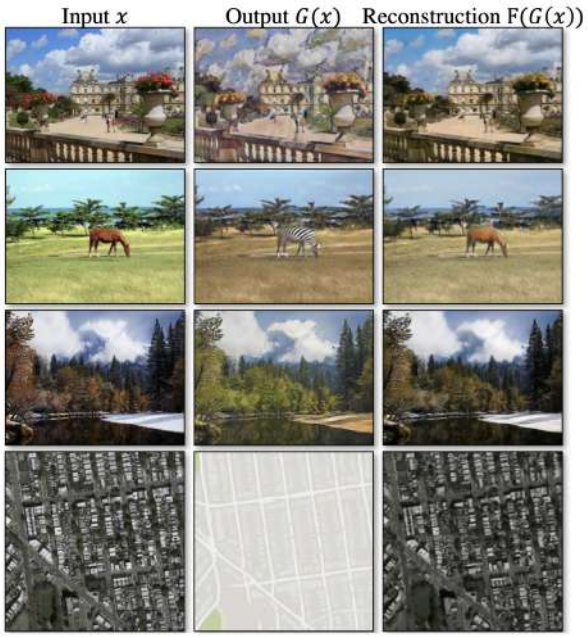

Figure 1.2: The behavior induced by the cycle consistency loss.

1.2.3 Full Objective

The full objective is:

$$\mathscr{L}(G,F,D_X,D_Y) = \mathscr{L}_{GAN}(G,D_Y,X,Y) + \mathscr{L}_{GAN}(F,D_X,Y,X) + \lambda \mathscr{L}_{cyc}(G,F).$$

In the equation, λ controls the relative importance of the adversarial loss and cycle consistency loss. Then the problem is to solve:

$$G^*, F^* = \arg\min_{G,F} \max_{D_X,D_Y} \mathscr{L}(G,F,D_X,D_Y).$$

The CycleGAN model can be regard as training two "auto-encoders" with special internal structures which map an image to itself via an intermediate representation. Besides, the author suggests that both objectives play critical roles in arriving at high-quality results empirically.

1.3 Applications

CycleGAN provides an unpaired image-to-image translation approach to transfer images from one distribution to the other without utilizing one to one correspondence between two training sets. The main application of CycleGAN is style transfer, the figure 5.1 shows the style transfer experiments conducted by the author. Actually, in our project, we try to use CycleGAN to achieve semantic segmentation and input-space UDA, which may also regard as special case of style transfer.

Figure 1.3: Typical application of CycleGAN: style transfer examples from real image input to the artistic styles of Monet, Van Gogh, Cezanne, and Ukiyo-e.

Chapter 2

Implementation

2.1 Dataset

In our project, GTA5 dataset [2] is downloaded from the website. Due to the constraints in GPU resources we can access, we only use 20% of the whole dataset, part 1 and part 2, as our training sets. There are totally 5,000 images and corresponding labels in the training set. Some examples can be found in the figure 2.1.

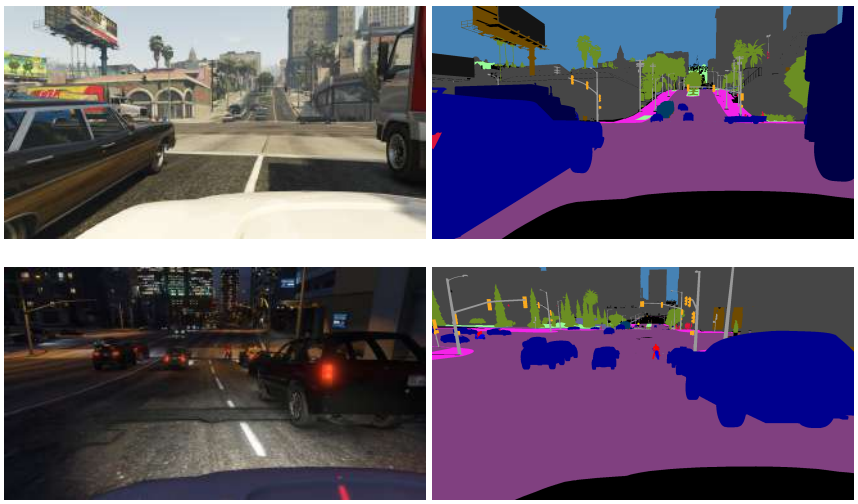

Figure 2.1: GAT5 dataset examples from the website.

Besides, Cityscapes dataset [3] is also downloaded from the website as our test set, which contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities with high quality pixel-level annotations of 5,000 frames. Some examples can be found in the figure 2.2.

2.2 Network Architecture

Similar to the original implementation, the generative networks are from the implementation of Johnson et al. [4]. The network contains three convolutions, several residual blocks, two fractionally-strided convolutions with stride $\frac{1}{2}$ and one convolution mapping features to RGB channels. 6 blocks are used for images with size 128×128 and 9 blocks for images with size equal or over 256×256. Instance normalization is also adopted. For the discriminator networks, 70×70 PatchGANs are adopted to classify whether 70×70 overlapping image patches are real or fake. According to the author, the patch-level discriminator architecture has fewer parameters, and is able to work on arbitrarily-sized images in a fully convolutional fashion.

Figure 2.2: Cityscapes dataset examples from the website.

Specifically, here are some network details at the code implementation level. Let **c7s1 − k** denotes a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1, **dk** denotes a 3×3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding was used to reduce artifacts. **Rk** denotes a residual block that contains two 3×3 convolutional layers with the same number of filters on both layer. **uk** denotes a 3×3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters and stride $\frac{1}{2}$. Then the architecture of network with 6 residual blocks consists of: **c7s1 − 64**, **d128**, **d256**, **R256**, **R256**, **R256** $\times$ 6, **u128**, **u64**, **c7s1 − 3**. the architecture of network with 9 residual blocks consists of: **c7s1 − 64**, **d128**, **d256**, **R256**, **R256**, **R256** $\times$ 9, **u128**, **u64**, **c7s1 − 3**. For the discriminators, let **Ck** denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. A convolution produces a 1-dimensional output after the last layer. And no InstanceNorm is used for the first layer of **64**. The slope of leaky ReLUs is 0.2. The architecture of discriminator consists of: **C64**, **C128**, **C256**, **C512**.

2.3 Training Details

During training, a least-squares loss instead of the negative log likelihood objective is used for $\mathscr{L}_{GAN}$ because it is more stable and generating higher quality results during training. Then, the equation becomes G minimizes:

$$\mathbb{E}_{x \sim p_{data}(x)}[(1 - D(G(x)))^2],$$

and D maximizes:

$$\mathbb{E}_{y \sim p_{data}(y)}[(1 - D(y))^2] + \mathbb{E}_{x \sim p_{data}(x)}[D(G(x))^2].$$

Besides, an image buffer storing the 50 previously generated images is kept for updating the discriminators in order to reduce model oscillation.

For the formulations, λ is set to 10 in the equation:

$$\mathscr{L}(G, F, D_X, D_Y) = \mathscr{L}_{GAN}(G, D_Y, X, Y) + \mathscr{L}_{GAN}(F, D_X, Y, X) + \lambda \mathscr{L}_{cyc}(G, F).$$

Adam optimizer is selected and the batch size is set to 1. All the networks were trained from scratch with a learning rate of 0.0002. The total number of epochs is 200. The learning rate is kept constant for the first 100 epochs, and then linearly decay the rate to 0 over the next 100 epochs. Meanwhile, the objective is divided by 2 while optimizing D to slow down the rate at which D learns relative to the rate of G. Weights are initialized from a Gaussian distribution $\mathscr{N}(0, 0.02)$.

Chapter 3

Semantic Segmentation by CycleGAN

3.1 Experiments on GAT5-based CycleGAN

In our first experiments, we trained a CycleGAN model on the 20% GTA5 dataset contains 5,000 images and their corresponding semantic segmentation labels. We trained 200 epochs, and the other training configurations are discussed above.

First, to ensure the effectiveness of the CycleGAN model trained on GTA5 dataset, we output some semantic segmentation maps generated from GTA5 images, in or out of the training set, via the model to see the performance. We input some GTA5 images in the training set and the results are shown in the figure 3.2. As we can see, on the training set, the model can efficiently predict the semantic segmentation maps, which means the training process achieves convergence.

Also, we input some GTA5 images out of the training set, to be specific, they are downloaded from the website part 3, and the results are shown in the figure 3.3. As we can see, on the validation set, the model can also efficiently predict the semantic segmentation maps for those images unseen before, which means the trained CycleGAN performs well on validation set.

Then, we test the model on the dataset Cityscapes to see the performance. The figure 3.4 shows some example results. However, different from the performance on GTA5 validation set, the trained model sometimes cannot distinguish the people in the images, and the semantic segmentation maps are full of black patches. In short, the generalization ability of the model is not very good when test on the unseen Cityscapes dataset. That is mainly because of the domain gap between GTA5 dataset and Cityscapes dataset. By observing the two sets, the images from Cityscapes dataset have very different style compared to GTA5.

3.2 CycleGAN Constraints

As we can see from the test results on Cityscapes, one of the major constraints of CycleGAN is the limit of the generalization ability. For each specific task, a particular CycleGAN should be trained on the given dataset. However, if there is a domain gap between the training set and the testing set, then the performance of the model is significantly reduced. Also, during the training process, we also find there is no obvious index to measure whether the model converges, we can only output the generated images to judge the performance of the model subjectively.

Besides, another major constraints of CycleGAN to achieve semantic segmentation is that CycleGAN cannot really understand the semantic information in the training images, it only learns the whole distribution of image set. For an individual image, CycleGAN uses cycle loss to limit the semantic segmentation map generated, so that the map can be translated back to be closer to the original images. However, the situation shown in the figure 3.1 may occur: we hope $G(x) = a$, but b also follows the distribution of semantic segmentation style. Then $G(x) = b$ and $F(b) = x$ may occur because the discriminators cannot distinguish. Also, G and F will cover up for each other during the training process. Then, the content of images may change with the style transfer. That's why as we can see from the

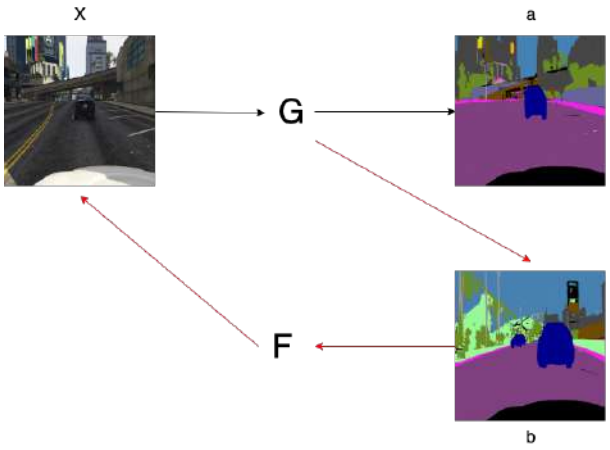

Figure 3.1: We hope $G(x) = a$, but the situation that $G(x) = b$ and $F(b) = x$ may occur according to the cycle loss.

results obtained in the figure 3.2 and 3.3, some grass in the background are wrongly translated to trees, and buildings are wrongly translated to trees and grass.

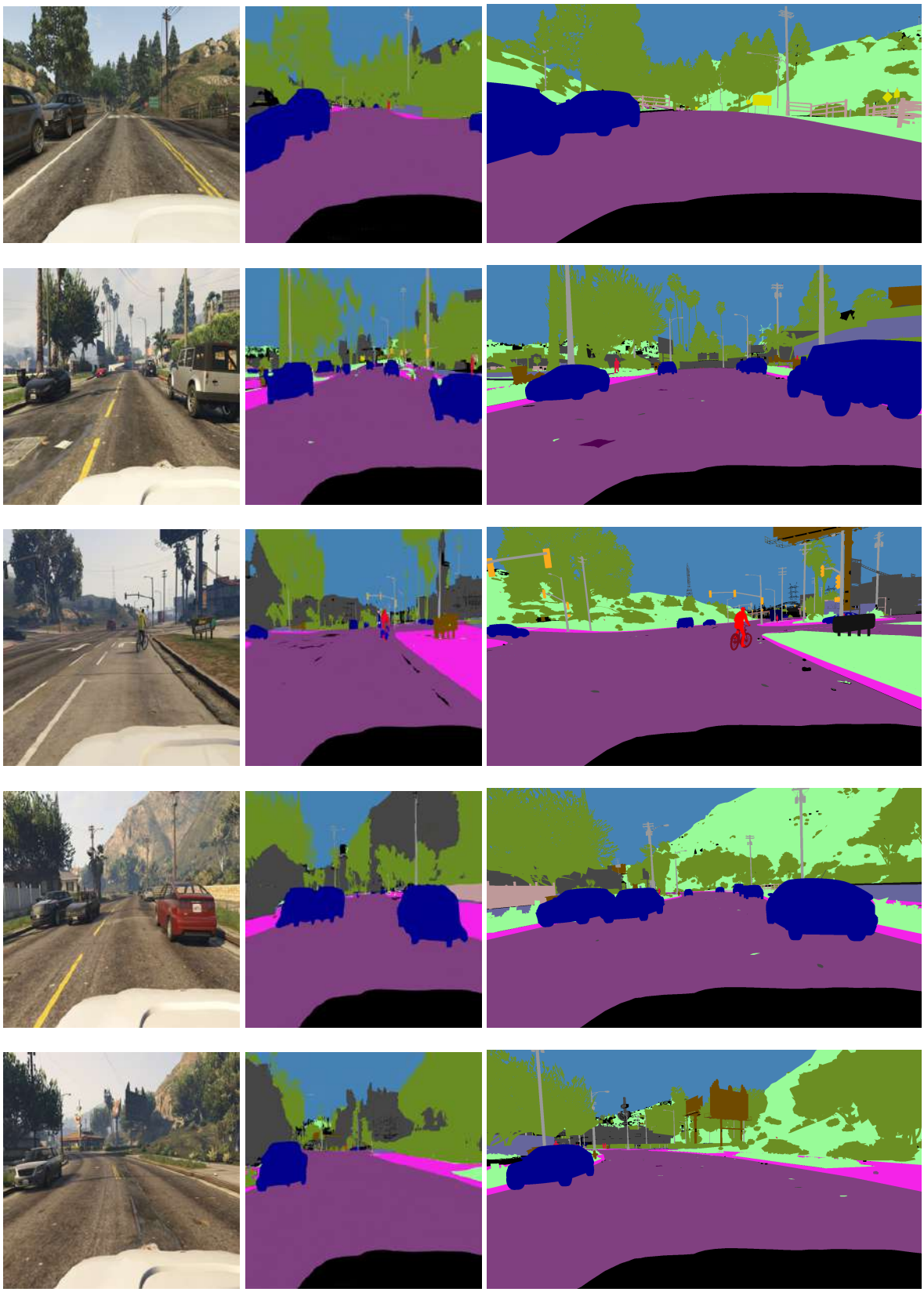

Figure 3.2: Test results on GTA5 training set to see whether the model achieves convergence. Original test images (left), test results (middle) and ground truth (right).

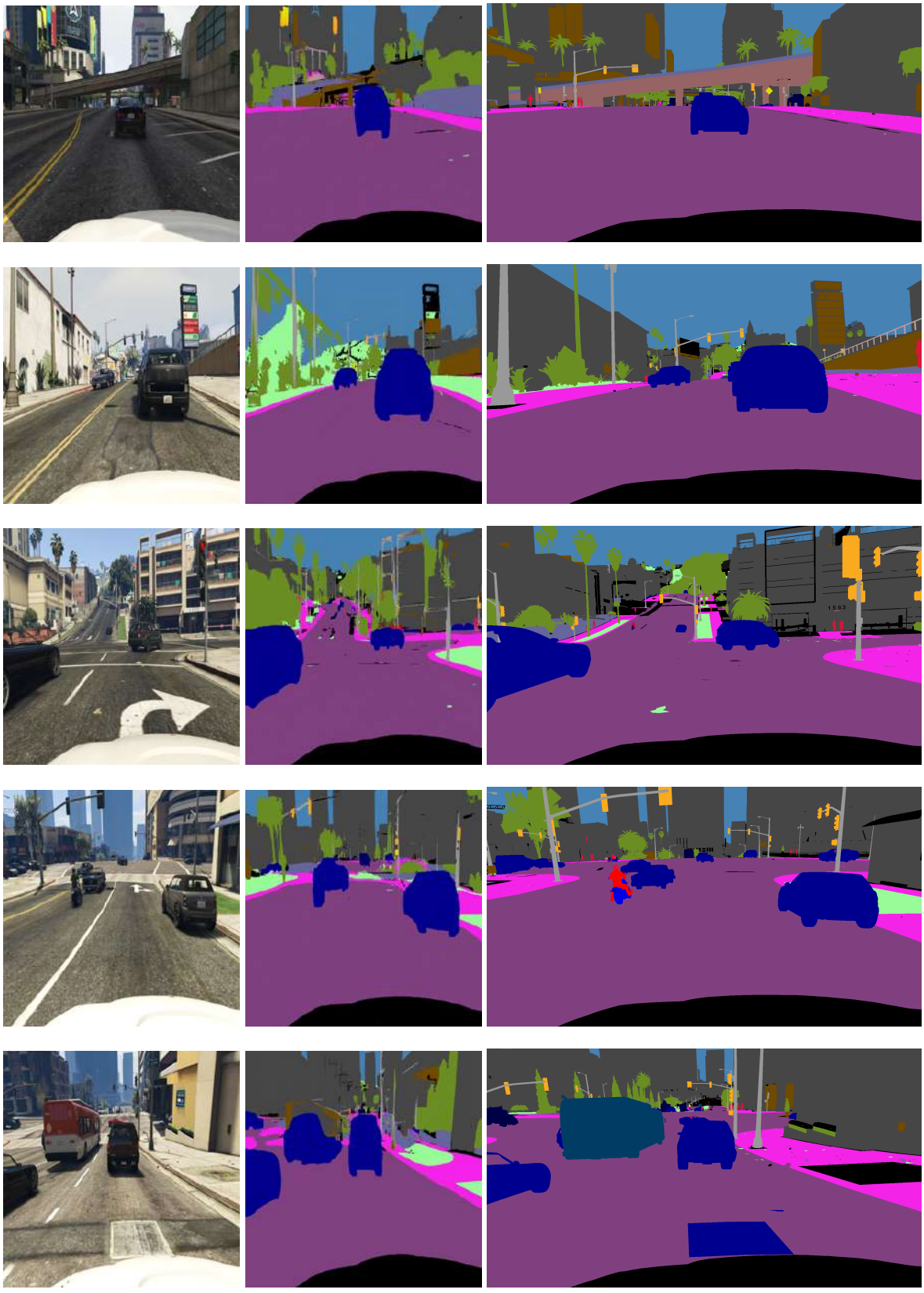

Figure 3.3: Test results on GTA5 validation set to see whether the model performance. Those images are from GTA5 but out of training set. Original test images (left), test results (middle) and ground truth (right).

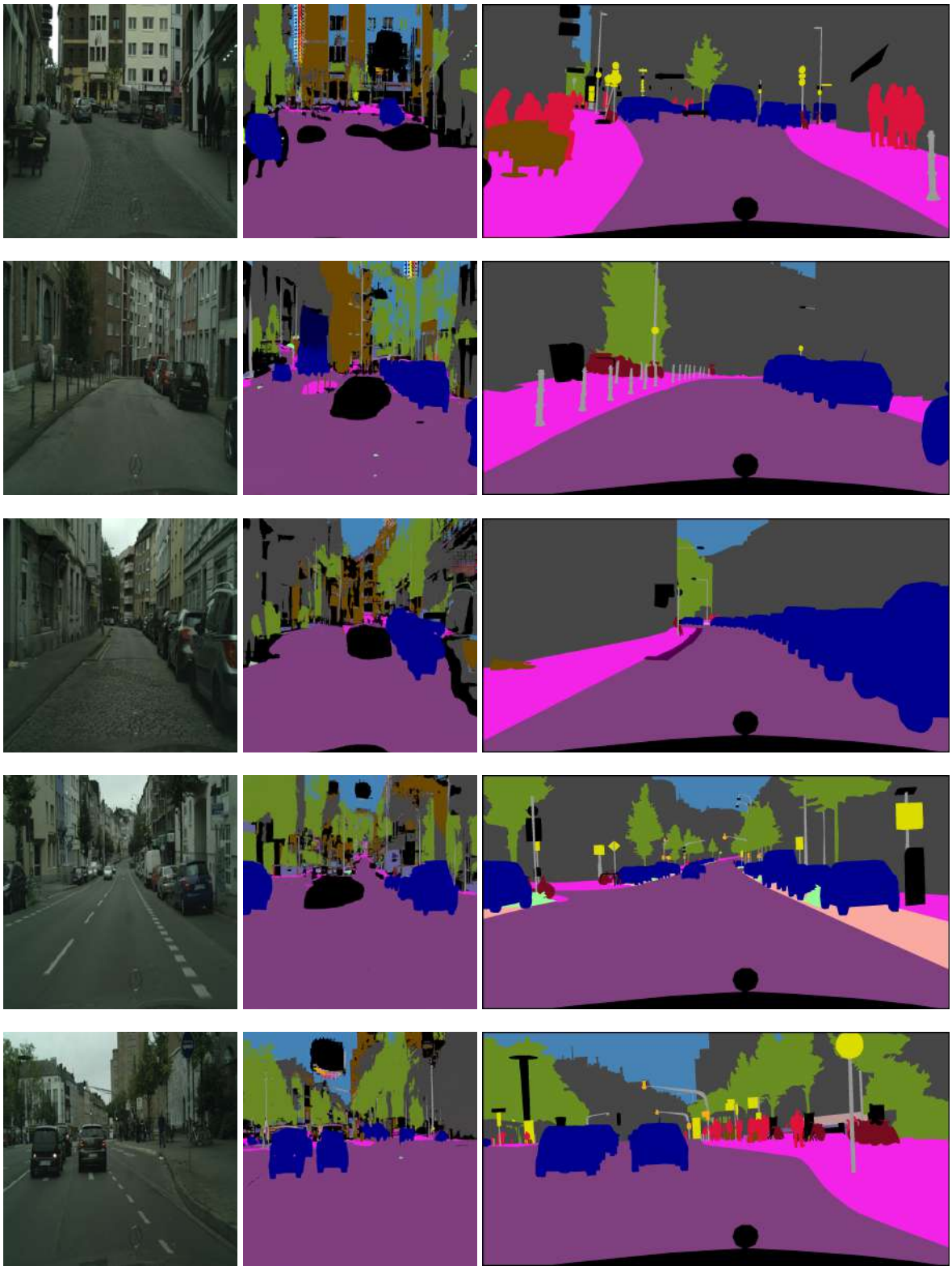

Figure 3.4: Test results on Cityscapes dataset, CycleGAN trained on 20% GTA5 dataset with 200 epochs. Original test images (left), test results (middle) and ground truth (right).

Chapter 4

Input-Space Unsupervised Domain Adaptation

In this chapter, we try to use input-space UDA to align the GTA5 domain and the Cityscapes domain to see whether the performance can be improved. Two input-space UDA methods are adopted, one is just using CycleGAN to achieve style transfer between GTA5 and Cityscapes datasets, another is that we use GTA5 dataset translated to Cityscapes style via CyCADA model [5].

4.1 Input-space UDA via CycleGAN

We trained an input-space UDA model via CycleGAN on the GTA5 and Cityscapes datasets. We set the two training sets as 20% GTA5 dataset with 5,000 images and Cityscapes dataset with 5,000 images and learn the mapping between these two sets. We trained 200 epochs, and the other training configurations are discussed above without any change. We directly input GTA5 images in the training set to the trained model to generate the translated GTA5 images with Cityscapes style. The figure **??** shows some example results.

As we can see from the results, the translated GTA5 images now have Cityscapes style, in particular, there is a noticeable change in the tone of the images. Besides, some contents of the images are also different, especially those in the background. There are more buildings and green belts in the translated images because of those frequent occurrences in Cityscapes. Besides, pedestrians appeared out of thin air on the streets inn some translated images. It proves that CycleGAN may change the content of images with style transfer which we discussed before, because the only cycle loss cannot strictly constraint the content information.

In summary, by CycleGAN, we can achieve input-space UDA to a certain extent. But due to the changed content of images, it may not perform well enough when we use translated images to train the CycleGAN for semantic segmentation task on Cityscapes. However, we can translate images between GTA5 and Cityscapes styles freely by using the trained input-space UDA CycleGAN model.

4.2 Input-space UDA via CyCADA

By comparison, we also download the GTA5 images which are translated to Cityscapes style by CyCADA proposed by Hoffman et al. [5]. We pick five same example images from the figure 4.1 to show the difference, the results are shown in the figure 4.2. As we can see from the images, different from CycleGAN UDA model, CyCADA UDA model doesn't change the content of the images and only changes the color style.

In the following experiments, we will compare the performance of the models trained on the translated GTA5 images generated by CycleGAN UDA model, and CyCADA UDA model, respectively.

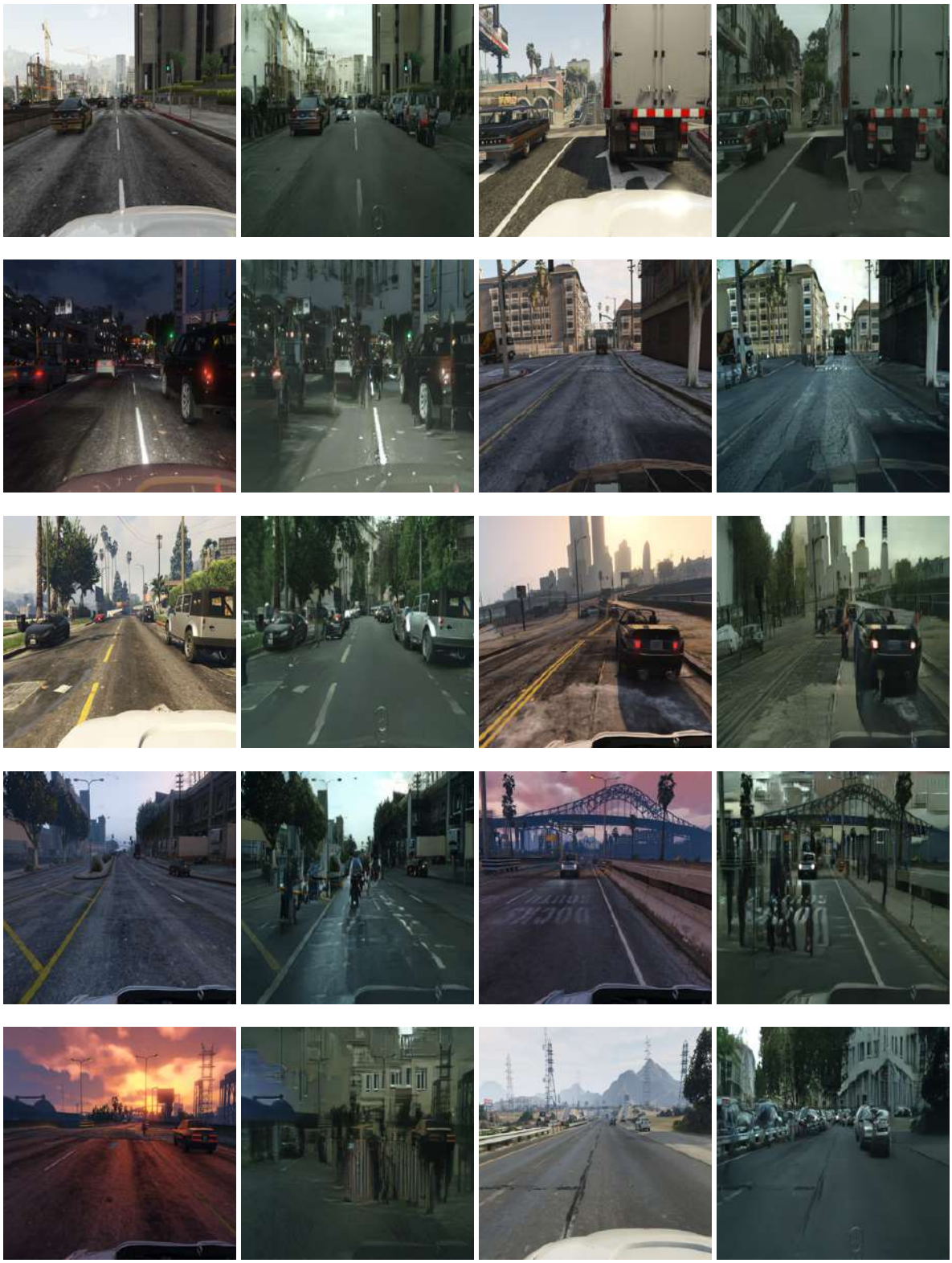

Figure 4.1: Test results of input-space UDA via CycleGAN learned mapping between GTA5 and Cityscapes datasets. Original GTA5 images (left) and translated GTA5 images with Cityscapes style (right).

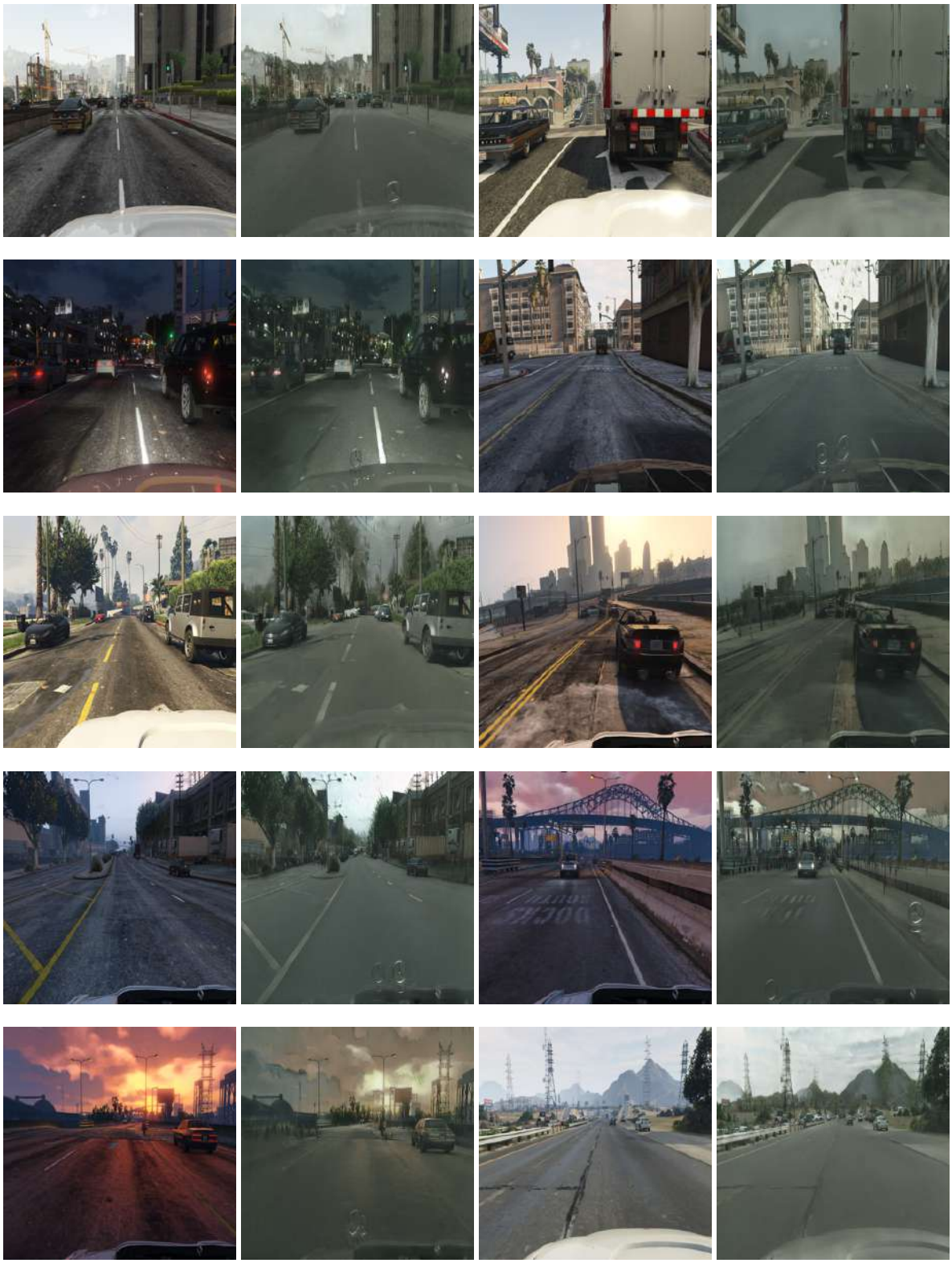

Figure 4.2: Test results of input-space UDA via CyCADA learned mapping between GTA5 and Cityscapes datasets. Original GTA5 images (left) and translated GTA5 images with Cityscapes style (right) by CyCADA.

Chapter 5

Results after Input-space UDA

In this chapter, we discuss the results obtained after introducing input-space unsupervised domain adaptation. Three experiments are proposed to compare the different effects:

- Style transfer on test set. We have a semantic segmentation CycleGAN model trained on GTA5 and a UDA CycleGAN model trained to achieve style transfer between GTA5 and Cityscapes. So, we directly transfer Cityscapes dataset to GTA5 style first by UDA model, and then generate semantic segmentatio maps by the trained CycleGAN model.
- Input-space UDA via CycleGAN. We use the trained UDA CycleGAN model to generate GTA5 dataset with Cityscapes style, and then use this translated dataset to train a semantic segmentation CycleGAN model and test on Cityscapes dataset.
- Input-space UDA via CyCADA. We download the GTA5 images with Cityscapes style generated by CyCADA model, and then use this translated dataset to train a semantic segmentation CycleGAN model and test on Cityscapes dataset.

In the following part, we compare the performance of each approach and discuss the mechanism behind.

5.1 Style Transfer on Test Set

First, we directly transfer the style of Cityscapes test set to GTA5 style by using trained input-space UDA CycleGAN model. We know that CycleGAN is able to achieve style transfer between the two distributions forward and backward: from GTA5 to Cityscapes, and also Cityscapes to GTA5. And then, we directly test on the transfered images by the traied semantic segmentation CycleGAN model. The figure 5.1 shows the results.

As we can see from the results, from Cityscapes to GTA5 style, the contents of the images are already changed: some cars on the street are disappeared, and the background areas are significantly changed from buildings to sky. Due to these changes, the semantic segmentation maps generated are even worse than those without style transfer process. For those cars on the street, because after style transferring, some cars are disappeared, so that in the semantic segmentation maps the cars are also unrecognized. For the background, more areas are wrongly recognized as sky. Therefore, by directly transferring styles of the testing set leads to changed contents of images, and then the semantic segmentation maps generated become even worse.

5.2 Input-space UDA via CycleGAN

In this part, we first trained a CycleGAN to achieve input-space UDA between GTA5 and Cityscapes datasets. Then, a CycleGAN model to do semantic segmentation task is trained on the translated GTA5

images with Cityscapes style generated by the input-space UDA CycleGAN model. We test the semantic segmentation CycleGAN model on Cityscapes dataset, and the figure 5.2 shows the results.

As we can see, after input-space UDA, the model performs much better than which without UDA because the meaningless black patches are disappeared and those cars on the street are well recognized. The generated semantic segmentation maps have less noises compared to those without UDA, and much closer to the groundtruth. However, due to the content gaps between Cityscapes and GTA5, especially the content difference in the background, the CycleGAN trained on the translated GTA5 dataset tends to recognize background as sky instead of buildings. Therefore, the performance of the model on the background is not ideal, but to recognize the typical objects on the street, such as cars and roads, the performance is much better than those without UDA. Meanwhile, compared with the test results by directly transferring style on the test images, the performance of the trained model is also much better because the objects are not disappeared in the test images.

5.3 Input-space UDA via CyCADA

For the last experiment, we directly utilize the trained CyCADA model to achieve input-space UDA between GTA5 and Cityscapes datasets. We dowloaded the translated GTA5 images by CyCADA model on the website. Then, a CycleGAN model to do semantic segmentation task is trained on the translated GTA5 images with Cityscapes style downloaded. We test the semantic segmentation CycleGAN model on Cityscapes dataset, and the figure 5.3 shows the results.

As we can see, similarly, after input-space UDA, the model performs much better than which without UDA because the meaningless black patches are disappeared and those cars on the street are well recognized. The generated semantic segmentation maps have less noises compared to those without UDA, and much closer to the groundtruth. Even more, due to the content gaps between Cityscapes and GTA5 are smaller because the contents of images didn't change much via CyCADA compared to those via CycleGAN (as we can see from the figure 4.2), especially the contents in the background are much closer, the performance of the model is much better on recognizing backgrounds. At the same time, the performance of the model to recognize the typical objects on the street still much better than those without UDA.

5.4 Conclusion

As we can see from those experiments, we conclude our observations as follows:

- Without UDA, the model trained only on the GTA5 dataset purely cannot well generate the sematic segmentation maps for those Cityscapes data unseen before due to the limited generalization ability and the domain gap between the training set and the test set.
- With style transfer directly on the test set by the UDA model to align the test set to the training set, and then feed them to the CycleGAN trained only on the GTA5 dataset, the performance of this approach is still limited due to the changed contents of the test images after style transferring. It can be explained that the error is amplified by processing the test images through two models.
- With input-space UDA via CycleGAN, the model trained on the translated GTA5 dataset can performs better on Cityscapes dataset due to the smaller domain gap between the training set and the test set. However, because when applying CycleGAN to do input-space UDA, the contents of the translated GTA5 images are changed which also due to the domain gap, especially the backgrounds, the model performs worse on those areas.
- With input-space UDA via CyCADA, similarly, the model trained on the translated GTA5 dataset can performs better on Cityscapes dataset. Meanwhile, compared to input-space UDA via CycleGAN, CyCADA didn't change much contents, so the performance of the model is much better on those areas like backgrounds.

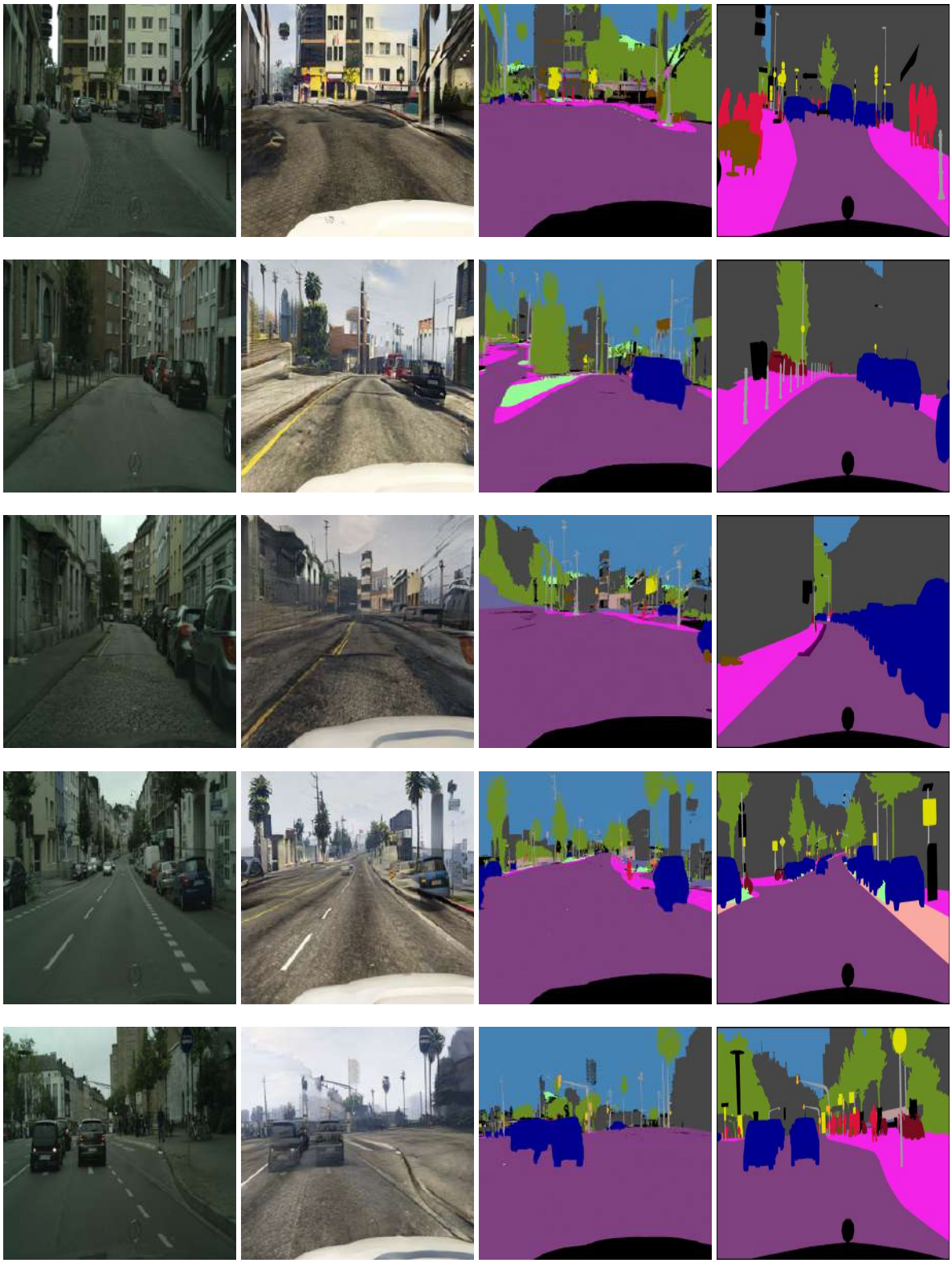

Figure 5.1: Test results on Cityscapes dataset by using the CycleGAN trained on translated GTA5 by input-space UDA via CycleGAN . Original test images (left), translated images with GTA5 style(center left), semantic segmentation maps generated by CyleGAN traied on GTA5 (center right) and ground truth (right).

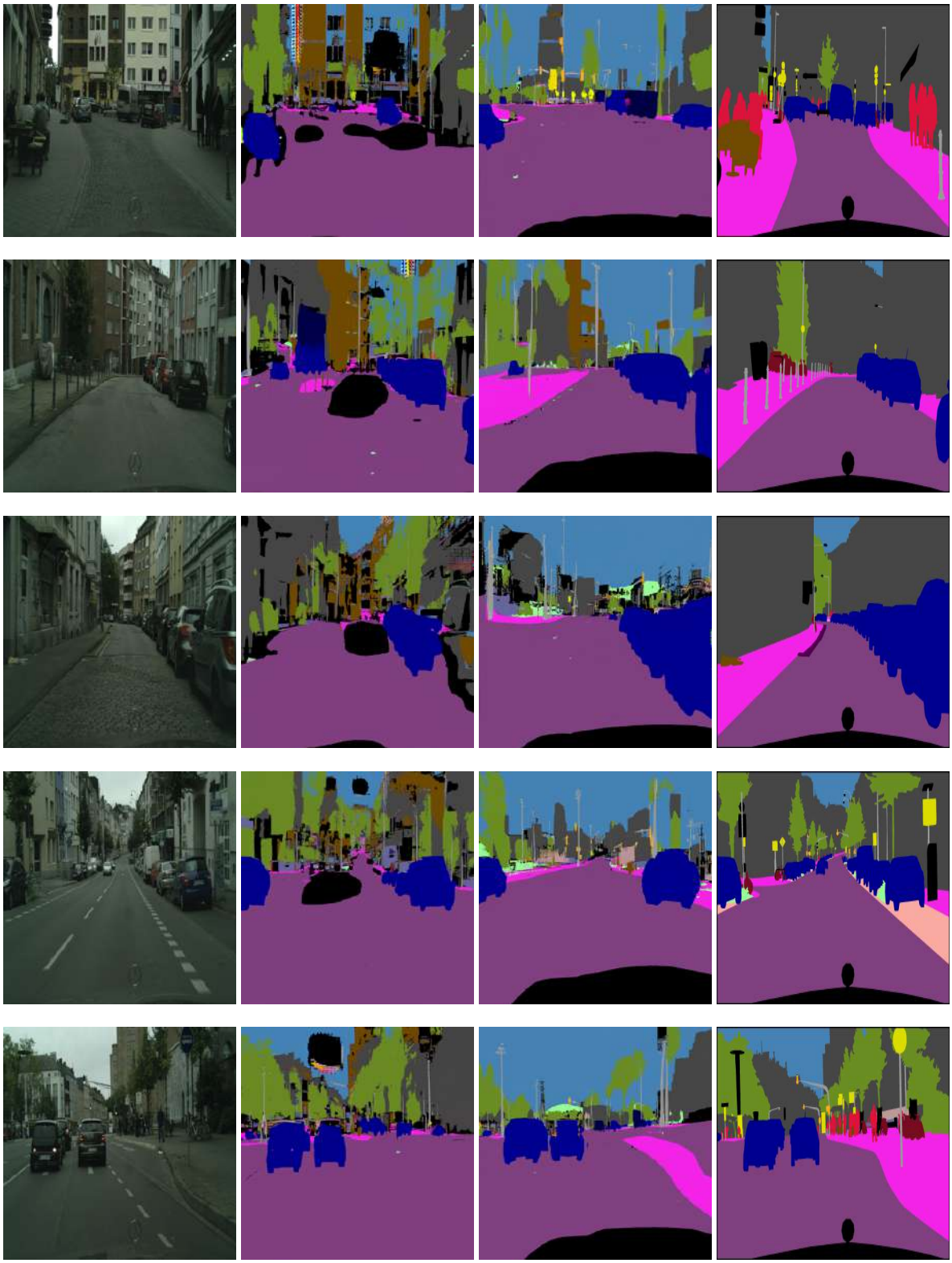

Figure 5.2: Test results on Cityscapes dataset by using the CycleGAN trained on translated GTA5 by input-space UDA via CycleGAN . Original test images (left), without UDA(center left), with UDA via CyleGAN (center right) and ground truth (right).

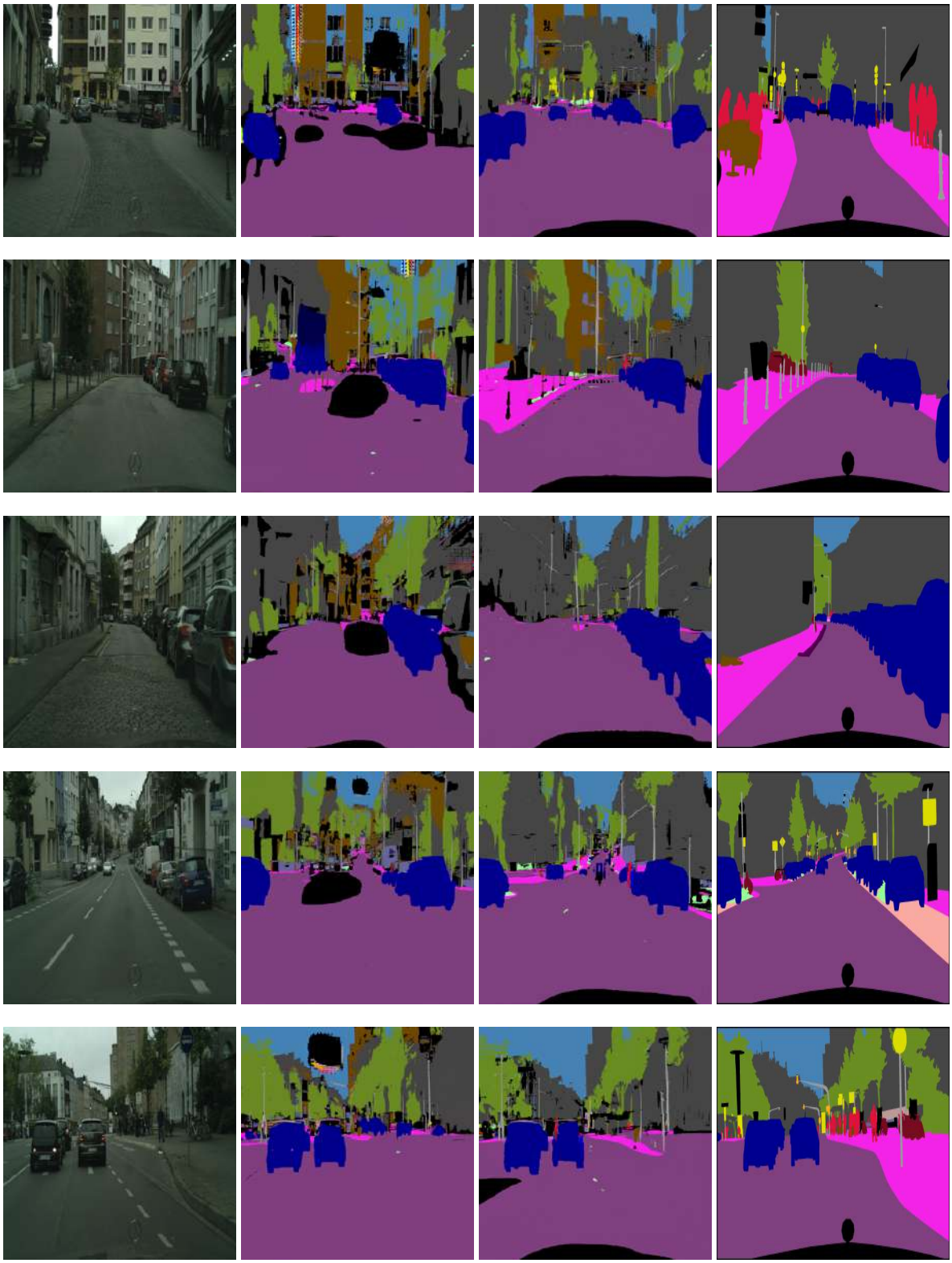

Figure 5.3: Test results on Cityscapes dataset by using the CycleGAN trained on translated GTA5 by input-space UDA via CyCADA . Original test images (left), without UDA(center left), with UDA via CyCADA (center right) and ground truth (right).

Chapter 6

Summary

In the project, we explored CycleGAN model, and use the model to achieve semantic segmentation by training on GTA5 dataset. We tested the trained model to Cityscapes dataset and discussed some constraints of CycleGAN, specifically, CycleGAN cannot really understand the semantic content of the images but only learns the mapping between image sets, and imitates to generate images that follow a certain distribution, which may lead to the changed contents. Also, the generalization ability of CycleGAN is limited so we need to use some unsupervised domain adaptation technologies to narrow the gap between training and test sets.

We also proposed some approaches to do unsupervised domain adaptation. An input-space UDA CycleGAN model is trained to learn the mapping between GTA5 and Cityscapes datasets. Firstly, we directly transfer the style of Cityscapes images to GTA5 style and feed them into the semantic segmentation CycleGAN model trained only on GTA5 model. We found due to the changed content after style transfer, the results are even worse. Then, we trained another two CycleGAN models on translated GTA5 datasets generated by the input-space UDA CycleGAN model, and the CyCADA model, respectively. We test these two models on the Cityscapes dataset and obtained better performance. Also, we discussed pros and cons of each approaches ad the mechanism behind as above.

Our conclusion is that by using unsupervised domain adaptation, the performance of CycleGAN on the task of semantic segmentation can be significantly improved.

Bibliography

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.

[2] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[3] 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, 2016.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[5] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017.

Appendix

The datasets, trained models and the testing results can be downloaded from OneDrive. If need, feel free to contact us via email yyu025@e.ntu.edu.sg. The following introduce the files in OneDrive:

File folder: datasets

- 01_images.zip: part 1 of the GTA5 dataset, contains 2,500 images from GTA5 for training.
- 02_images.zip: part 2 of the GTA5 dataset, contains 2,500 images from GTA5 for training.
- 03_images.zip: part 3 of the GTA5 dataset, contains 2,500 images from GTA5. We only use it to validate the trained CycleGAN on GTA5.
- 01_labels.zip: groundtruth of the part 1 GTA5 dataset, contains 2,500 semantic segmentation images for 01_images.zip.
- 02_labels.zip: groundtruth of the part 2 GTA5 dataset, contains 2,500 semantic segmentation images for 02_images.zip.
- 03_labels.zip: groundtruth of the part 3 GTA5 dataset, contains 2,500 semantic segmentation images for 03_images.zip.
- leftImg8bit_trainvaltest.zip: containes 5,000 images from Cityscapes dataset.
- gtFine_trainvaltest.zip: groundtruth semantic segmentation maps for leftImg8bit_trainvaltest.zip.

File folder: models. Each folder contains four models: 200_net_D_A.pth and 200_net_D_B.pth are the discriminator models; 200_net_G_A.pth and 200_net_G_B.pth are the generator models to generate from set A to B and to generate from set B to A, respectively.

- cyclegan_semantic_segmentation_no_uda: models of CycleGAN trained only on GTA5 to do semantic segmentation. Set A: GTA5; set B: semantic segmentation maps of GTA5.
- cyclegan_style_transfer_gta5_to_cityscapes: models of CycleGAN trained from GTA5 to Cityscapes to do style transfer. Set A: GTA5; set B: Cityscapes.
- cyclegan_semantic_segmentation_uda_cyclegan: models of CycleGAN trained on translated GTA5 by CycleGAN to do semantic segmentation. Set A: translated GTA5; set B: semantic segmentation maps of GTA5.
- cyclegan_semantic_segmentation_uda_cycada: models of CycleGAN trained on translated GTA5 by CyCADA to do semantic segmentation. Set A: translated GTA5; set B: semantic segmentation maps of GTA5.

File folder: results.

- cityscapes_no_uda: semantic segmentation maps of Cityscapes generated by CycleGAN trained only on GTA5.

- gta5_to_cityscapes_cyclegan: GTA5 images with Cityscapes style generated by CycleGAN trained to do style transfer between GTA5 and Cityscapes.
- cityscapes_to_gta5_cyclegan: Cityscapes images with GTA5 style generated by CycleGAN trained to do style transfer between GTA5 and Cityscapes.
- cityscapes_to_gta5_to_semantic_segmentation: semantic segmentation maps of Cityscapes generated by translating GTA5 style first and then going through the CycleGAN trained only on GTA5.
- cityscapes_uda_cyclegan: semantic segmentation maps of Cityscapes generated by the CycleGAN trained on translated GTA5 via CycleGAN.
- cityscapes_uda_cycada: semantic segmentation maps of Cityscapes generated by the CycleGAN trained on translated GTA5 via CyCADA.

Implementation steps:

We directly clone the git repository from github, the official pytorch implementation of CycleGAN. The typical process to train and test the model is shown as follows, here we take training on GTA5 as an example:

- unzip the files 01_images.zip and 02_images.zip to the folder trainA:
  unzip 01_images.zip; unzip 02_images.zip
- unzip the files 01_labels.zip and 02_labels.zip to the folder trainB:
  unzip 01_images.zip 02_images.zip
- unzip the test set:
  unzip leftImg8bit_trainvaltest.zip
  mkdir testA
  mv leftImg8bit/*/*/*.png testA/
- clone the repo:
  git clone https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
- go into the repo folder:
  cd pytorch-CycleGAN-and-pix2pix
- configure the environment:
  pip install -r requirements.txt
- put the dataset folder into the repo:
  mkdir datasets/gta5
  mv ../trainA datasets/gta5/
  mv ../trainB datasets/gta5/
  mv ../testA datasets/gta5/
- train the model:
  python train.py –dataroot ./datasets/gta5 –name gta5 –model cycle_gan –display_id -1
- test the model:
  cp ./checkpoints/gta5/latest_net_G_A.pth ./checkpoints/gta5/latest_net_G.pth
  python test.py –dataroot ./datasets/gta5/testA –name gta5 –model test –no_dropout

The trained models will be stored in ./checkpoints and the test results will be stored in ./results as described in the repository. For more details, please refer to the github repository. We trained our all models on the server GPU provided by Nanyang Technological University.