AI6121 Project: Image Translation and UDA

Group Members & Contribution:

1. Wang Yuhui (Matriculation Number: **G2202262J**)
    - Implementation and experimentation with task 1 code, result discussion part1 of the report
2. Kishore Rajasekar (Matriculation Number: **G2101949G**)
    - Implementation and experimentation with task 1 and task 2 code, discussion of constraints in report
3. Sean Goh Ann Ray (Matriculation Number: **G2202190G**)
    - Task 2 Problem formulation and methodology, CycleGAN and UDA discussion in report

Table Of Contents

1. Introduction

This report describes the procedure for 2 tasks, image to image (I2I) translation and unsupervised domain adaptation (UDA) via I2I translation.

For the first task of I2I translation, we will first describe what it is, as well as the introduction of CycleGAN, which is a widely used method to perform I2I translation. This is followed by our implementation, which uses an open source tool, and lastly our discussion of the results from the model that we trained.

For the second task of UDA via I2I translation,we will first describe what it is, followed by our implementation on 2 different models, first a source only semantic segmentation model and second a domain adaptive semantic segmentation model. Our discussion of the results from the 2 models that we trained will be the last segment of this report.

2. Image to Image (I2I) Translation with CycleGAN

I2I translation involves the transfer of styles from a target image to a source image, such as grayscale images to coloured images, images to semantic labels, and color style of one image to another. This is done by training a model with the source images as the source dataset, and the target images as the labels dataset. With the paired source-label data, the model can learn features from the paired images and translate other images from source domain to label domain.

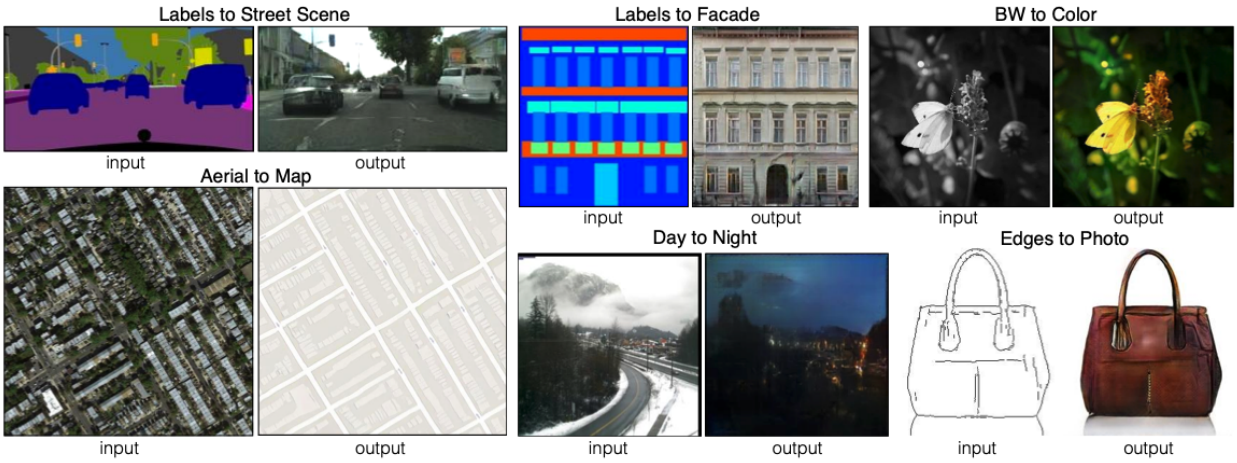

Figure 1. Examples of I2I translation (Source: [1])

For example, it is possible to train a model using semantic labels and street images (Figure 1, top left), and we can input a semantic label image (that is not part of the training dataset) to output a realistic street image.

There are 2 types of models, a discriminative and a generative. A discriminative model learns to map features to labels, thus producing a label when given a set of features. A generative model learns to map labels to features, thus producing a set of features when given a label.

Generative Adversarial Networks (GANs) pairs a generative model to a discriminative model, whereby the generative model creates realistic, fake images, from which the discriminative model learns to distinguish the real images from the fake.

The discriminator trains on real images from the training dataset and fake images produced by the generator. The discriminator then produces a discriminator loss and a generator loss, but it uses only the discriminator loss and performs backpropagation to update its weights to improve. As the discriminator trains, the generator does not.

The generator produces fake images by adding noise to the images, and uses the generator loss from the discriminator training and performs backpropagation to update its weights to improve

the ‘realness’ of the fake images it produces. As the generator trains, the discriminator trains, but the discriminator does not update its own weights.

2.1 CycleGAN

With the use of CycleGAN, these sources and labels do not require to be paired. It does so by introducing an inverse mapping, which is to say $F(G(X)) \approx X$, whereby $G(X) = Y$ translates an image from one domain to another and $F(Y) = X$ is the inverse translation. It also introduces a cycle consistency loss with each mapping, thus the CycleGAN is able to produce image translation from one domain to another with one training.

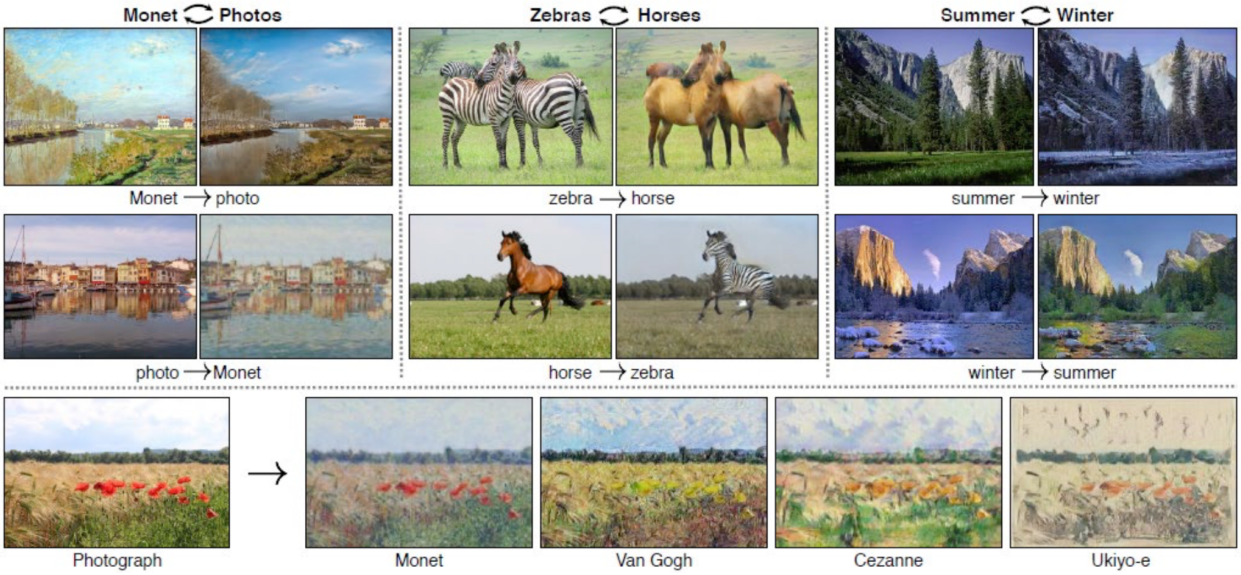

Figure 2. CycleGAN Unpaired I2I Translation Examples (Source: [2])

For example, training a CycleGAN model using unpaired images of zebras and horses (Figure 2, center) would enable it to produce both a zebra image from a horse image input, as well as a horse image from a zebra image input.

2.2 Implementation

Our implementation of CycleGAN utilizes an open source tool [3]. As stated earlier, the algorithm requires a source dataset, called trainA, and a labels dataset, called trainB, which do not need to be paired. Figure 3 shows the implementation of the I2I translation model.

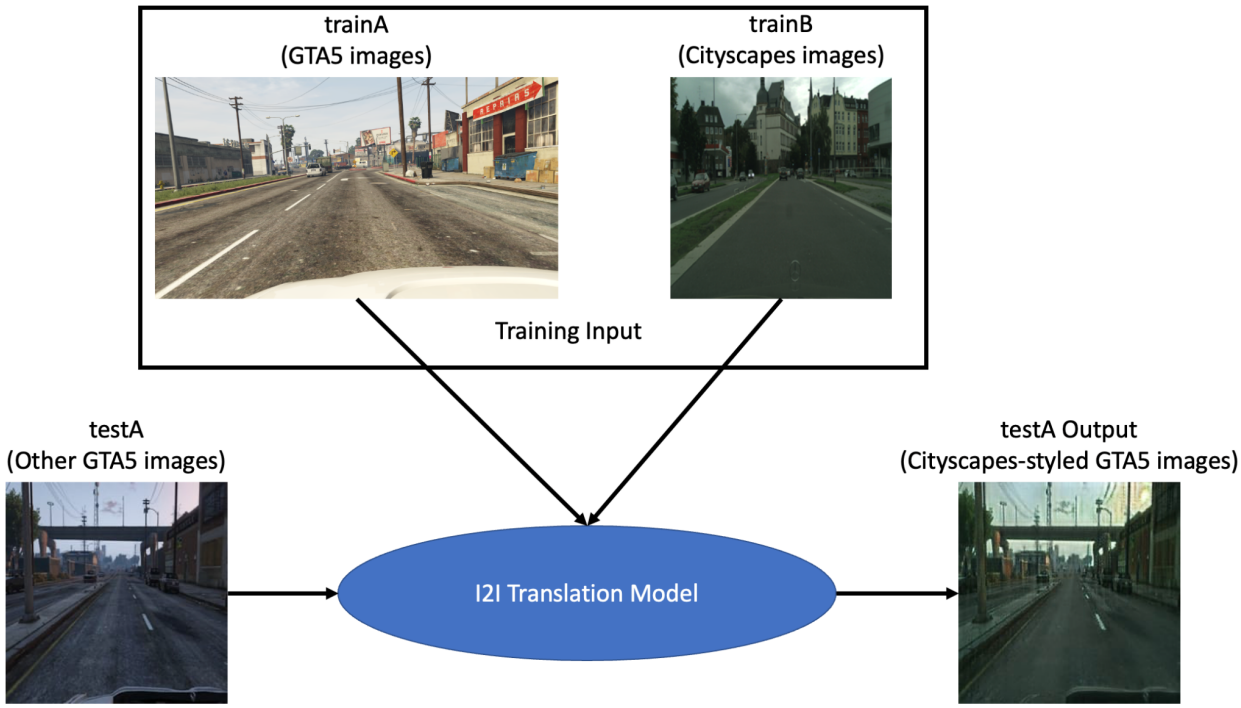

Figure 3. Implementation of I2I Translation Model

Our trainA dataset is 2000 images from GTA5 part 1 [4] and trainB dataset is 2975 images from Cityscapes leftimg8bit_trainvaltest.zip [5] which is the whole train dataset . This reduced number of training GTA5 images is due to computational restraints. Our testA dataset is 2000 images from GTA5 part 1 [4], thus producing 2000 images of Cityscape-styled GTA5 images.

We train the model for 30 epochs and each epoch takes around 30 - 35 minutes. There are two reasons for the decision of epoch number: one is still the constraints in GPU sources. Although we reduce the GTA5 training data, the training time for one epoch is still too long if we want to train more epochs. Another important reason is we observed that after 20 epochs, the change of the output of the model becomes very small. So we choose to train another 10 epochs and use the output as our final result. The batch size is setted to 1 followed by the instruction of the original codebase. The instruction states that the batch size for all experiments in the paper [2] is 1 and different batch sizes may impact the training and lead to different results.

2.3 Result Discussion

Figure 4 shows some of the good images produced by the I2I translation model, where the left column is the original GTA5 images and the right column is the Cityscapes-styled GTA5 images.

Figure 4. Good Output Images from I2I Translation Model

For all the rows in Figure 4, the GTA5 images are translated with a good cityscapes style. All the buildings, cars and trees are translated correctly based on the ground truth. When comparing with the left images and right images, there is nothing different except the overall style. The left images seem brighter than the right images. Apart from this, we cannot tell the difference between them. No objects are transferred wrongly and there is no style loss. We treat these images as good outputs.

Figure 5 shows some of the bad images produced by the I2I translation model, where the left column is the original GTA5 images and the right column is the Cityscapes-styled GTA5 images.

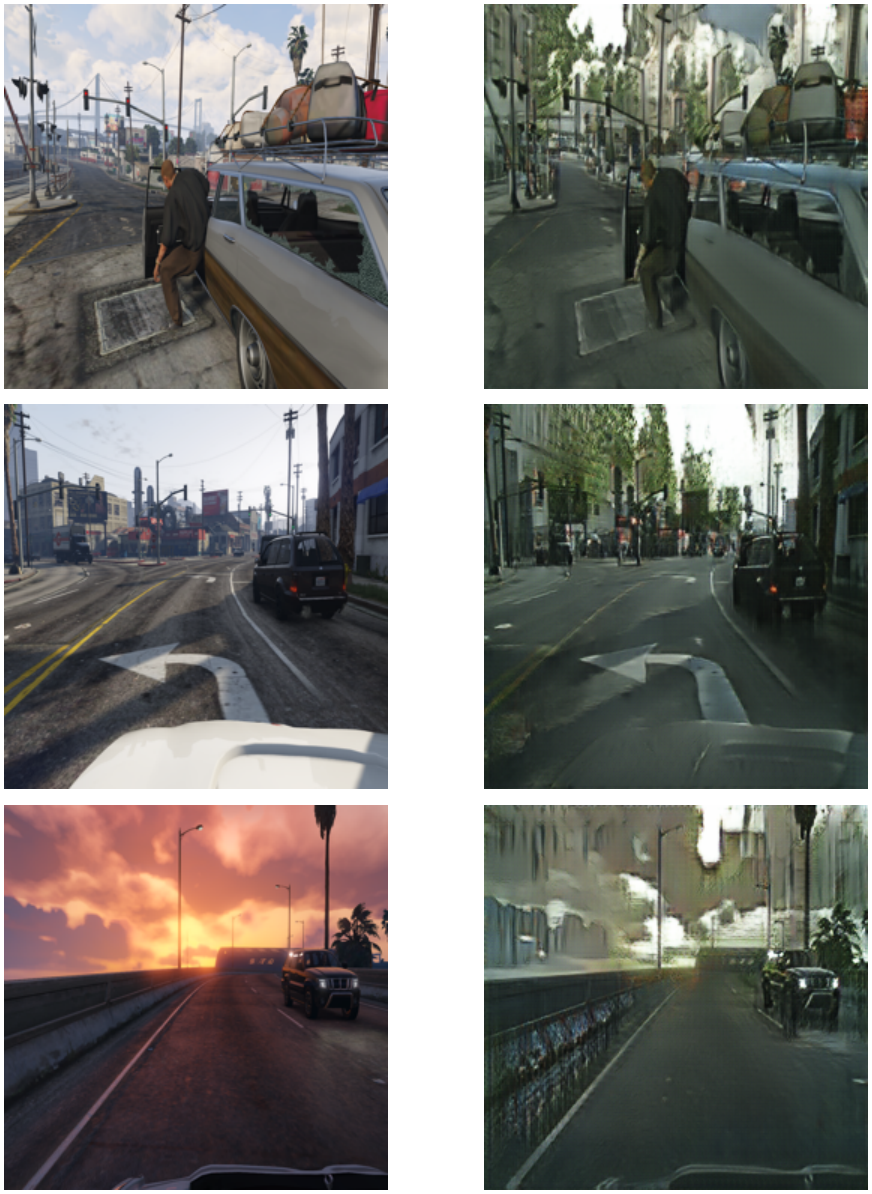

Figure 5. Bad Output Images from I2I Translation Model

These image pairs are not very satisfactory compared with the pairs in Figure 4. As we can observe from rows 1 and 2, some noise with green color is produced. These noises seem like the trees but these don't exist in the original GTA5 pictures. One possible reason is the poles and buildings are too dense so that the models lack the ability to translate them. This situation may improve with more training epochs. Also, some styles are transferred wrongly. In rows 2 and 3, the parts with sky are transferred with a strange style. In the top right corner some distortion exists. This phenomenon is even more evident in the translated image in the third row. The clouds in the sky are transferred to a terrible style. These images produce additional noise and strange styles are treated as bad outputs.

2.4 Constraints of CycleGAN

Although the CycleGAN produces promising results, it does have its constraints. For instance, this method performs very well on translating color and textures from one to another. However, there are several cases where it doesn't perform well. One such instance is when a task involves geometrical changes. In those cases, it tends to make minimal changes to the input image.[2] In our case, often the elements that appear with the background of the sky, like clouds or even buildings are not translated properly. Since there are usually trees with the background of sky in the cityscapes example, the translated images often have embedded trees in them even though the input image doesn't have trees. It is often translated to a tree or vice versa as seen in Figure 5. Bad output examples.

3. Unsupervised Domain Adaptation (UDA) via I2I Translation

3.1 UDA

UDA is a form of I2I translation which involves an input space and a label space. The samples in the input space are images of one domain, while the label space are images of another domain. In this report, the intention is to produce a semantic segmentation model of the Cityscapes dataset. The first model uses a CycleGAN model which was trained in labeled GTA5 dataset, where the source is the GTA5 images and the label is the semantic segmentation images. The second model utilizes UDA by using the Cityscapes-styled GTA5 images produced from the first task in Section 2 as the source and their original semantic segmentation images as the label.

3.2 Source-Only Semantic Segmentation Model (Task 2 Model A)

Our implementation of the source-only semantic segmentation involves a CycleGan trained as per Figure 6 below.

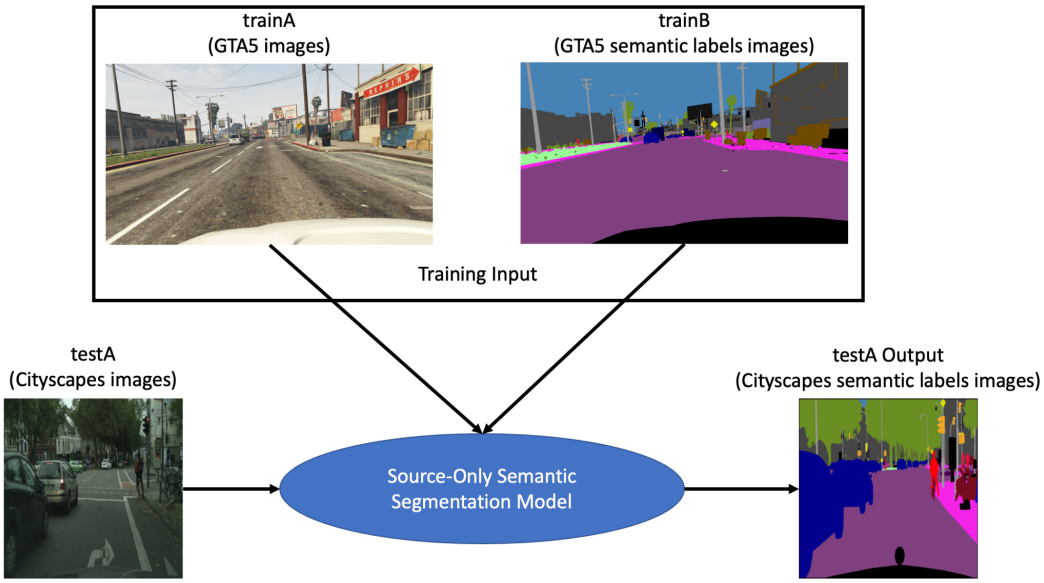

Figure 6. Implementation of Source-Only Semantic Segmentation Model

Our trainA dataset is 2000 images from GTA5 part 1 [4] and trainB dataset is 2000 semantic label images from GTA5 part 1 [4]. This reduced number of training images is due to computational restraints. Our testA dataset is 2000 images from Cityscapes leftimg8bit_trainvaltest.zip [5], which would produce 2000 semantic label images.

The model was trained for 20 epochs with a batch size of 4 and approximate training time of 9 hours.

3.3 Domain Adaptive Semantic Segmentation Model (Task 2 Model B)

Our implementation of the domain adaptive semantic segmentation model involves a CycleGAN trained as per Figure 7 below.

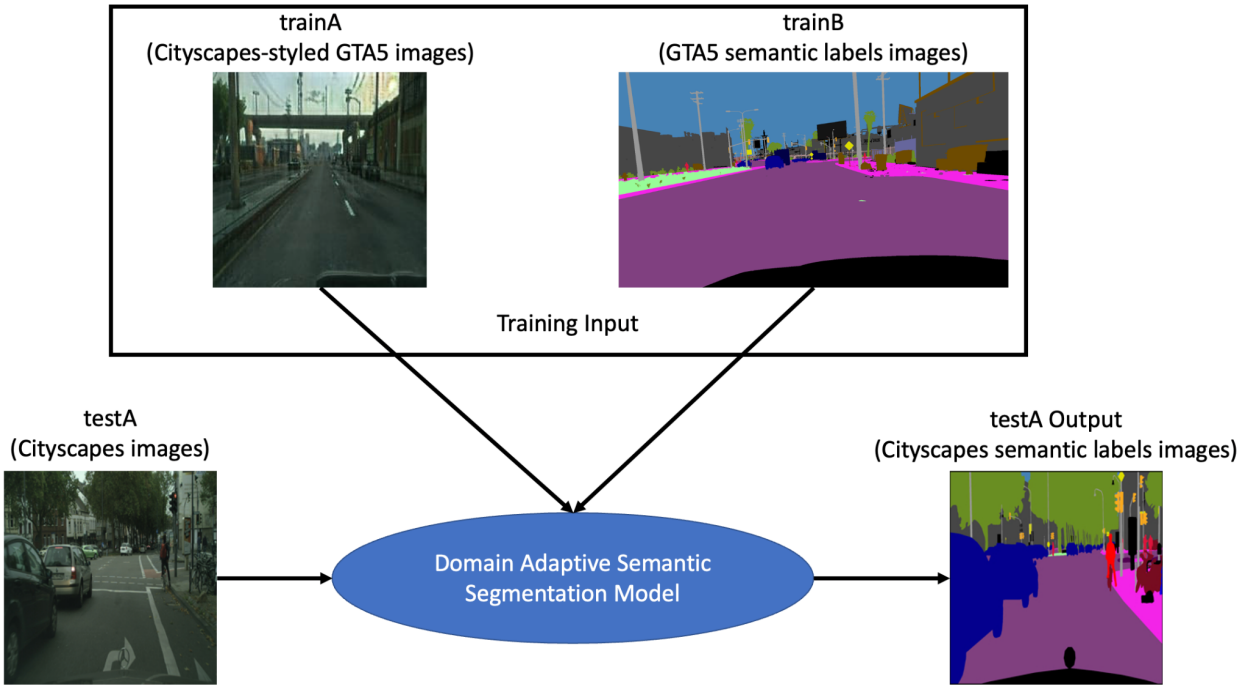

Figure 7. Implementation of Domain Adaptive Semantic Segmentation Model

Our trainA dataset is the 2000 Cityscape-styled GTA5 images produced from our I2I Translation Model in Section 2, and trainB dataset is their respective semantic label images from GTA5 part 1 [4]. This reduced number of training images is due to computational restraints. Our testA dataset is 2000 images from Cityscapes leftimg8bit_trainvaltest.zip [5], which would produce 2000 semantic label images.

The model was trained for 20 epochs with a batch size of 4 and approximate training time of 9 hours.

3.4 Result Discussion

As observed from Sections 3.2 and 3.3, the output of both models are semantic label images from the same testA images. Figure 8 shows some of the better performing images produced by our semantic segmentation models, with the first column being the original Cityscapes image, the second column being the source-only model image, the third column being the domain adaptive model image, and the last column being the ground truth.

In the first row of Figure 8, buildings in the background were correctly identified in both models. However, the source-only model managed to identify some of the pedestrians while the domain

adaptive model was unable to do so. In the second row, most of the cars were correctly identified by both models, with the source-only model performing slightly better as it also managed to identify more trees correctly than the domain adaptive model. In the last row, the domain adaptive model successfully identified some of the grass patch and cars, while the source-only model performed slightly better in identifying the trees and part of the building in the background.

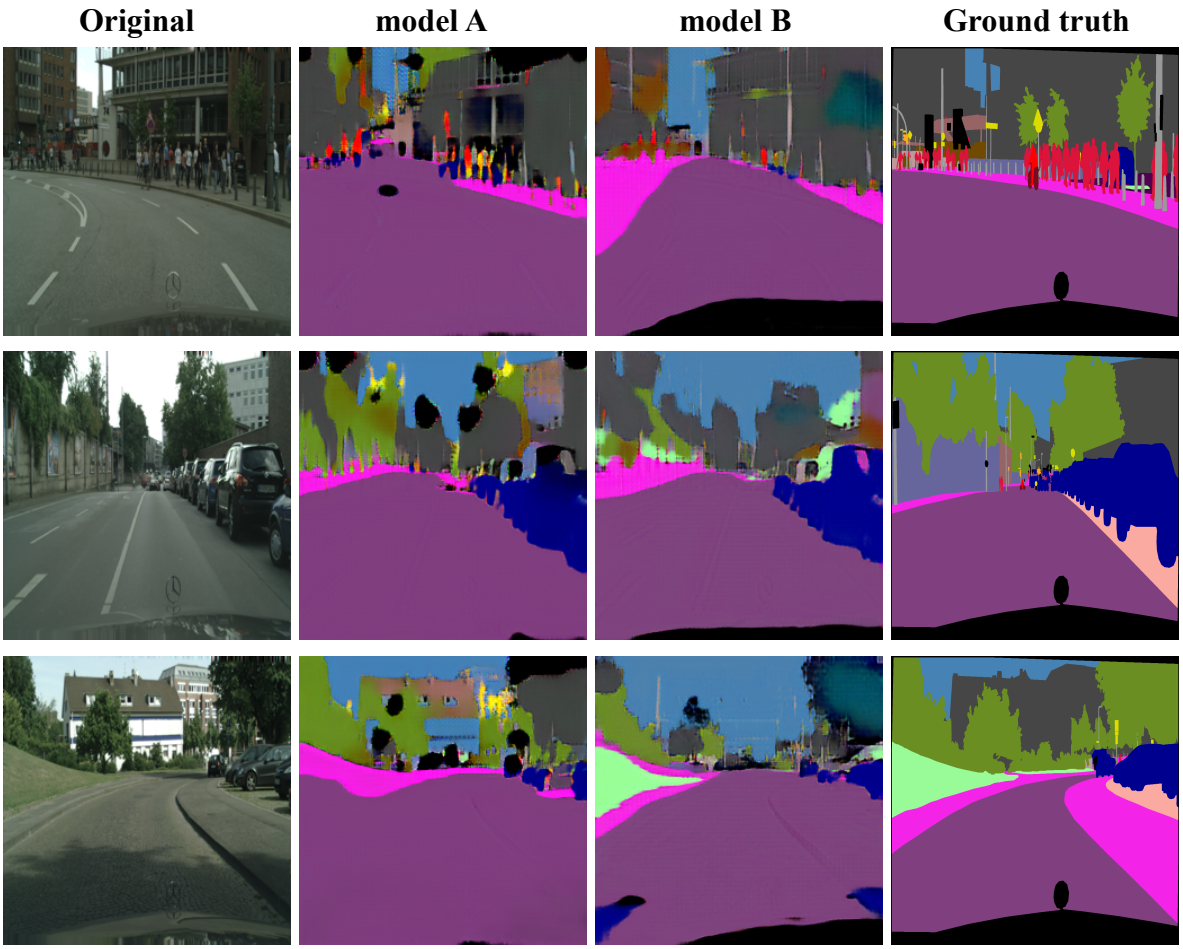

Figure 8. Good Output Images from Semantic Segmentation Models

Figure 9 shows some of the bad performing images produced by our semantic segmentation models, ordered in the same way as in Figure 8. As observed in the first row, the semantic segmentation models were unable to identify the person standing on the street. In the second row, the person on the right side of the image is also not identified, and the car on the left side is also only partially identified by the source-only model. The fourth row shows that the domain adaptive model identified the wall on the right side of the image as a vehicle and the trees as a building, while the source-only model sees the wall as part of the road but still successfully identified the tree. In the last row, the domain adaptive model performed worse than the

source-only model as it could not identify the vehicle on the left side of the image, and both models could not identify the railings on the left side of the image.

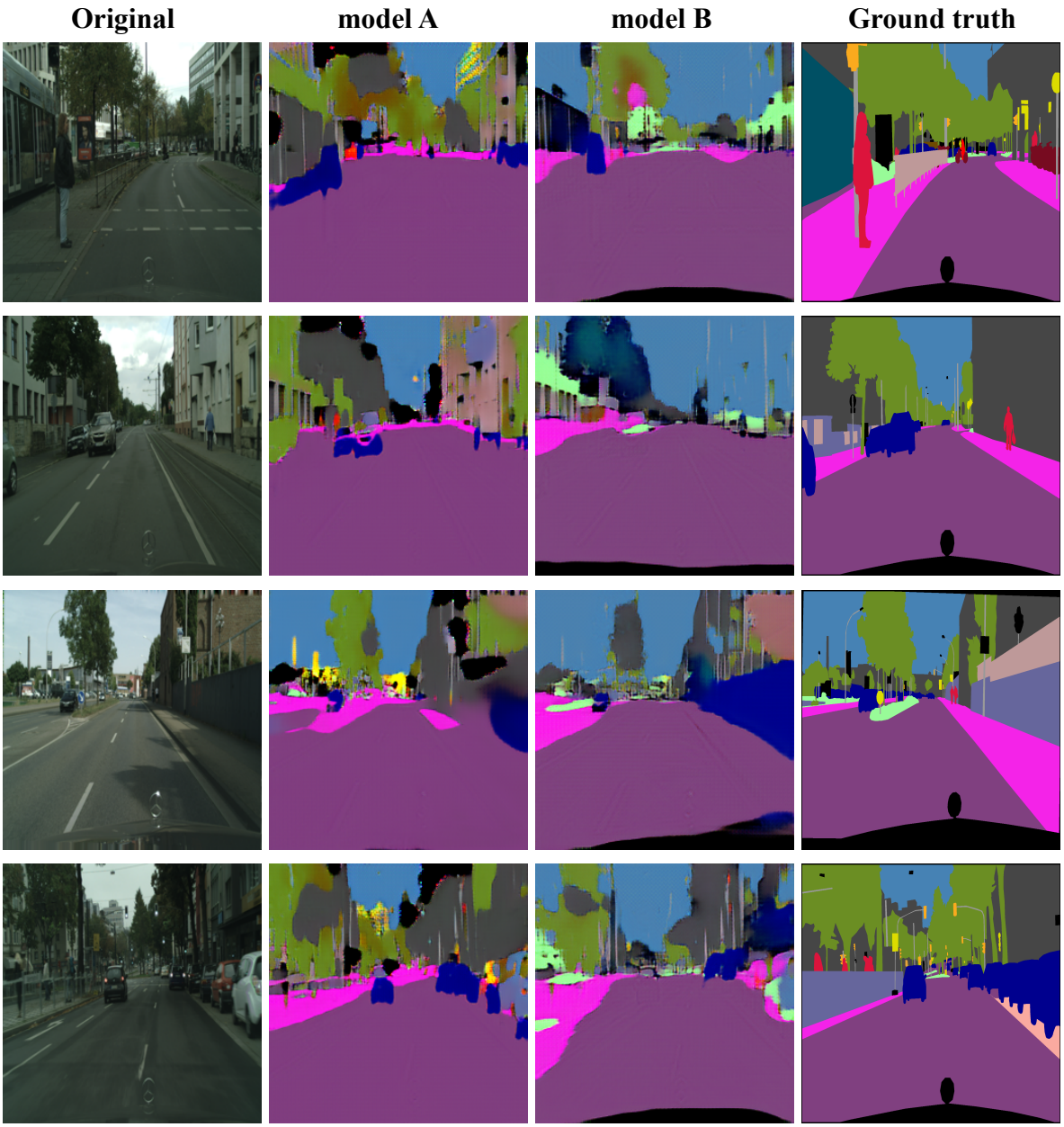

Figure 9. Bad Output Images from Semantic Segmentation Models

A large reason for the poor performances is due to limited computational power that we have access to, thus limiting the number of training images and number of epochs. Additionally, it can be observed that the domain adaptive model performed worse than the source-only model, which is not what we were expecting. However, taking into consideration that the domain adaptive model is dependent on the produced images from the I2I translation model in Section 2 for training, it can be due to a domino effect which causes its performance to be drastically reduced.

4. References

1. Isola, P. *et al.* (2017) “Image-to-image translation with conditional adversarial networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [Preprint]. Available at: https://doi.org/10.1109/cvpr.2017.632.
2. Zhu, J.-Y. *et al.* (2017) “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *2017 IEEE International Conference on Computer Vision (ICCV)* [Preprint]. Available at: https://doi.org/10.1109/iccv.2017.244.
3. Junyanz. *Junyanz/Pytorch-Cyclegan-and-pix2pix: Image-to-image translation in pytorch*, *GitHub*. Available at: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
4. *Playing for data: Ground Truth from computer games*. Available at: https://download.visinf.tu-darmstadt.de/data/from_games/
5. *Cityscapes Dataset*. Available at: https://www.cityscapes-dataset.com/downloads/

5. Appendix

5.1 I2I Translation Algorithm

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
!git clone https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
%cd pytorch-CycleGAN-and-pix2pix
```

```
!unzip ../drive/MyDrive/AI6121/AI6121_Group/GTA2cityscape.zip
```

```
!pip install -r requirements.txt
```

Training

```
!python train.py --dataroot ./datasets/GTA2cityscape --name
GTA2cityscape --model cycle_gan --display_id -1 --batch_size 4
```

pick the checkpoint file

```
!cp ./checkpoints/GTA2cityscape/latest_net_G_B.pth
./checkpoints/GTA2cityscape/latest_net_G.pth
```

testing

```
!python test.py --dataroot datasets/GTA2cityscape/trainA --name
task2model1 --model test --num_test 2000 --no_dropout
```

save results

```
!zip -r results.zip
/content/pytorch-CycleGAN-and-pix2pix/results/GTA2cityscape/test_lat
est/images/
```

5.2 Source-Only Semantic Segmentation Model Algorithm

```
!unzip /content/gdrive/MyDrive/AI6121/Project/task2model1.zip
```

Training

```
!python train.py --dataroot ./datasets/task2model1 --name
task2model1 --model cycle_gan --display_id -1 --batch_size 4
```

pick the checkpoint file

```
!cp ./checkpoints/task2model1/latest_net_G_A.pth
./checkpoints/task2model1/latest_net_G.pth
```

testing - evaluation against cityscapes

```
!python test.py --dataroot datasets/GTA2cityscape/trainA --name
task2model1 --model test --num_test 2000 --no_dropout
```

save results

```
!zip -r results.zip
/content/pytorch-CycleGAN-and-pix2pix/results/task2model1/test_lates
t/images/
```

5.3 Domain Adaptive Semantic Segmentation Model Algorithm

```
!cp -r ../drive/MyDrive/AI6121/Project/model2 ./datasets
```

Training

```
!python train.py --dataroot ./datasets/model2 --name model2 --model
cycle_gan --display_id -1 --batch_size 4
```

pick the checkpoint file

```
!cp ./checkpoints/model2/latest_net_G_A.pth
./checkpoints/model2/latest_net_G.pth
```

testing - evaluation against cityscapes

```
!python test.py --dataroot datasets/GTA2cityscape/trainA --name
model2 --model test --num_test 2000 --no_dropout
```

save results

```
!zip -r results.zip
/content/pytorch-CycleGAN-and-pix2pix/results/model2/test_latest/ima
ges/
```