Reproducing CycleGAN

Amin Fadaeinejad
York University
Toronto, Ontario, Canada
afadaei@yorku.ca

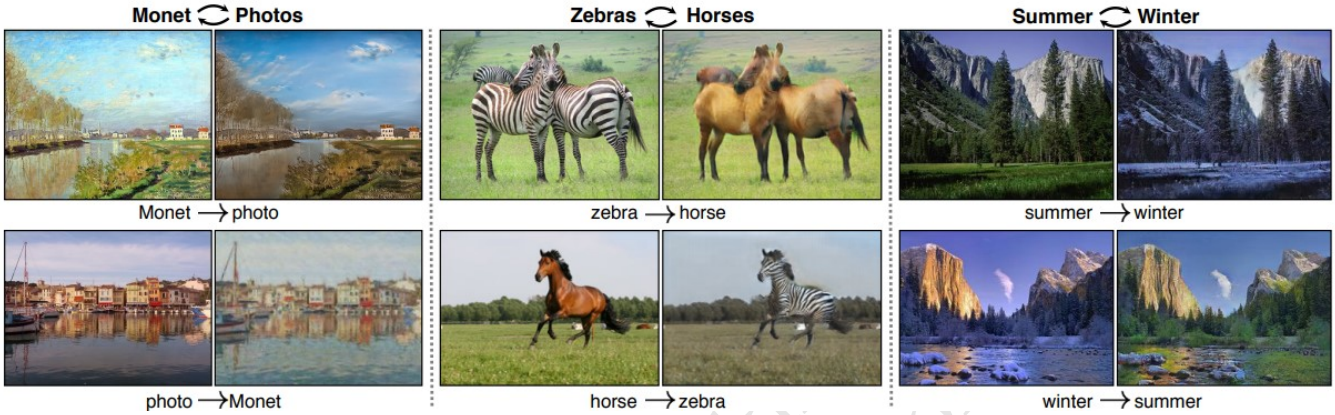

Figure 1: Sample of the CycleGAN results from the original paper

ABSTRACT

Image to image translation is one of the tasks in deep learning that has gotten a lot of attention in the deep learning and computer vision community. However in the more traditional approaches there most be labeled images for corresponding images, e.g for every image from domain A there must be an image B in the other domain. However in the approach that Jun-Yan Zhu et el [12] take they did not need the corresponding images from the two domain. In the paper they described the two different domains in various ways, in their paper the main two domains were Horse/Zebras, in other cases there were some domains such as Monet/Photo, Summer/Winter and Image/Segmentation. In our reproducing challenge we are going reproduce the Image/Segmentation domain, it is also interesting since Mondal et al[8] did use the CycleGAN for segmentation task. Image segmentation is one of the fields in deep learning that there has been a lot of work done, therefor it will be interesting to see how this method will work for it. In this report at first we are going to cover the main contribution of these paper and what they exactly did to make image translation without paired data possible. After we are going to cover the experiments that we have done by the authors. In the end we will discuss if the paper was reproducible or not.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

KEYWORDS

Generative adversarial network, Semi-Supervised Learning, Image-Translation

ACM Reference Format:
Amin Fadaeinejad. 2022. Reproducing CycleGAN . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In this paper [12] they represent a method for learning to translate an image from source domain X to a target domain Y, in the absence of of paired images. They are trying to find a transformation function G that $G : X \rightarrow Y$, in the case where $G(X)$ and Y have same distribution. They also find the inverse function F that will do the exact opposite. Function $F : Y \rightarrow X$ will find the a transformation that will convert domain Y to domain X. In addition they introduced a cycle consistency loss where this loss tries to push $F(G(X)) = X$.

1.1 Paired Data

In image to image translation tasks one important thing is the need of paired data, figure 2 show an example of paired data. How ever there are limited number of data sets that have paired data. But CycleGAN introduces a new method that could to the image-to-image translation task without paired data. So technically this is going to be great boost in the field of image-to-image translation.

1.2 Adversarial training

In this project as mentioned before we are going to do the image translation between an image and the segmentation of that image.

Figure 2: Paired data

To fully understand the entire model and how it works we are going to cover all the sections of the model. Figure 3 shows the

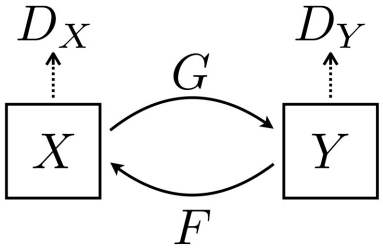

Figure 3: CycleGAN model scheme

entire model, where we indicate that domain X is the real-image, domain Y is the segmentation domain. Model G is a generator where is converts images from domain X to domain Y, in other words $G : X \rightarrow Y$. Model F is a generator where is converts images from domain Y to domain X, in other words $F : Y \rightarrow X$. There are also two discriminators where their objective is to detect if the input image is a real image of a fake. Discriminators D_X will detect if the ground image (the real image from the scene) is real or not. Discriminators D_Y will detect if the segmentation is real or not.

1.3 Cycle Consistency

Adversarial training can, in theory learn mappings G and F that produce outputs identically distributed as target domains Y and X respectively.However, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. Thus, adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To further reduce the space of possible mapping functions, we argue that the learned mapping functions should be cycle-consistent.

Figure 4 shows the concept of cycle consistency, in this (cycle consistency) approach, after applying generator G to an image, we will also apply the function F which is going to convert $G(X)$ back to domain Y. The overall goal is to make X and $F(G(X))$ as close as possible. We also the same thing in the reverse way, where the goal is to make Y and $G(F(Y))$ as close as possible.

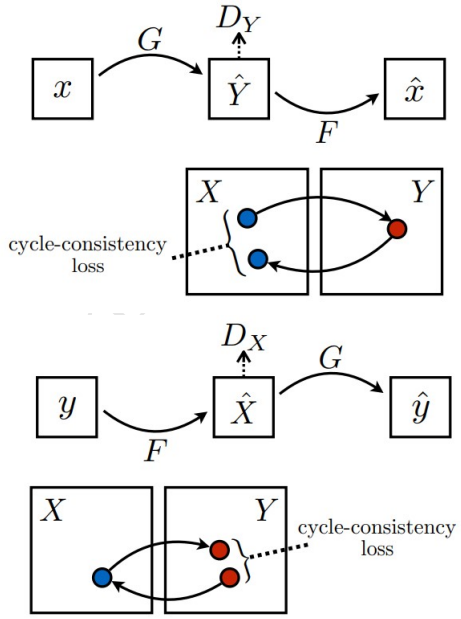

Figure 4: Cycle Consistency

1.4 Identity loss

Other than the two methods that we just mentioned there is also one additional loss that they used to train their model. If we apply generator G to an image that is in domain X we expect it to stay in the same domain, which means $G(X) = X$. The opposite should also work as well, which means if we apply generator F to an image from domain Y is should still remain in domain Y. Figure 5 shows this concept.

1.5 Full Objective

The final loss function should look something like this:

$$\mathcal{L}(G, F, D_x, D_y) = \mathcal{L}_{GAN}(G, D_y, X, Y) + \mathcal{L}_{GAN}(F, D_x, X, Y)$$

$$+\lambda_{Cycle}\mathcal{L}(G, F) + \lambda_{identity}\mathcal{L}_{identity}(G, F)$$

1.6 Data set

Depending on the two different domains that they were using there are different data set that they used. However for our task which is segmentation they used the cityscapes, which we also used the same thing in our own implementation in order to reproduce the same results.

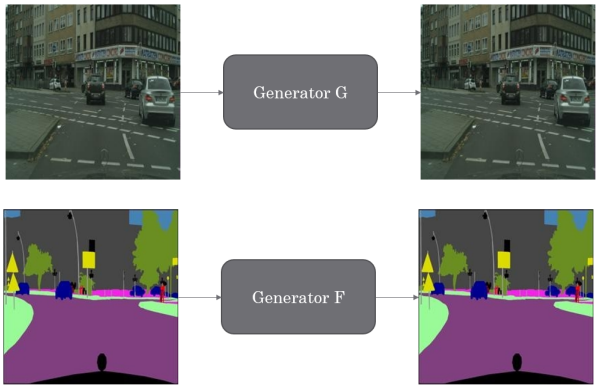

Figure 5: Identity function

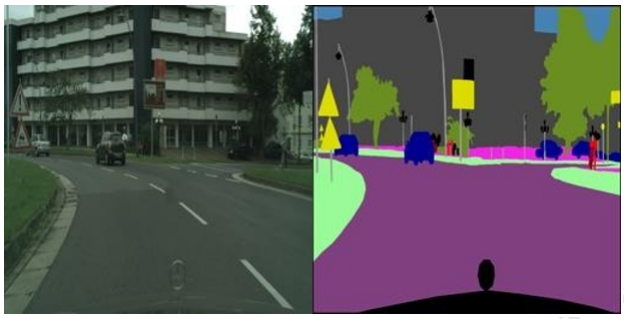

Figure 6: Sample Images of data set

1.7 Network Architecture

In thier implementation they used Johnson et al [4] model, since in Johnson et al were able to achieve impressive results for neural style transfer and super resolution. Their network contains three convolutions, several residual blocks [1] two fractionally-strided convolutions with stride $\frac{1}{2}$, and one convolution that maps features to RGB. They use 6 blocks for 128 × 128 images and 9 blocks for 256×256 and higher-resolution training images. They also used instance normalization in their model [11]. For the discriminator networks they used 70 × 70 PatchGANs [3, 6, 7] which aims to classify whether 70 × 70 overlapping image patches are real or fake.

1.8 Training Details

They trained their model from scratch (with using a pre-trained model), with a learning rate of 0.0002. They devided the objective (loss) by 2 while optimizing D, which slows down the rate at which D learns, relative to the rate of G. They kept the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs. Weights are initialized from a Gaussian distribution $\mathcal{N}(0, 0.02)$.

2 EXPERIMENTS & RESULTS

We trained our model over the cityscapes dataset. The dataset that we used had 2975 real images and also 2975 segmentation images. We also were using the Google Colab Server to train our model on its GPU's, however since we were using the GPU for a few days, the google colab limitations did not let us to use it much longer. However we had multiple google accounts and used others instead. The hyper-parameters such as batch-size, learning rate, etc, were all used from the paper itself.

In order to reproduce their results we did two different tasks to reproduce their result, one was Cycle Consistency Loss Only, where we assumed $\lambda_{Identity} = 0$, in the Cycle Consistency Loss + Identity Loss part we assumed $\lambda_{Identity} = \lambda_{Cycle}$

Figure 7 shows discriminator loss for the ground image and the segmentation. However since we were using limited resources out google account will run out of RAM, therefore we were only able to show the loss for the first 66 epochs. Other than the discriminator loss we plotted the Cycle Consistency loss (figure 8) and the generators loss (figure 9) As you could see in all cases the loss is dropping however the way that they are dropping is a bit different.

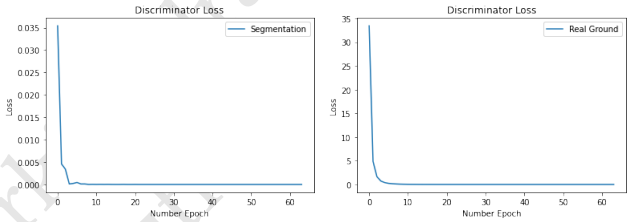

Figure 7: Discriminator loss

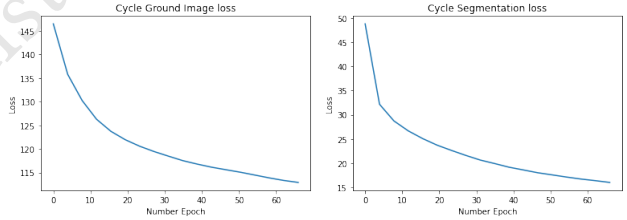

Figure 8: Cycle Consistency loss

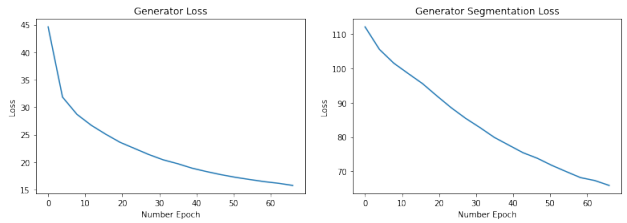

Figure 9: Generator loss

2.1 Cycle Consistency Loss Only

We trained our model for 200 epochs, figure 10 shows the accuracy of the Discriminator, where it shows the percentage of correct guesses for the discriminator for detecting real ground images of

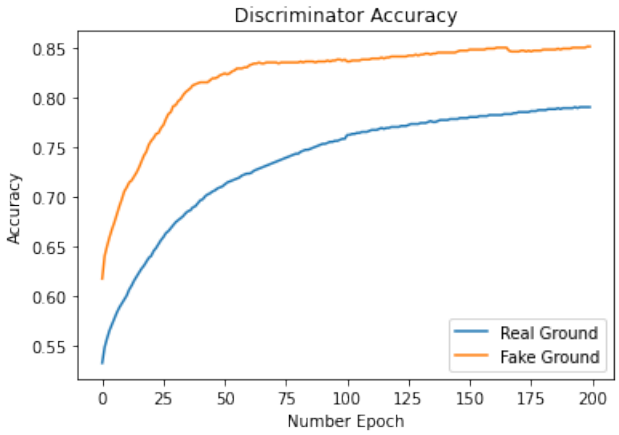

Figure 10: Discriminator Accuracy

fake ground images. Out result and output was quite reasonable however due to limited resources we had, we were not able to train the entire model with full potential. The results were not bad, the performance for generating ground images was better. Figure 11 shows the result of out CycleGAN after 200 epochs of training without applying the identity loss. Figure

Figure 11: Generated Ground image (without identity)

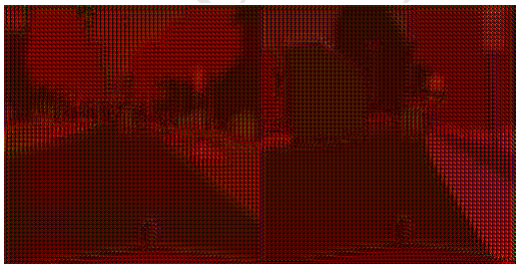

Figure 12: Generated Segmentation (without identity)

2.2 Cycle Consistency Loss + Identity Loss

Since we had limit access to Google Colab servers instead of just re-training our model, we used the weights from the previous model (the model with no Identity loss), then we fine tuned the model with the new hyper-parameters. The only difference in the two hyper-parameters is that in the case we used the identity loss, parameter $\lambda_{Identity}$ was not 0 and was the same value as λ_{Cycle}. We trained our model for another 70 epochs and the figure 13 shows the accuracy of the model during training. This accuracy shows the number of times in total the model has detected if the image was real or fake. Since we were fine tuning the model the first 200 epochs are the same as in figure 10. The final result from fine tuning the model is in figures 14 and 15. We can see the great improvement in the ground image result in figure 14. Meanwhile the result for the segmentation model in figure 15 did not show much improvement.

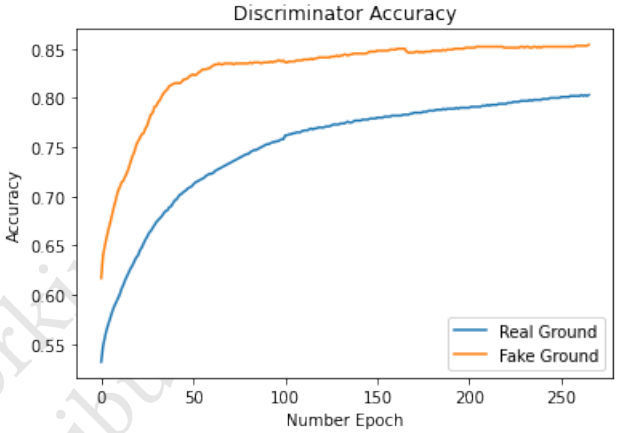

Figure 13: Discriminator Accuracy

Figure 14: Generated Ground image (with identity)

We also wanted to reproduce table 2 in paper [12] however since the result from the segementation model was a RGB image we were not able to categorize each pixel to each class, therefore we were not able to reproduce that part. In order to solve that part we need a complete description of how they converted a RGB image to a catagorzied image so that each pixel will correspond to a class.

3 DISCUSSION

To sum up the result we got from this project, we were not able to get the exact same numbers and outputs, however the results were quite acceptable. The images from the paper and the one from our result may not 100% match but we could assume that

Figure 15: Generated Segmentation (with identity)

if we trained for a longer amount of time we might have gotten the same result. By looking at the generated images we could see that the models performance over generated ground images are quite well, but of course not as good as the main images. For the generated segmentation images the results were not as good as the ground images, one thing that we need to consider is that every color in the segmentation image is going to represent a particular class, however what the CycleGAN model is seeing is a normal image and it will treat the image the same way and that is why the segmentation results are not categorized. Since the segmentation output is not categorized(the output was just a RGB image) we were not able to fully reconstruct that part, the reason could be that there wasn't enough information about segmentation process. In the paper [12] we were going to reproduce table 2, but due the reasons that I just mentioned about the segmentation we were not able to calculate the table since there was no clear approach on how to map a RGB image to a category (such as roads, trees, cars, etc). It is also interesting since in the paper [12] in one figure (look at figure 16) the result they have when they used Cycle Alone is quite similar to the result that we have gotten in our implementation, however their final result is better than ours.

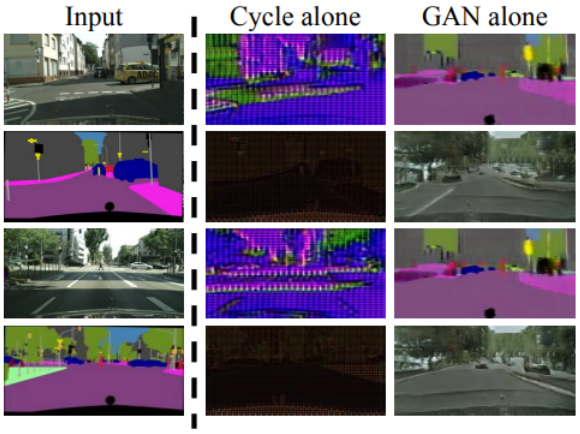

Figure 16: Generated Segmentation (with identity)

4 DETAILS

For the Video of the project you could go to my youtube channel and watch it there (link), since the video was 300 MB, I was not able to upload it one Eclass, therefor I mentioned my youtube link. For the project details, I used two Google Colab accounts so I could train my model. Python notebooks are not easy to commit on git since each time you need to download them first, then you need to commit on git. Here my Google Colab links if you want to ensure that the project was conducted at a reasonable pace. First account Second Account, there are other Google colab codes here and here. Since when the network is training we were not able to run anything else, we needed to use multiple notebooks to hangle them all. In one case we were getting the accuracy of the model but since the Run time ended we weren't able to save the accuracy therefor we just copy pasted the numbers (which was frusterating task). . However there are some sorted python files in my repository (I tried to format them is a correct and standard way), however, they are same code in my google colab. GitHub Repository

Also great thanks to internet blogs [10] and youtube videos [2, 5, 9] that helped me to understand the details of the paper.

REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016.
[2] Developers Hutt. [Online] unpaired image to image translation || cyclegan tutorial.
[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. pages 5967–5976, 07 2017.
[4] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. 03 2016.
[5] Ahlad Kumar. [Online] deep learning 46: Unpaired image to image translation network (cycle gan) and discogan.
[6] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.
[7] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *ArXiv*, abs/1604.04382, 2016.
[8] Arnab Mondal, Aniket Agarwal, Jose Dolz, and Christian Desrosiers. Revisiting cyclegan for semi-supervised segmentation, 08 2019.
[9] Aladdin Persson. [Online] cyclegan paper walkthrough.
[10] Hardik Bansal Archit Rathore. [Online]understanding and implementing cyclegan in tensorflow.
[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. 07 2016.
[12] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

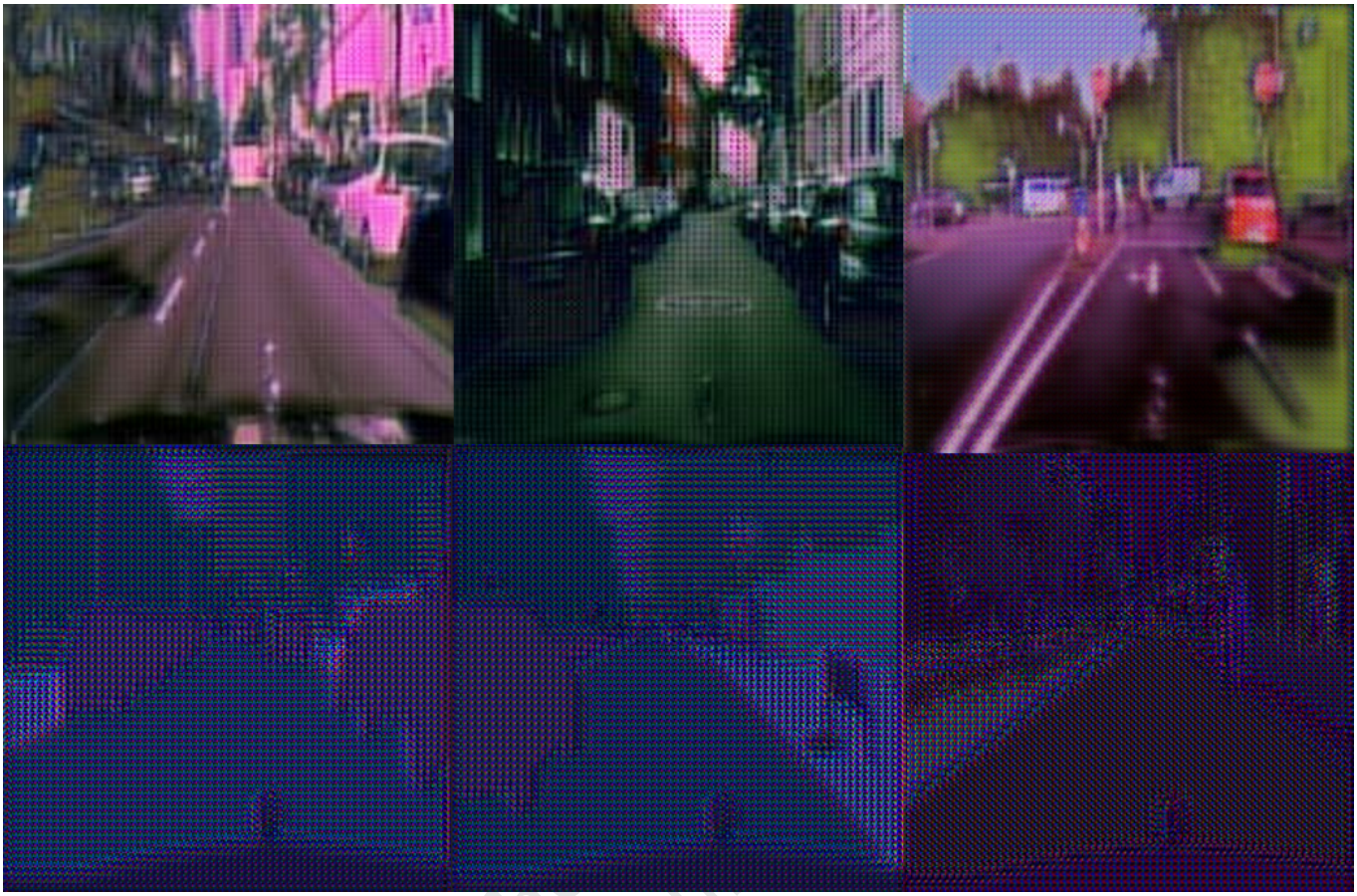

Figure 17: Some other Samples of our result