CS543 Final Report

Image Segmentation to Improve CycleGAN

Andrew Chen (aachen3) Patrick Cole (pacole2) Sameer Manchanda (manchan2)

1 Introduction

In recent years Generative Adversarial Networks (GANs) have become an effective way of generating a sample of a distribution given a random latent variable as input [6]. This is done by modeling the problem as a min-max optimization problem. The generator network is responsible for transforming images from the latent space to the observed space, and the discriminator network discriminates between generated and true examples. We want to train our generator to fool the discriminator, but we want to train the discriminator to correctly identify true samples.

Furthermore, GAN architectures have been successfully adapted to image to image translation by mapping the probability distribution of the input space to that of the output space. The generators accomplish this by encoding the input space into some low dimensional distribution that represents key features in both the input and output space and then decoding the output from the low dimensional distribution [3]. This optimization technique has been applied to unpaired image to image translation in the CycleGAN paper [5]. The results have been impressive, but for some images the generator affects more of the image than originally intended. We hypothesize that some spatial information about the object to be transformed is sometimes incorrectly encoded and decoded when generating a sample, causing the transformation to "leak" out of the intended object region. To solve this we propose using segmentation on the original image to obtain a mask which will be used, with our generator output, to give the final output of our model. The idea is that the mask received by segmentation will retain the spatial properties and improve our results.

2 Methods

2.1 CycleGAN Specification and Training

Let X denote a set of images that share a common characteristic, and let Y be another such set of images. A standard CycleGAN model will learn two generators, $G : X \to Y$ and $F : Y \to X$, such that $G(F(x)) \approx x,\ x \in X$ and $F(G(y)) \approx y,\ y \in Y$. The discriminators are denoted $D_X : X \to [0, 1]$ and $D_Y : Y \to [0, 1]$, whose goals are to classify whether these image belongs to the respective classes, X and Y. From this we can construct an adversarial training model as follows:

$$\min_{G,F} \max_{D_X,D_Y} \mathcal{L}_{GAN}(F, Dx, X, Y) + \mathcal{L}_{GAN}(G, Dy, X, Y) + \mathcal{L}_{cyc}(G, F, X, Y) \tag{1}$$

where the cyclic loss is defined as

$$\mathcal{L}_{cyc}(G, F, X, Y) = \mathbb{E}_{x\sim p(X)}\left[\|F(G(x)) - x\|_1\right] + \mathbb{E}_{y\sim p(Y)}\left[\|G(F(y)) - y\|_1\right], \tag{2}$$

the GAN loss for a particular function and discriminator is defined as

$$\mathcal{L}_{GAN}(F, Dx, X, Y) = \mathbb{E}_{x\sim p(X)}[D_x(x)] + \mathbb{E}_{y\sim p(Y)}\left[(1 - D_Y(y)\right]. \tag{3}$$

An additional loss term, called the identity loss can be added to suggest that multiple applications of a generator will be equivalent to a single application of the generator.

$$\mathcal{L}_{id}(G, F, X, Y) = \mathbb{E}_{x\sim P(X)}\left[\|F(x) - x\|_1\right] + \mathbb{E}_{y\sim p(Y)}\left[\|G(y) - y\|_1\right]. \tag{4}$$

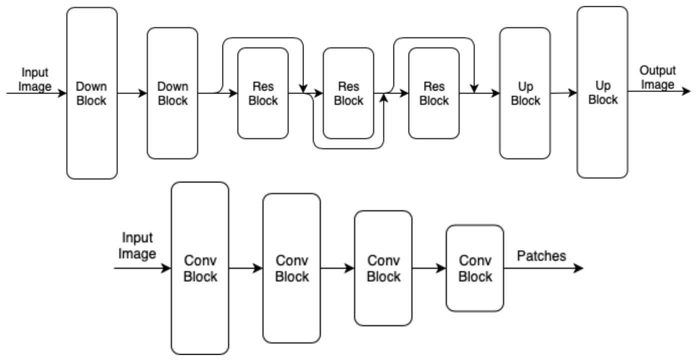

Figure 1: Generator and discriminator architectures

2.2 Mask R-CNN

Segmentation was used to detect objects that need to be transfigured (i.e. horse). We used a pretrained Facebook Mask R-CNN to segment the image and create masks for each image [2]. The Mask R-CNN architecture consists of two different segmentation stages and combines them to get the mask. The first stage performs region of interest (RoI) proposal on the input image. A RoI is a specific patch in the image. Then, the network predicts the class of the object and refines the bounding box around the object. Finally, the network combines the proposal and bounding box to generate a mask.

2.3 Model Architecture / Implementation

2.3.1 Generator

The generator uses a Encoder-Decoder structure with ResNet style residual layers in the middle. This can be seen in Figure 1. Below are the general structures of each block type inside the generator, i.e. "Down Block", "Res Block", and "Up Block". This was influenced by the architecture used in the original CycleGAN paper [5].

Down Block

1. Convolution 2d with a kernel size of 3 and stride of 2
2. Instance Norm
3. ReLU activation layer

Res Block

1. Reflection Pad the image by 1
2. Convolution 2d with a kernel size of 3
3. ReLU activation layer
4. Reflection Pad the image by 1
5. Convolution 2d with a kernel size of 3
6. Instance Norm

Up Block

1. Transposed Convolution 2d with a kernel size of 3 and stride of 2
2. Instance Norm
3. ReLU activation layer

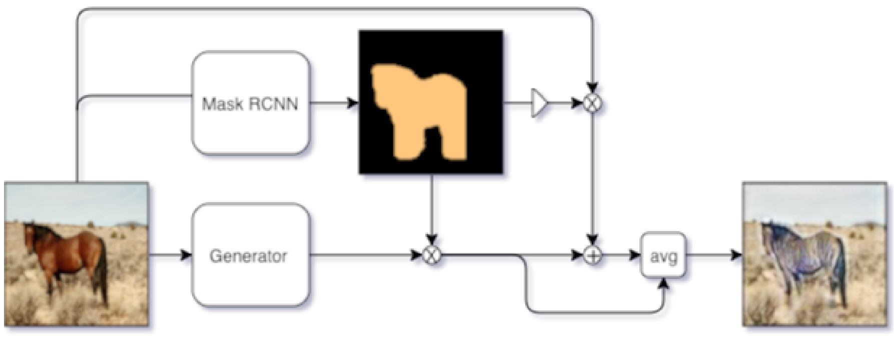

Figure 2: Segmentation integrated with generator (Soft Mask)

2.3.2 Discriminator

The discriminator was heavily based off of a the Patch GAN discriminator that was used in the Cycle GAN paper [5]. The discriminator is altered slightly from the reference discriminator which can be found on the official git repository [4]. This will create a $n \times n$ output of patches with values for whether each patch was real or fake. This architecture has been stated to perform just as well as a traditional discriminator, but with the advantage of having significantly less parameters. This architecture can be seen in Figure 1 and below the general structure of the "Conv Block" used can be seen.

Conv Block

1. Convolution 2d with a kernel size of 4 and a stride of 2
2. Instance Norm
3. ReLU activation layer

2.3.3 Soft Mask

We didn't want to use a hard segmentation as a border between what was transformed and what remained the same, so we developed a pipeline that we call "Soft Mask" which can be seen in Figure 2. The Mask R-CNN portion of the pipeline used the pretrained model from the FAIR Mask R-CNN team [1]. Using that mask, we found the region of the object in the input image to the generator. The mask is applied on the output of the generator and the resulting region is added to the output of the generator. Then, the result of the generator is averaged with the previous output to create the soft mask.

3 Results

3.1 Computation and Photo Acquisition

For training and testing we utilized the "AISE TensorFlow NVIDIA GPU Notebook" instance on Google Cloud compute configured with the NVIDIA Tesla V100 GPU and 8 VCPUS. The Monet photos and photographs were gathered from a Flickr source by the UC Berkeley team who originally worked on the Cycle GAN project. They also gather the zebra and horse photos from the ImageNet data set. They have made the data publicly available so we used these for our training and testing purposes. Note, for training purposes and the limited computational resources 128×128 images were used for training, but the default images are much larger so our results may seem blurry because they are being scaled up.

3.2 monet2photo

- Monet: Training data (1073 images) Testing data (122 images)
- Photos: Training data (6288 images)Testing data (752 images)

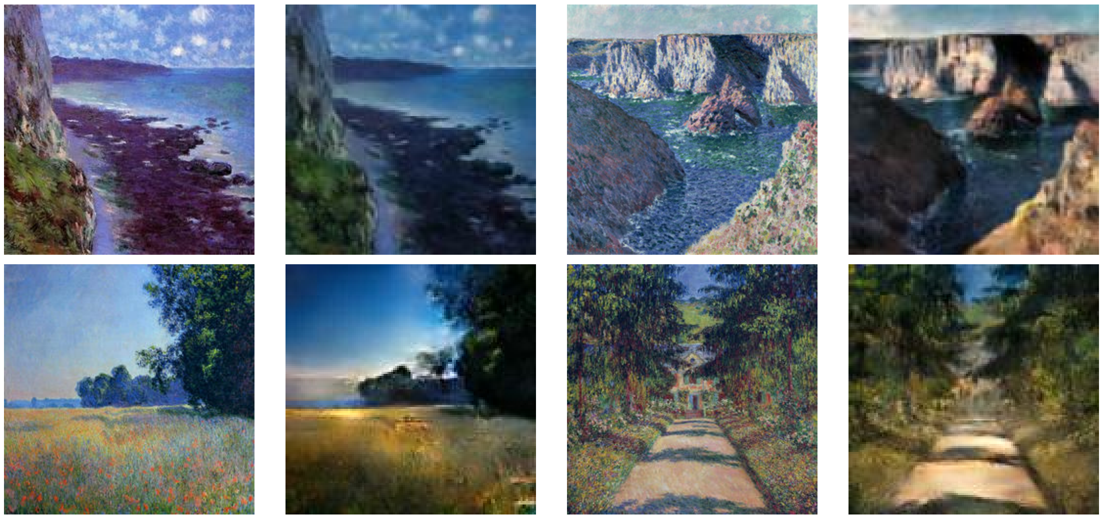

Figure 3: Several good examples of Monet paintings converted to photographs from our test set.

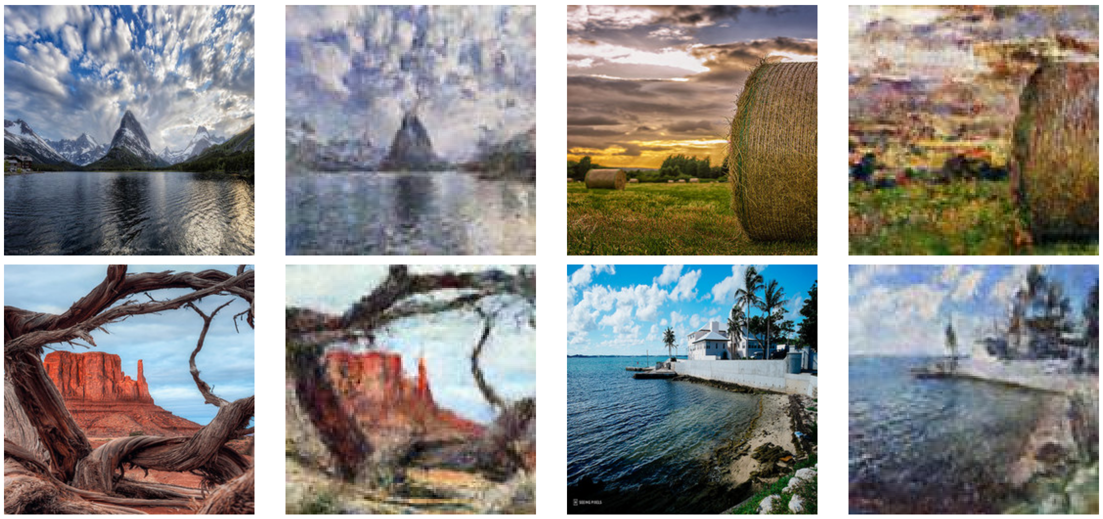

Figure 4: Several good examples of photographs converted to Monet paintings from our test set

It took approximately 23 hours to train the model for 200 epochs with a learning rate of 2e-4 and Adam optimizer with beta values (0.5, 0.999). The learning rate linearly decayed from 2e-4 to 0 between epochs 100 and 200. Some losses were more important to the actual output so they were weighted differently. The GAN loss given by Equation 3 had no weight to it, the Cyclic loss given by Equation 2 was increased by a factor of 10 and then the Identity loss given by Equation 4 was increased by a factor of 5. Identity loss was used for this particular dataset rather than segmentation because there is no clear advantage of using segmentation for the problem of style transfer due to the fact that it is desired for the whole image to be effected. Identity loss was used instead because it introduces a penalty on the output images color composition by enforcing that the generated image must still be relatively close to the input, but with less weight than the cyclic loss.

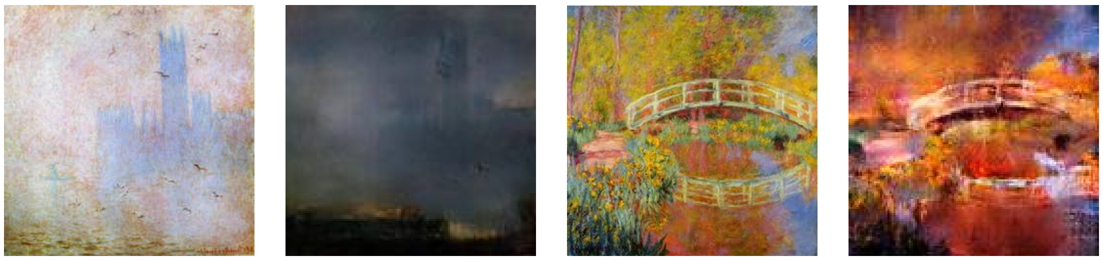

Figure 5: Several bad examples of Monet paintings converted to photographs from our test set. Generally, the GAN darkens the impressionist works; however, this leads to issues when converting foliage or mist.

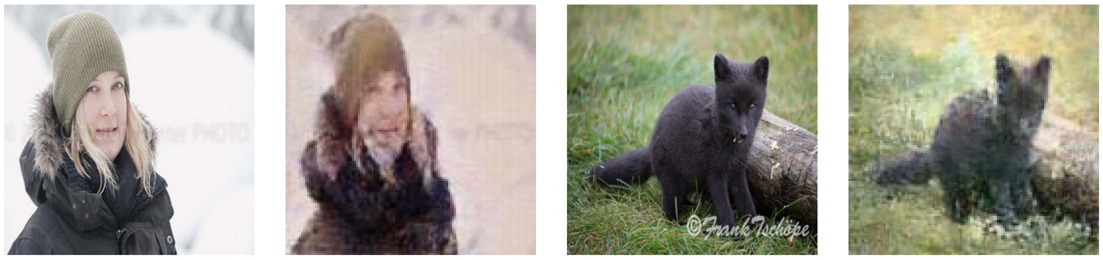

Figure 6: Two bad examples of photographs converted to Monet paintings from our test set. Notice the issues pertain to objects in the foreground.

In Figure 3 and Figure 4 we display some of the better results, while in Figure 5 and Figure 6 we display some failure cases. We will discuss/analyze the results of the failure case in the discussion section.

3.3 horse2zebra

- Horse: Training data (1067 images) Testing data (120 images)
- Zebra: Training data (1334 images) Testing data (140 images)

First, we generated masks for the training images using available Mask R-CNN code; selecting the top two masks per image [1]. After generating masks, we trained the CycleGAN, alternating between optimizing discriminator and generator parameters for 200 epochs, with a learning rate of 2e-4, $\lambda = 10$, and the Adam optimizer. The learning rate was linearly decayed from 2e-4 to 0 between epochs 100 and 200. The identity loss was not used because we felt it would adversely effect the resulting horse and zebra images by forcing the color to be similar. In Figure 7 and Figure 8 some results are shown from the test set used. Figure 9 and Figure 10 show some of the failure cases that will be discussed in further detail in the discussion section.

4 Discussion

4.1 monet2photo

When analyzing the data sets we noticed that the Monet paintings were mostly landscape photos, while the photographs were pretty wide spread and not constricted to just one category. This means that we were able to perform some visually appealing style transfer from one landscape to another. Such results can be seen in Figure 3 and Figure 4.

There were were a few pitfalls, in part due to the training data with which we were working. Most of the Monet photos were landscapes, and any specific objects appeared in a much smaller scale in the background. In Figure 6, the photos were of a woman and a little fox. Since these photos were mainly focused on one subject with different detail than a landscape photograph, the transformation appeared to blur the image.

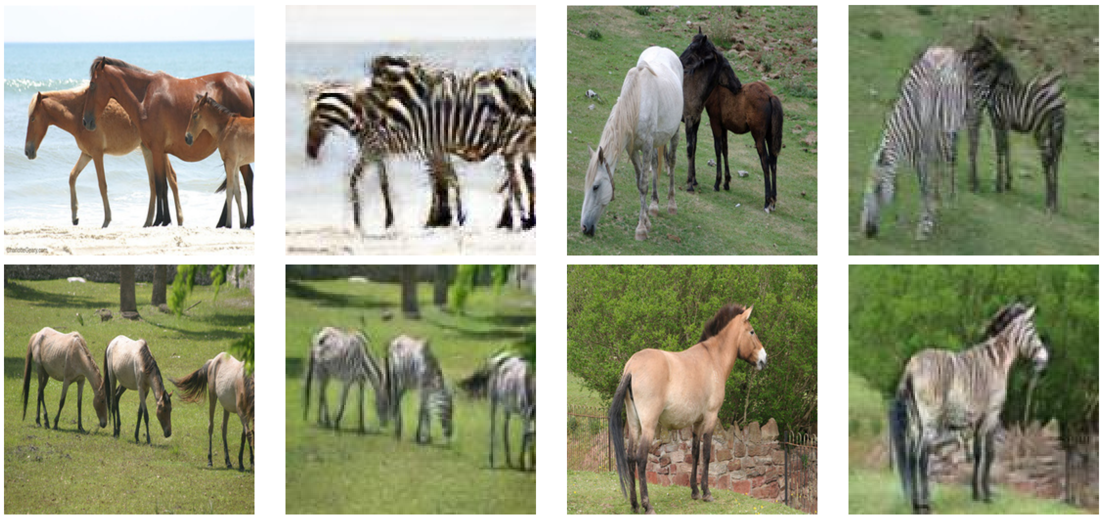

Figure 7: Several good examples of horse photographs converted to zebra photographs from our test set

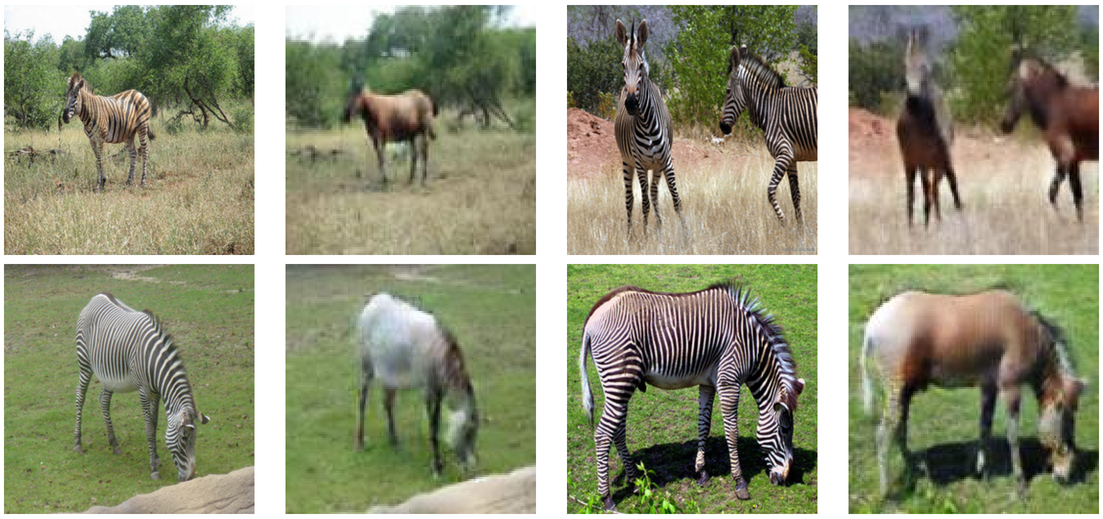

Figure 8: Several good examples of zebra photographs converted to horse photographs from our test set

Another issue with the translation of Monet painting to photos seemed have to do more with the color pallete of the input painting than the focus. The main problem cases seemed to be where the input colors were unnatural in real life photos. In Figure 5 it can be seen that the the first image seems to require a mist of some sort, but it ends up translating it to a dense fog. This could be due to the fact that the input colors were not representative of real life mist. The second paintings contained bright colors, and the output picture seemed very red and unrealistic. One solution for this could be to add a new loss that constrains regions in the image to have the same general color as their inputs.

4.2 horse2zebra

There are similar number of horse and zebra pictures in the dataset; however, we found the horse to zebra conversion was better than the zebra to horse. We hypothesize that generating zebras is easier because they are always striped black and white, while there are multiple ways to color a horse (chestnut brown, white, black, etc.). In addition, our results contained some unexpected surprises. The CycleGAN produced decent results, but, as seen in Figure 11, the background of the image was also altered. Also, sometimes additional

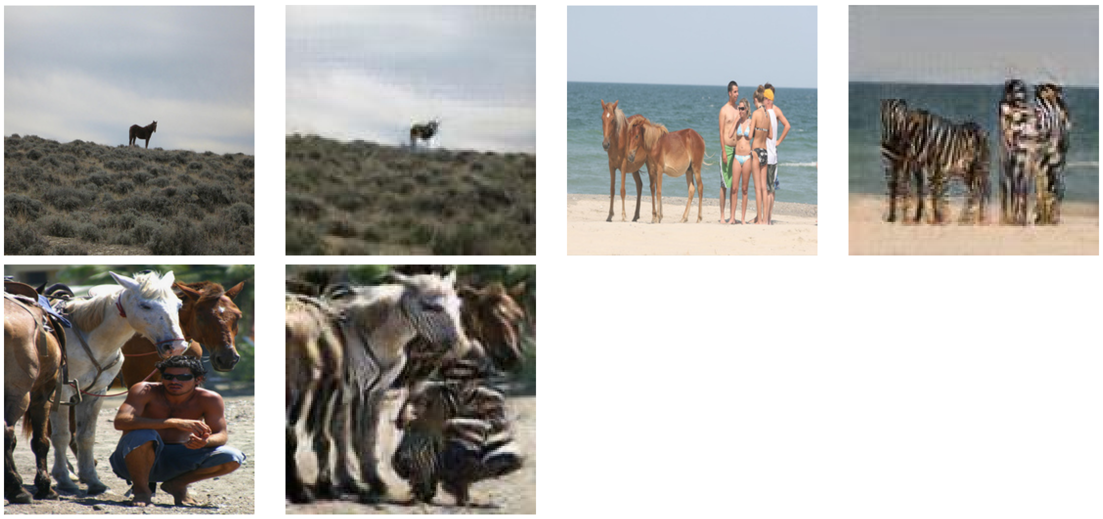

Figure 9: Three bad examples of horse photographs converted to zebra photographs from our test set. In the first image pair, the horse is not segmented, so it is not processed. In the second and third image pair, the humans get segmented along with the horses, so they get "zebrafied".

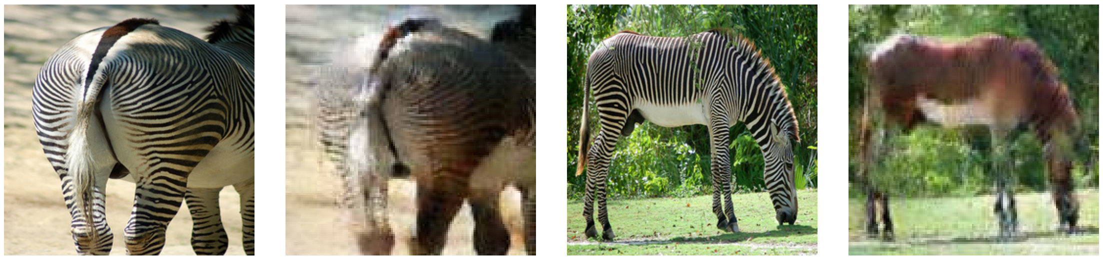

Figure 10: Two bad examples of zebra photographs converted to horse photographs from our test set. In the first image pair, the results retains striping. In the second image pair, the horse's head gets blurred beyond recognition.

parts of the image were unnecessarily transformed (ie. people were "zebrafied"). To combat these issues, we proposed and tested a "segmentation GAN". This method yielded some surprising results. In most of the cases, the mask produced by segmentation covered more area than the object itself, creating artifacts around foreground objects. In addition, the image gradient where the segmentation ends changed sharply. In a few cases, the converted object became blurred mixture of colors—an output for which we have no explanation. An example of this phenomenon can be seen in Figure 11. With the soft mask, the method described in Section , the results between the CycleGAN and the segmentation methods are averaged. This produced good results; simultaneously generating the transfigured object while making minimal changes to the background. Some of these results can be seen in Figure 8 and Figure 7.

However, there were a few cases in which the results were unexpected due to the training data or segmentation. We observed many cases in which segmentation gave unexpected results. One case was where the whole horse/zebra was not present in the image. Due to this, the segmentation did not pick up on the whole object, leading to a partially converted zebra as seen in Figure 10. Here, the back of the zebra is the only portion of it that was visible, and so it was only partially segmented and converted to a horse. Another issue was when the entire horse/zebra is not segmented. In this case, the soft masking of the image erased part of the zebra. For example, in Figure 10, the legs and head of the converted horse are missing since segmentation did not capture them. Although there are some issues with segmentation, it did help retain the background's original coloring (Figure 7).

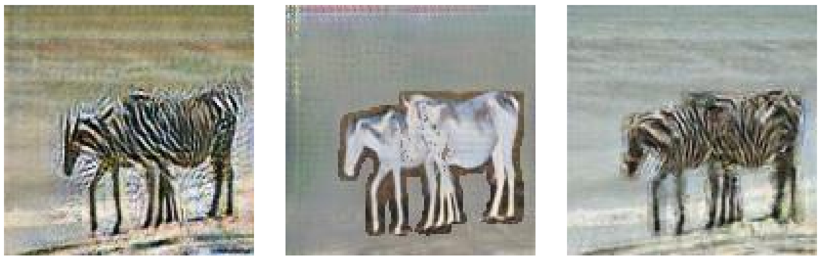

Figure 11: The results of 3 configurations of the GAN. The original cycleGAN makes strange changes to the water. The segmentation CycleGAN oddly fails to generate a zebra. The softmask CycleGAN generates a zebra while making minimal changes to the background. The input image was a pair of horses taking a walk on the beach.

5 Contributions

The work was evenly distributed. Sameer dealt mostly with the generator architecture and implementing segmentation. Patrick implemented the discriminator, data utilities, training and testing, while also assisting with the loss functions within the CycleGAN class. Andrew created the CycleGAN class and assisted with the training and discriminator code. Overall, we all worked together to debug the code and and run experiments on the Google Cloud Instance.

References

[1] F. Massa and R. Girshick, *maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch*, `https://github.com/facebookresearch/maskrcnn-benchmark`, Accessed: [May 6, 2019], 2018.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *CycleGAN and pix2pix in PyTorch*, `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`, Accessed: [May 6, 2019], 2017.

[5] ——, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.