Object Transfiguration using Cycle Generative Adversarial Networks

Yunhai Han
Department of Mechanical and Aerospace Engineering
University of California, San Diego
La Jolla, CA 92037
y8han@eng.ucsd.edu

Hanyang Xu
ECE
University of California, San Diego
La Jolla, CA 92037
hax032@ucsd.edu

Xinyuan Lu
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92037
xil069@eng.ucsd.edu

Zhenzhen Zhu
Department of Mathematics
University of California, San Diego
La Jolla, CA 92037
zhz065@ucsd.edu

Abstract

This project implemented Cycle Generative Adversarial Networks (CycleGAN) to achieve object transfiguration, which transfers a specific object in a given image to another different type. In specific, our goal is to translate between horses and zebras as well as between oranges and apples, using images originally downloaded from ImageNet. Our CycleGAN network used UNet-256 as the generator and PatchGAN as the discriminator. We trained our method on unpaired images where a source image and a corresponding image are from two different domains (e.g., horses and zebras are different domains). We implemented a final loss function defined as the sum of the regular adversarial loss, the cycle consistency loss, and the identity loss. We evaluated our results by plotting the related training loss curves and examining the actual output images from the test set. We found that ResNet-9 generated much better obaject transfiguration results than UNet-256 as we could clearly see zebra strips showing on the translated hours images on using ResNet-9. One interesting observation was that the suitable coefficient of identity loss was different from training horses↔zebra (0.5) to turning oranges↔apples (1).

1 Introduction

Image to image translation is a subclass in the computation vision and graphics problems. Image to image translation are tasks that take in input images and generate or manipulate them into a different visual space. Traditionally, different tasks such as converting grayscale to color, image to semantic labels, and edge-map to photograph require different hand-crafted machinery. We are in particular interested in object transfiguration, which transfer a specific object in a given image to another different type. Years of research in computer vision, image processing, computational photography, and graphics have produced powerful translation systems in the supervised setting, where example image pairs are available. However, obtaining paired training data can be difficult and expensive. We therefore seek an algorithm that can learn to translate using unpaired examples where a source image X and a corresponding image Y are from two different domains. Recent development of GANs offers a more general approach for such similar tasks through learning an "adversarial" loss that adapts to data. Based on our research, Cycle GAN is the suitable network for our task of object transfiguration since it turns images of one domain into images of another domain without the need

for explicitly providing matching pairs in the training set. The idea behind this project is to make the translation from one domain to the other domain. We train a mapping $G : X \rightarrow Y$ such that the output $\hat{y} = G(x), x \in X$, is distinguishable from images $y \in Y$ by an adversary trained to classify $\hat{y}$ from y [10]. Our project experimented with two sets of data, horse2zebra and apple2orange to demonstrate the CycleGAN network capabilities.

2 Motivation

Generative Adversarial Network applications are increasing rapidly. We as a group are intrigued by the realistic generated images that some GAN networks created during our final project research. Therefore, we want to deep dive into the architecture and implementation of some GAN network by doing a final project on it. Image Translation Problem is a suitable task for GAN network to solve. Previous works on image translation requires paired dataset in training which added complexity to the project and effects model generalization. CycleGAN architecture is particular interesting to us since it improves GAN network's ability drastically while maintaining the classic GAN network architecture. Its ability to learn the mapping between two unpaired dataset and generate images without deliberately learning the specific correspondence between features in the image pair was ground breaking when it was first introduced . We hope by implementing the CycleGAN network, we can have a good understanding of the Generative Adversarial Network from both the conceptual level and from the implementation level.

3 Related Work

CycleGAN Generative Adversarial Network solves the Unpaired Image-to-Image Translation problem. Image-to-Image Translation aims to learn a translation function between a pair of image datasets targeted for translation. Unpaired Image-to-Image Translation further reduces the requirement on dataset by aiming to learn the translation function on two unrelated datasets [8]. CycleGAN is built on two major tools: Generative Adversarial Networks[2] and Convolutional Neural Network for Image Generation and Classification. Generative Adversarial Networks is a class of machine learning frameworks that typically involves training two Neural Networks which objectives are conflict with each other. The generative network generates candidates while the discriminative network evaluates them. Generative Adversarial Networks are widely used in image generation area and thus the appropriate tool for solving Image-to-Image Translation problem. CycleGAN uses standard UNet[7] or ResNet[3] deep Convolutions Neural Network to generate the image of target category from the input category, and uses PatchGAN[4] network as an image sub-section level discriminative network. PatchGAN is a classic convolutional network classifier that classify a N×N pixel patch of the image into real or generated image. This classifier is applied to the entire image convolutionally and the final classification is the average of all results. It shows equally state-of-the-art performance in the classification task with less parameters in previous image generation tasks and therefore was adopted in the CycleGAN architecture. We also incorporated the implementation framework from CSE251B Programming Assignment 4 into the implementation of our version of CycleGAN since we really appreciate the modularity and simplicity to use of that implementation framework.

4 Method

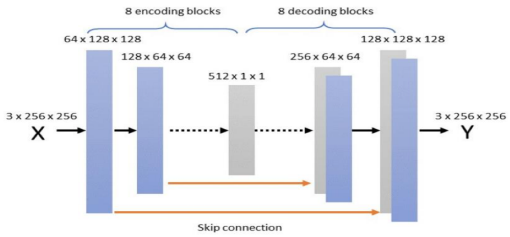
4.1 Data

We trained our networks using images in the horse2zebra.zip and orange2apple.zip file provided by Jun-Yan Zhu et al.[10]. These images were originally downloaded from ImageNet using keywords wild horse, zebra, apple, and navel orange, and were scaled to 256×256 pixels. There are 939 horse images, 1177 zebra images, 996 apple images, and 1020 orange images in the training set.

4.2 CycleGAN

Our CycleGAN model used UNet generator and PatchGAN discriminator. The UNet generator is an encoder-decoder network with symmetrical long skip connections. The network consists of 8 encoding layers and 8 decoding layers, with skip connections from layer i go layer $n - i$, where n

is the total number of layers. Each encoding and decoding block follows the form of Convolution-BatchNorm-LeakyReLU. Since it's easy to overfit these networks to our training images, the UNet network introduced skip connections that helps the decoder learn more details from the encoder directly, and improves the gradient flow as we go backwards. Figure1 illustrates the UNet architecture.



(a) U-Net

Figure 1: UNet generator network architecture

.

We used a convolutional PatchGAN classifier as our discriminator. PatchGAN discriminator determines whether an image is real or fake by using local patches of size 70 × 70, rather than the entire image. The discriminator takes in two images, tht input image x and the unknown image $G(x)$, passes them through 5 downsampling convolutional-InstanceNorm-LeakyRely layers, and determines whether the image is real or fake.

4.3 Loss function

The goal of Image-to-Image translation is to learn the mappings between a set of input images and a set of output images. In our task, we would try to translate horses into zebras or turn oranges into apples and vice versa (e.g. summer $\leftrightarrow$ winter). These are the most typical examples of object transfiguration. However, in order to fulfill these tasks, the aligned image pairs are not always available and often hard to obtain. Thus, an approach of learning from an image domain X to the target domain Y in the absence of paired training data is really useful. Jun-Yan Zhu et al.[10] proposed Cycle-consistent Adversarial Networks, which is able to learn such mappings using an adversarial loss in the absence of paired example images. They introduces the concept of cycle consistency loss to enforce that the inverse mapping would be similar to the original image data. As shown in Fig. 2, the final loss function would be the sum of the regular adversarial loss, the cycle consistency loss, and the identity loss.

The goal is to learn the mappings between two domains X and Y, given the training samples $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and $\{y_j\}_{j=1}^{M}$ where $y_j \in Y^1$. There are two mappings in the model, say $G : X \rightarrow Y$ and its inverse$F : Y \rightarrow X$. The two discriminators D_X and D_Y aim to distinguish between images Y and translated images $G(x)$ and between images X and translated images $F(y)$, respectively. The adversarial losses are used to match the distribution of generated images to the original images as mentioned before. On the other hand, the cycle consistency is to prevent the learned mappings G and F from contradicting each other.

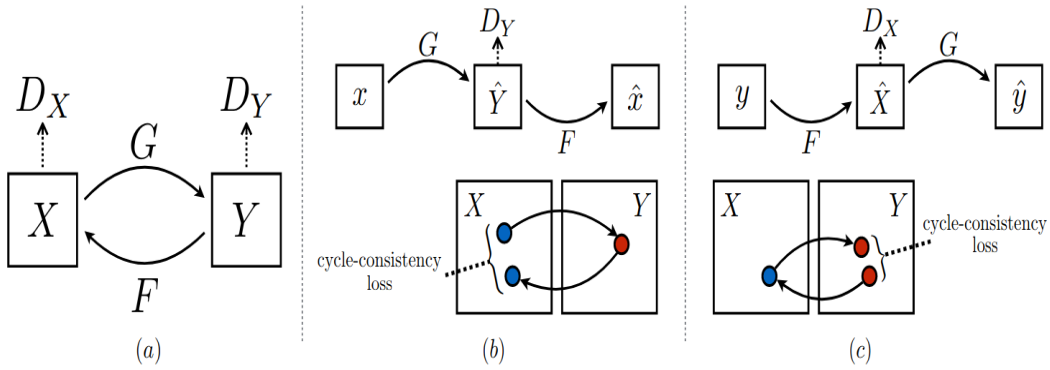


Figure 2: Demonstration of the adversarial loss and the cycle consistency loss (from [10]). (a) The model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X.D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

.

4.3.1 Adversarial Loss

For the mapping function $G : X \rightarrow Y$ and its discriminator D_Y, the objective as:

$$\begin{aligned} \mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) = & E_{y \sim p_{\text{data}}(y)}\left[\log D_Y(y)\right] \\ & + E_{x \sim p_{\text{data}}(x)}\left[\log\left(1 - D_Y(G(x))\right)\right] \end{aligned} \tag{1}$$

where G works as generator, which tried to create images $G(x)$ that look similar to images from domain Y, while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. G aims to minimize this objective against an adversary D_Y that tries to maximize it. The same is for F and D_X.

4.3.2 Cycle Consistency Loss

Only adversarial loss can not guarantee that the learnt mapping can translate an individual input to a desired output. In order to reduce the space of possible mappings, based on the intuition that they should be cycle-consistent (that is $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ or $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$), the cycle consistency loss function is defined as:

$$\begin{aligned} \mathcal{L}_{\text{cyc}}(G, F) = & E_{x \sim p_{\text{data}}(x)}\left[\|F(G(x)) - x\|_1\right] \\ & + E_{y \sim p_{\text{data}}(y)}\left[\|G(F(y)) - y\|_1\right] \end{aligned} \tag{2}$$

After adding this loss, it is reasonable to see the reconstructed images pretty similar to the input images. This Cycle Consistency Loss constraint added a week constraint on the form of generated image so that the generated image would have similar overall form with the input image. Therefore, Cycle Consistency loss is an important constraint that help to enable the unpaired image translation ability of CycleGAN However, it is proven that, in training, this constraint is not strong enough to enforce the absolute color and texture consistency. Therefore, we also introduce Identity Loss into the overall loss function.

4.3.3 Identity Loss

Identity loss minimized the difference between the transformed image and input image. Mathematically, it is expressed as follow:

$$\begin{aligned} \mathcal{L}_{\text{identity}}(G, F) = & E_{x \sim p_{\text{data}}(x)}\left[\|G(y) - y\|_1\right] \\ & + E_{y \sim p_{\text{data}}(y)}\left[\|F(x) - x\|_1\right] \end{aligned} \tag{3}$$

Intuitively, the Identity Loss enforces the constraint that minimal changes to the generated picture should be made by the generator compared to the input image. Since Discriminator has imposed the constraint that generated image should be recognized as the target category, these two constraints combined to enforce the generator only changes the target object and nothing else. For example, a well trained horse to zebra generator only changes the pixel related to horse, but not other pixels such as grass or tree. For the above reason, identity loss is an important constraint that help to enable the unpaired image translation ability of CycleGAN.

4.3.4 Full Loss Function

Combine the above two functions, we then obtain the full loss function:

$$\begin{aligned}\mathcal{L}\left(G, F, D_X, D_Y\right) = & \mathcal{L}_{\mathrm{GAN}}\left(G, D_Y, X, Y\right) \\ & + \mathcal{L}_{\mathrm{GAN}}\left(F, D_X, Y, X\right) \\ & + \lambda \mathcal{L}_{\mathrm{cyc}}(G, F) \\ & + \lambda \mathcal{L}_{\mathrm{identity}}(G, F)\end{aligned} \tag{4}$$

Where, λ controls the ratio of the two loss functions. Thus, by training cycle-GAN, we want to solve:

$$G, F = \arg\min_{G,F} \max_{D_z, D_Y} \mathcal{L}\left(G, F, D_X, D_Y\right). \tag{5}$$

5 Results

We built our own UNet-256 generators and PatchGAN discriminator, and trained the method as described in Section4. It took us about 6 hours to train **horses↔zebras**(using Unet-256 and almost 1300 images for 200 epochs) or about 13 hours to train **oranges↔apples**(using Unet-256 and almost 700 images for 400 epochs). The transform results of **horses↔zebras** using the latest model is shown in Fig. 3 and Fig. 4.

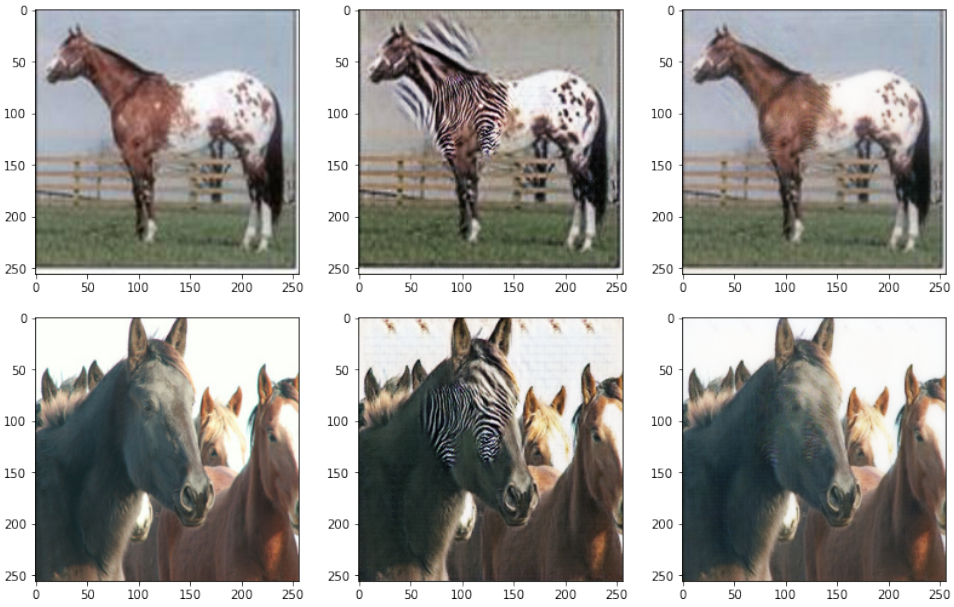


Figure 3: Horse2Zebra Translation: We transformed the horse images into zebra images (left column: input horse images; center column: output zebra images; right column: recreated horse images).

The loss curves of **oranges↔apples** is shown in Fig. 5. Using the best model, the translation results are shown in Fig. 6.

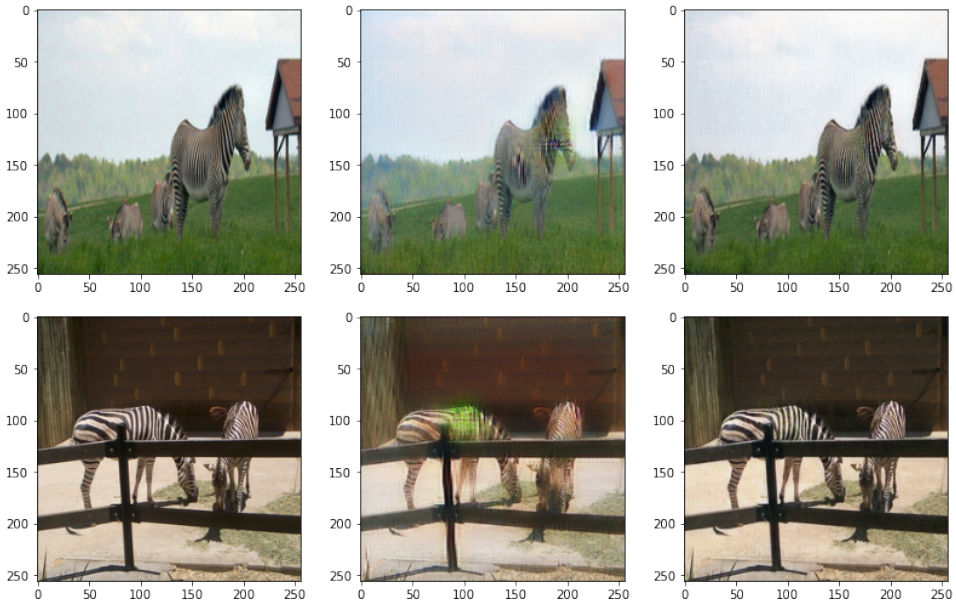


Figure 4: Zebra2Horse Translation: We transformed the zebra images into horse images (left column: input zebra images; center column: output horse images; right column: recreated zebra images).

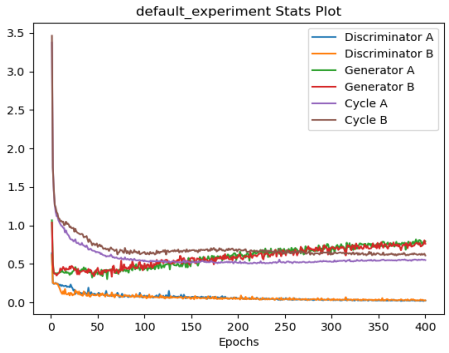


Figure 5: The loss curves for the oranges↔apples training

We can conclude that UNet-256 can not provide as good results of **horses↔zebras** as shown in Jun-Yan Zhu et al.'s paper[10]. The reason is that Unet-256 can not learn well the features in the input images (256×256) of horses and zebras. But, compared with the results of **oranges↔apples**, we can conclude that Unet-256 is good enough to learn the features from the fruit images(since they are less complex). Thus, for **horses↔zebras**, we replaced the previous generator with a built-in ResNet-9 network and re-trained the network from scratch. The results of **horses↔zebras** are shown in Fig. 7 and 8. For each figure, the first row shows the input images and the second/third/fourth row shows the output images after 5/45/90 epochs. From both sets, a clear change between the input and output image can be found, which illustrates that the neural network learns the two different image distributions and how to map from one to the other.

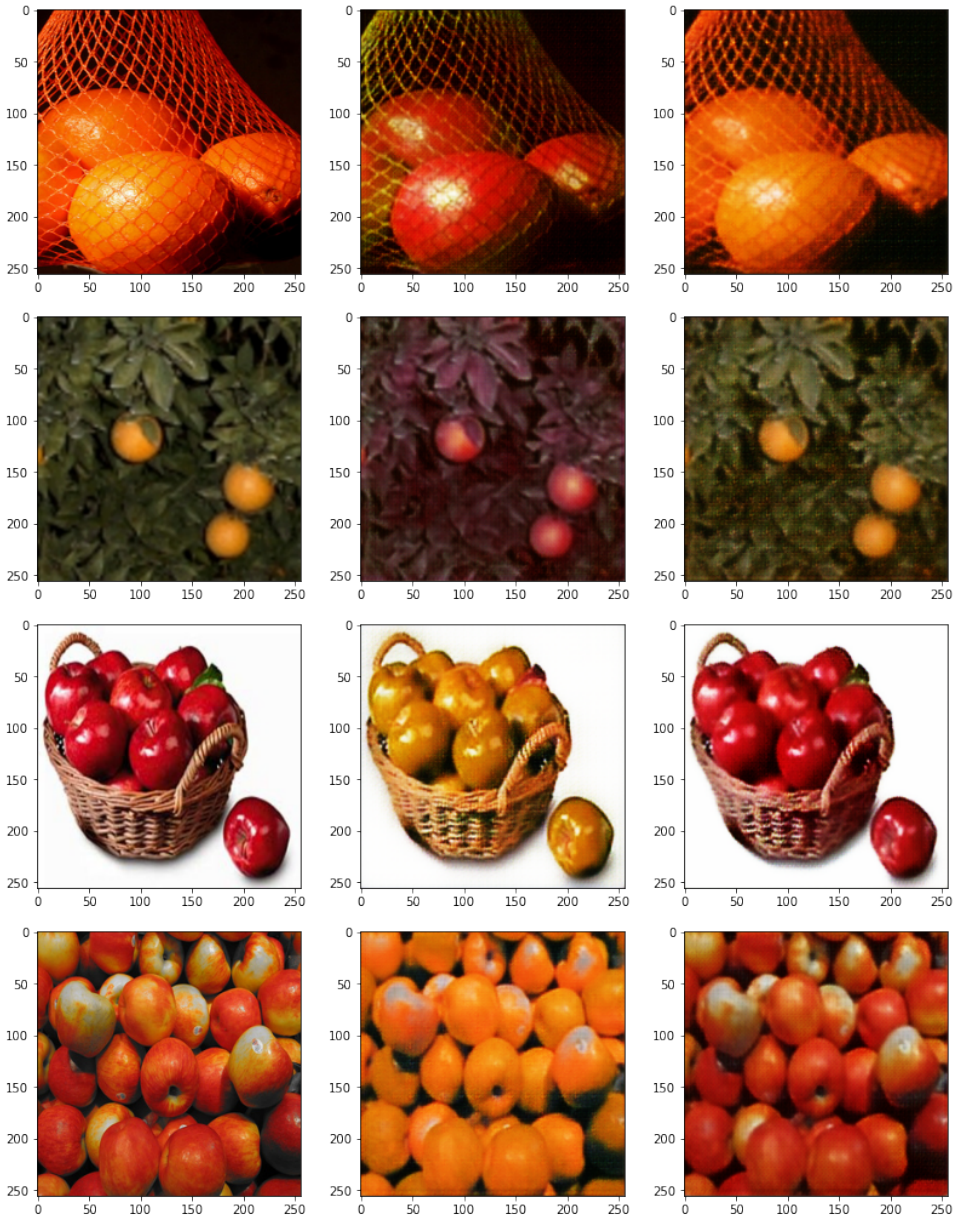


Figure 6: Oranges↔apples Translation using 380 epochs with batch size 16: We transformed between orange images and apple images (left column: input images; center column: output images; right column: recreated images).

Figure 7: Horse2Zebra Translation: We transformed the horse images into zebra images (first row: input horse images; second/third/fourth rows: output zebra images after 5/45/90 epochs).

Figure 8: Zebra2Horse Translation: We transformed the zebra images into horse images (first row: input zebra images; second/third/fourth rows: output horse images after 5/45/90 epochs).

6 Discussion

6.1 Loss Curves

GAN network performance is generally not reflected from the loss curve since the GAN training can be unstable, and the Generator/Discriminator losses are in equilibrium due to adversarial loss. However, we can still learn some useful insights from the loss curve. First, the Discriminator MSE loss are stable around 0.25 which implies that the Discriminator has a 75% accuracy, and the discriminator is working properly. Secondly, the cycle loss reduces constantly implies that the recreated images should be similar to the input image, which is the case in the training we performed.

6.2 Learning

During training, we observed that the Generator network has the most important effect on the generated image. UNet-128 produce the worst results in general. UNet-256 can produce good result on simple task such as converting between apple image and orange image, but it fall short of producing good result on converting between horse image and zebra image.[Fig.6]. ResNet-9, a more powerful network, produces good result on converting between horse image and zebra image.[Fig.7].

A very important improvement we have done upon the original paper is that the discriminator will not only judge the newly generated "fake" images, but also it has a capacity of 50 mini-batches. For each of the batches of the newly generated images, the image pool will have 50% probability using it and 50% probability storing it to the image pool and retrieve a batch of old "fake" images to use as the generated fake image. By doing so, we have increased the stability of the training process.

6.3 Identity loss helps fight against inverting colors

During the development process, we have noticed an inverted color problem in the transferred image. i.e., white inverted to black, red inverted to blue, blue inverted to green, etc. After some investigation, we found that the problem was caused by the network initialization issue. The cycle loss only ensured that the reconstructed result is similar to the original image but it did not force the generated result to be similar to the original one. Moreover, it would cause a cascade failure since the inverted color and the original color are from different domains and therefore lead the discriminator believe the difference between a real and a fake is the color rather than the target object. By adding an identity loss, we were able to generate the image similar to the original image.

6.4 Hyper-parameter tuning

There are several important hyper-parameters needed to be tuned based on each task. Since the cycle loss we used are averaged over pixels instead of summation, the cycle loss can be very small in comparison to other losses like discriminator loss or generator loss. Thus, we introduced a coefficient that helps boost the importance of the cycle loss, namely lambda. Changing lambda makes the reconstructed image clearer, the cycle effect will force the learning tend to recreate to the original image as much as possible.
Another important hyperparameter is the identity coefficient. As described in the previous subsection the idea behind the identity loss is to help fight the inverted color and make sure the changes are keep minimally. Different domain transfer will have different coefficient for identity losses. The identity loss coefficients for Horse2Zebra is 0.5 and for Apple2Orange is 1.

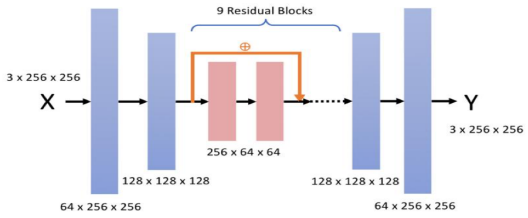
6.5 Moiré pattern in the result

Since we are using U-net and lots of filters of 4 by 4 to recreate the image, the final image has some Moire pattern when we display it. We first thought it was some defect in the UNet training but it turns out to be the zebra patterns. Once the network trains for more epoch the moire pattern in the image disappeared.

7 Future Work

At early epochs of the training process, we noticed that CycleGAN sometimes wrongly translated the unrelated objects (e.g. person, building) and background as the stripe texture in the horses $\rightarrow$ zebras translation. In order to address this problem and only translate regions containing target objects instead of whole images, we can try Attention-GAN. For example, Zihan Ye et al. proposed a novel pipeline called Deep Attention Unit Generative Adversarial Networks (DAU-GAN)[9]. DAU computes attention masks that point out where the target objects are, and hence makes GAN concentrate on translating target objects while ignoring meaningless backgrounds.
To achieve superior results, we can also change the generator to a residual-based network based on the ResNet model in Johnson et al.[5]. The ResNet generator composes of 2 encoding blocks, 6 or 9 residual blocks (ResNet-6 or ResNet-9), and 2 decoding blocks. Six residual blocks are used for 128×128 training images, and 9 residual blocks for 256×256 or higher-resolution train-

ing images[10]. Each encoding or decoding block follows the two-stride Convolution/Transpose Convolution-InstanceNorm-ReLU structure, and each residual block follows the Convolution-InstanceNorm-ReLU-Convolution-InstanceNorm residual connection structure. Reflection padding can be used to reduce artifacts. Fig.9 illustrates the ResNet-9 architecture.



(b) ResNet-9

Figure 9: ResNet-9 generator network architecture

.

8 Conclusion

We have achieved the result we want. Both horse2zebra and apple2orange have great result after training 400+ epoches. Since apple2orange has less feature, it is easier to train compared to the horse2zebra. But after all we are satisfied with the result. The link to our codebase is: https://github.com/y8han/UCSD_CSE251B/tree/master/Final%20Project

9 Team Contribution

9.1 Yunhai Han

Coding: I wrote the discriminator. I trained the horses↔zebras using the built-in Resnet network and generated the results. For the report, I wrote the parts of results.

9.2 Xinyuan Lu

Implemented the cycleGAN structure and implementation framework with Hanyang Xu. Written the discusion section.

9.3 Hanyang Xu

Implemented the cycleGAN structure and implementation framework with Xinyuan Lu. I believe Xinyuan deserve some extra credit since he continue to make significant contribution to the project after he broke his arm in a accident in the final week.

9.4 Zhenzhen Zhu

Coding: I wrote the UNet generator and sketched the ResNet generator (not used in the end). I trained the oranges↔apples and generated the results. As of the report, I wrote Abstract, Method-Data CycleGAN, and Future Work.

References

[1] L. A. Gatys, A. S. Ecker and M. Bethge. A Neural Algorithm of Artistic Style. arXiv:1508.06576v, 2015. https://arxiv.org/pdf/1508.06576v2.pdf

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[3] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for Image Recognition". arXiv:1512.03385

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Imageto-image translation with conditional adversarial networks. In *CVPR*, 2017.

[5] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[6] B. Liu, K. Song and A. Elgammal. Sketch-to-Art: Synthesizing Stylized Art Images From Sketches. arXiv:2002.12888v3, 2020. https://arxiv.org/pdf/2002.12888v3.pdf

[7] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597

[8] R. Rosales, K. Achan, and B. J. Frey. Unsupervised image translation. In *ICCV*, 2003.

[9] Zihan Ye, Fan Lyu, Linyan Li, Yu Sun, Qiming Fu, and Fuyuan Hu. Unsupervised Object Transfiguration with Attention. *Cognitive Computation 11*, pages 869–878, 2019. https://doi.org/10.1007/s12559-019-09633-3

[10] J. Zhu, T. Park, P. Isola and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *IEEE International Conference on Computer Vision (ICCV), Venice, Italy* 2017.

Object Transfiguration Using Cycle GANs with Unpaired Examples

Yunhai Han, Xinyuan Lu, Hanyang Xu, Zhenzhen Zhu

March 19, 2021

1

Problem of Interest

Fig. 1 The input images x, output images G(x) and the reconstructed images F(G(x))

Source: J . Zhu, T. Park, P. Isola and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *In ICCV*, Venice, Italy, 2017.

- Object Transfiguration

  E.g., Horses ⟷ Zebras

  Apples ⟷ Oranges

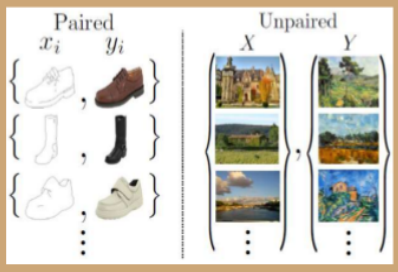
  Bicycles ⟷ Motorcycles

2

Motivation

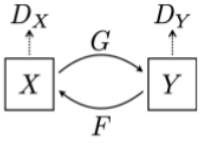
https://youtu.be/9reHvktowLY

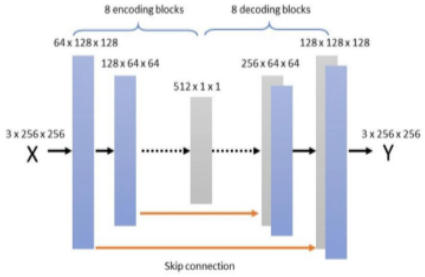
- Unpaired examples



Source: J . Zhu, T. Park, P. Isola and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *In ICCV*, Venice, Italy, 2017.

Methods - Cycle-GAN

- 



(a) U-Net

- **U-Net Generator**
  - 8 encoder blocks: Conv-BatchNorm-LeakyReLU
  - 8 decoder blocks: Trans Conv-BatchNorm-ReLU
  - Skip connections: information flow
  - Dropout: 50%
- **PatchGAN Discriminator**
  - Input images: x, G(x)
  - 5 downsampling layers: Conv-InstanceNorm-LeakyReLU
  - Output: 70 * 70 submeshes of Real or Fake?
- **Dataset**
  - 939 horse images and 1177 zebra images from ImageNet

4

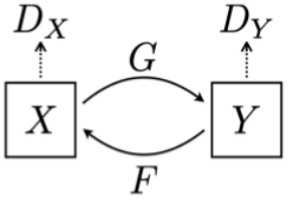
Methods - Loss

- **Standard GAN loss For Image Generation:**

Generator/Discriminator MSE loss

Generated Image want to fool the Discriminator

Discriminator wants to correctly recognize Generated Image



5

Methods - Loss

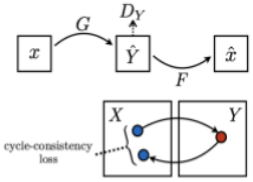
- **Additional Losses for Unpaired Image Translation Constraint**

Cycle L1 Loss between Input A and Recovered A G_A(G_B(A))**:**

- We should get A back if we translate A to B then back to A

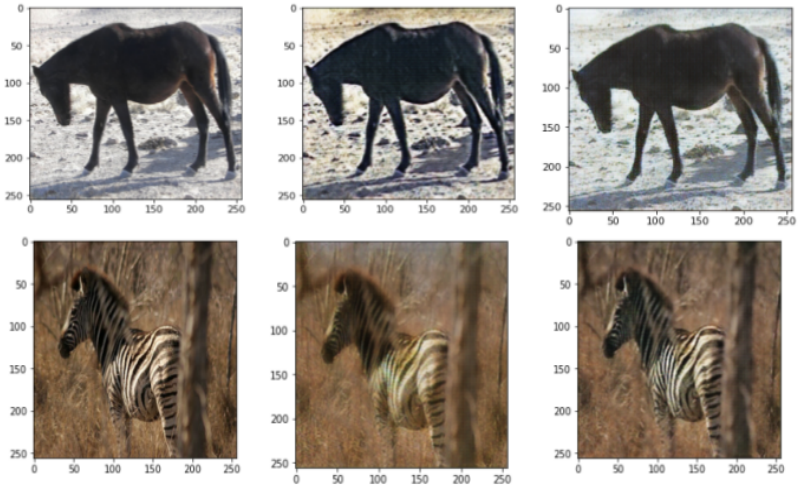
Identity L1 Loss between Input A and Output G_B(A)

- The Translated Image should be the mostly the same excepted the object being translated



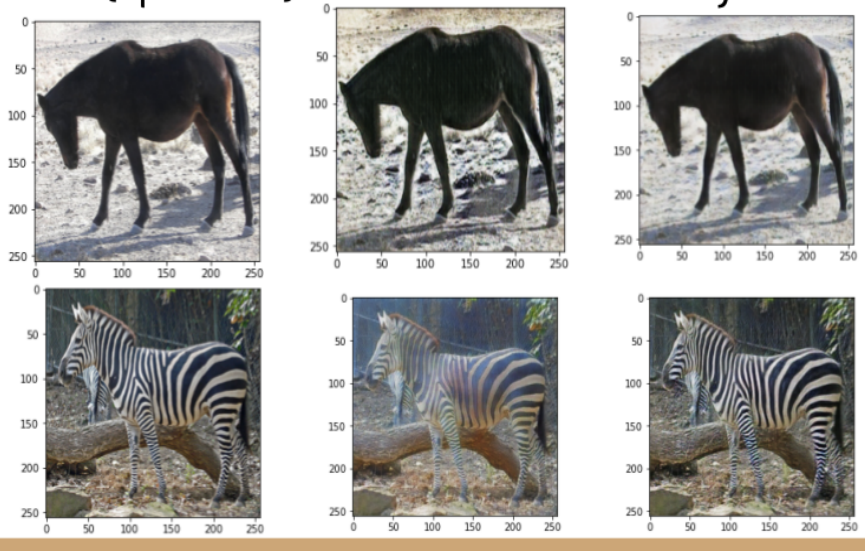
6

Result (epoch:20) Horse2zebra with 7 layer Unet



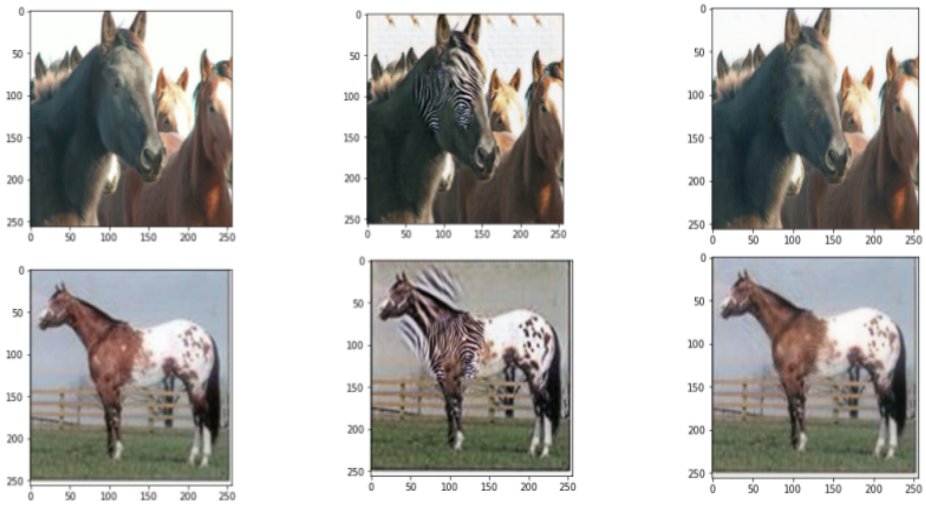
7

Result (epoch:80) Horse2zebra with 7 layer Unet



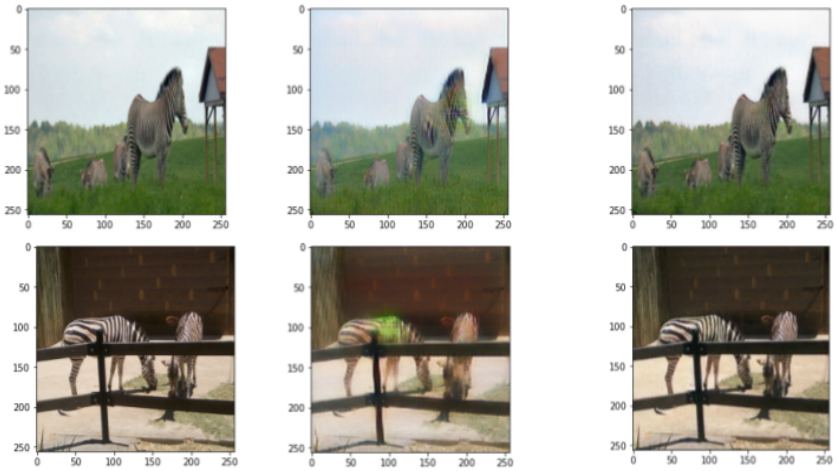
8

Result (epoch:200) Horse2zebra with 8 layer Unet



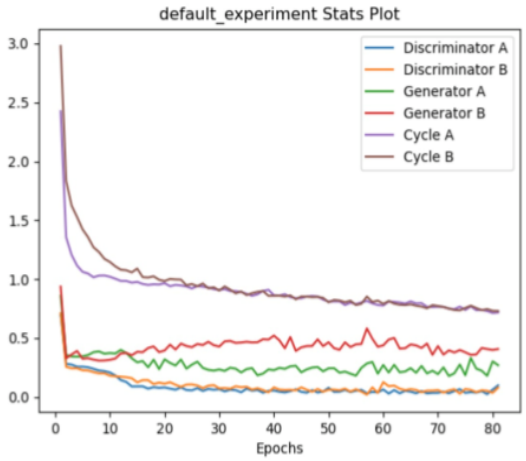
9

Result (epoch:200) zebra2horse with 8 layer Unet



10

Result



11

Some observations and discussion

Inverted Color problem

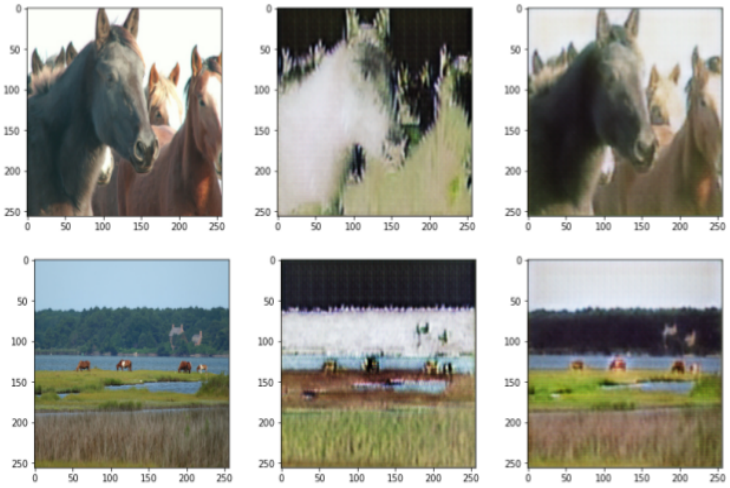
Moire Pattern in the image in early training

Image pooling to reduce the loss vibration

etc.

12

Inverted color problem

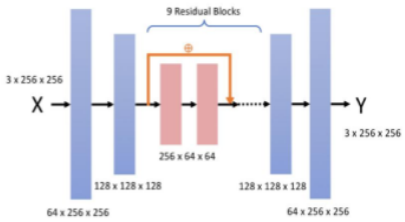


13

Moire pattern at early stage of training

14

Future Work



(b) ResNet-9

- ResNet generator [J. Johnson, et al.; Kaiming He, et al.]
  - 2 encoder/decoder, each with 2-stride Conv (Trans Conv)-InstanceNorm2d-ReLU
  - 6 or 9 Residual blocks, each with Conv-InstanceNorm-ReLU-Conv-InstanceNorm
  - Easier to learn the identify function
- Deep Attention Unit (DAU) GAN [Zihan Ye, et al.]
  - Computes attention masks that point out where the target objects are
  - Ignore meaningless backgrounds

15

Reference

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for Image Recognition". arXiv:1512.03385

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Zihan Ye, Fan Lyu, Linyan Li, Yu Sun, Qiming Fu, and Fuyuan Hu. Unsupervised Object Transfiguration with Attention. *Cognitive Computation* 11, pages 869–878, 2019. https://doi.org/10.1007/s12559-019-09633-3.

J. Zhu, T. Park, P. Isola and A. A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *IEEE International Conference on Computer Vision* (*ICCV*), Venice, Italy. 2017.

16

Thank you!