PROGRESS REPORT: INNER-CLASS AND INTER-CLASS IMAGE-TO-IMAGE STYLE TRANSFER USING CYCLEGAN CSE 253 DEEP LEARNING FINAL PROJECT

Shujie Chen
A53303042
shchen@ucsd.edu

Felix Gabler
A53329004
fgabler@ucsd.edu

Yongchuang Huang
A53322973
yoh001@ucsd.edu

Jingpei Lu
A91033661
jil360@ucsd.edu

Yiran Xu
A53270264
y5xu@ucsd.edu

May 8, 2020

ABSTRACT

In this project, we explored the domain gaps between *inner-class* domains and *inter-class* domains. Also, to solve the problem of image alignment, we implemented CycleGAN [1] and trained it on different data. As a result, we can generate reasonable images among inner-class domains but have a poor result on inter-class domains.

1 Introduction

Image-to-Image translation [2] (I2I) aims to look for the mapping from one visual domain to another. In fact, many other computer graphics problems can be merged into I2I problem, including style transfer, image synthesis, and colorization. I2I have also be applied to domain adaptation problem recently.

The main challenge for I2I is that aligned and paired images are difficult to collect or even do not exist (*e.g.* artwork $\leftrightarrow$ real photo). Humans can, however, imagine how a harbor would be like if Vincent Van Gogh paints it, even if Van Gogh has never painted a harbor before. Many researchers [3–5] are trying to solve this problem. The most famous method is CycleGAN [1], which introduces cycle-consistency loss and allows the networks to learn without paired images.

In this project, we focus on I2I translation among classes. Here, the class refer to the specific attribute or style, *e.g.* artworks, animals, photograph as described in [6]. *Inner-class* has objects or scenes that own similar attribute. For example, in *animal* class, we can have *dog, cat, zebra* and *horse*. Those sub-classes are all animals and some of them have similar attributes, *e.g horse* and *zebra*. Also, *Inter-class* denotes the domains that is across different classes. For instance, in *artwork* and *animals*, *cat* and *Van Gogh* are an inter-class pair. We also show some examples in Figure 1.

The goal of this project is to explore the domain gaps between inner-class and inter-class using CycleGAN.

2 Motivation

In this project, we implemented CycleGAN [1] as our model to generate fake images. Compared to previous methods, CycleGAN can handle unpaired images by using cycle-consistency loss, which means we have more datasets available instead of using aligned datasets. In particular, there is rare images for artworks with specific objects or topics. For example, it is hard to find a painting of a skyscraper by Vincent van Gogh in the real world.

Also, to realize our goal for this project: to explore the domain gaps between inner-class and inter-class, we experimented with a lot of datasets. The idea of classes is inspired by BAM dataset [6], where they introduces different classes with

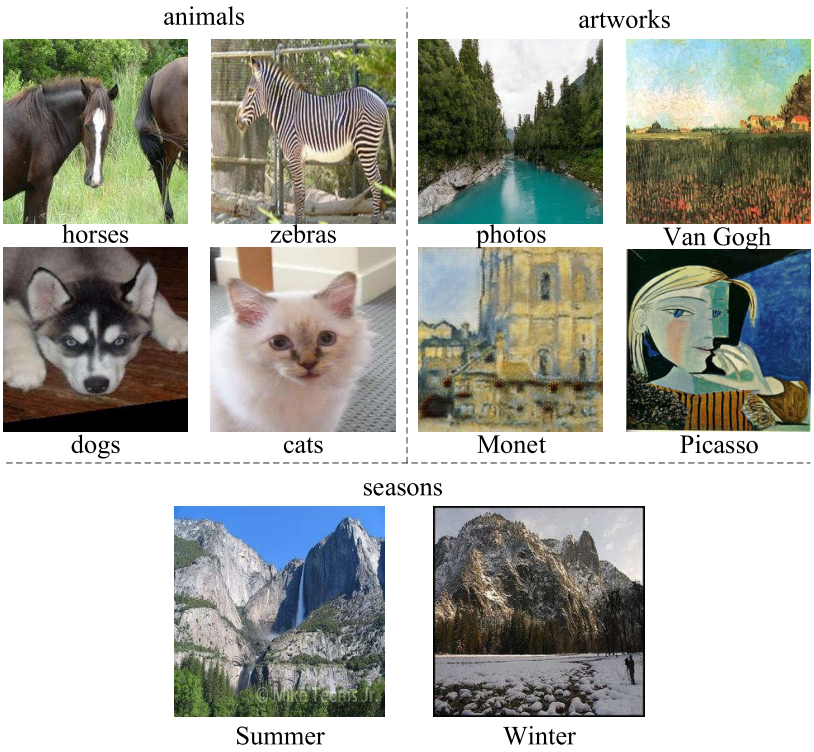


Figure 1: Inner-class and Inter-class. Here, **animals**, **artworks** and **seasons** are all classes. We define regular photos as one of the artworks.

specific attributes. Intuitively, we humans can imagine different transfer even if there is a big gap for the networks. For example, we can imagine that how a "horseman" drawn by Pablo Picasso looks like, but for the machines, it seems hard for them to find the connection between a horse and a painting by Picasso. We are curious about how much this domain gap can be learned by CycleGAN. Therefore, we explored the those gaps using inner-class and inter-class images.

3 Related Work

Image-to-image translation. The idea of I2I was brought at least by Hertzmann et al [7], where a non-parametric model on a single input-output training image pair was used. Recently, as the emerge of large datasets and the success of CNNs, parametric methods are popular by training with paired images. Later, pix2pix [8] becomes a paradigm by using a conditional generative adversarial network to learn the mapping. Similar frameworks then are implemented into other I2I tasks, such as pictures generation from sketches [9], style transfer [10] and semantic layouts [11].

GANs. Generative Adversarial Networks (GANs) [12] have achieved incredible performance in domain adaptation problems, such as image generation [13, 14], text2image [15], future prediction [16], video frames [17] and 3D object generation [18]. The key to the success is the introduction of adversarial loss and its training pattern, which involves a generator and a discriminator. However, the training method is also restricted by paired images. Later, several papers explored unpaired image training. CoGAN [3] exploited weight-sharing strategy across different domains. Other people [5] used different representations for "content" and "style" (or "attribute") to encourage the network to learn same "content" features. Finally, CycleGAN [1] used cycle-consistent loss as well as adversarial loss to train the networks without paired images. Most recently, DRIT [2] is proposed to solve paired I2I and mode collapse at the same time. In this project, we implemented CycleGAN and then tried to explore its usage on I2I problem, especially on style transfer.

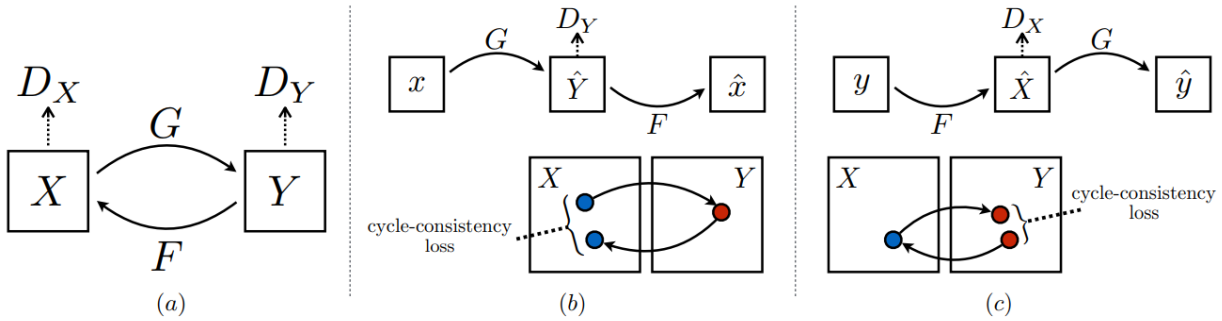


Figure 2: (a) The training process of CycleGAN [1]. Two mappings, $G : x \rightarrow y$ and $F : y \rightarrow x$ are contained. The corresponded discriminators are D_Y and D_X. (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$.

Figure 3: Summer to winter in the Yosemite National Park

4 CycleGAN

4.1 Overview

An overview of CycleGAN is shown in Figure 2. Different from previous GANs, it involves two generators G, F and two discriminators D_X, D_Y. Suppose we have two domains X and Y, our goal is to estimate the mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. To do this, CycleGAN uses cycle-consistency training paradigm. First, we compute forward cycle-consistency by $x \rightarrow G(x) \rightarrow F(G(x)) \approx \hat{x}$ and then we can compute the forward cycle-consistency loss between x and $\hat{x}$. Meanwhile, an adversarial loss is computed based on $\hat{y} = G(x)$. Next, the backward cycle-consistency loss is calculated by $y \rightarrow F(y) \rightarrow G(F(y)) \approx \hat{y}$, also an adversarial loss is computed according to $\hat{x} = F(y)$. To sum up, there are two steps and three losses involved. The implementation details of our generator, discriminator and losses will be discussed in the following sections.

4.2 Generator

Following [1], we used 9-block Residual network as our generator, which is actually an encoder-decoder architecture. Each block contains a two convolutional layers with residual connection [19]. The details can be found in Table 1. The generator includes a downsampling process and an upsampling process. At the end, the output will be a 3-channel image with the same size of the input image.

layer name	layer setting
conv0_1	$[7 \times 7, 64]$, pad=1, stride=1 Instance Norm + ReLU
conv0_2	$[3 \times 3, 128]$, pad=1, stride=1 Instance Norm + ReLU
conv0_3	$[3 \times 3, 128]$, pad=1, stride=2 Instance Norm + ReLU
Residual Block 1	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 2	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 3	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 4	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 5	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 6	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 7	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 8	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
Residual Block 9	$[3 \times 3, 256]$, pad=1, stride=1 Instance Norm + ReLU $\times$ 2
deconv 1	$[3 \times 3, 128]$, stride=2 Instance Norm + ReLU
deconv 2	$[3 \times 3, 64]$, stride=2 Instance Norm + ReLU
conv final	$[7 \times 7, 3]$, pad=1, stride=1 Tanh

Table 1: Detailed architecture of generator: 9-block

4.3 Discriminator

Similar to [1, 8], we used the idea of PatchGANs,which aim to classify whether 40×40 overlapping image patches are real or fake. The details are shown in Table 2.

layer name	layer setting
conv 0	$[4 \times 4, 64]$, pad=1, stride=2 Leaky ReLU
conv 1	$[4 \times 4, 128]$, pad=1, stride=2 Instance Norm + Leaky ReLU
conv 2	$[4 \times 4, 256]$, pad=1, stride=2 Instance Norm + ReLU
conv 3	$[4 \times 4, 512]$, pad=1, stride=1 Instance Norm + ReLU
conv final	$[4 \times 4, 1]$, pad=1, stride=1

Table 2: Detailed architecture of discriminator

4.4 Losses

The objective for the CycleGAN is to transfer the images in one domain to the realistic images in another domain. The loss function is the combination of the adversarial losses [12] and the cycle consistency loss, such that:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) \tag{1}$$

where λ controls the relative importance of the two different losses, $\mathcal{L}_{GAN}$ and $\mathcal{L}_{cyc}$ are adversarial loss and cycle consistency loss, which will be introduced in the following.

The adversarial losses $\mathcal{L}_{GAN}$ is utilized to train the mapping functions and the discriminator. For mapping function $G : X \rightarrow Y$ and the discriminator D_Y, the adversarial loss can be expressed as:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \tag{2}$$

where $y \sim p_{data}(y)$ denotes the data distribution in domain Y and $x \sim p_{data}(x)$ denotes the data distribution in domain X. The mapping function G tries to minimize the loss while the discriminator D_Y tries to maximize the loss, i.e. $\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y)$. Similar for the mapping function $F : Y \rightarrow X$ and the discriminator D_X, the objective is $\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X)$.

To avoid the mapping functions just learning the random permutation of images in the target domain, the cycle consistency loss is utilized to further constraint the learning for the mapping functions. As illustrated in Fig. 2, the cycle consistency loss can be expressed as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1] + \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] \tag{3}$$

where it calculates the L_1 norm of the difference between the remapped images and the actual images.

To effectively learn the mapping functions, the objective of the whole CycleGAN system aims to solve that:

$$G^*, F^* = \text{argmin}_{G,F} \max_{D_Y, D_X} \mathcal{L}(G, F, D_X, D_Y) \tag{4}$$

where the optimal mapping functions should minimize the loss described in the equation 1 while the discriminators try to maximize it.

5 Experiments

5.1 Setup

We prepare our data from [1, 2] and a dataset[1] including many artworks from Kaggle. The data are split into training set and test set with a rough ratio of $4 : 1$.

We used a 9-block Residual network as our generator and 40×40 PatchGAN as our discriminator. There are 64 channels in the last layer of the generator's encoder as well as in the first layer of the discriminator.

The learning rate is 0.0002. We trained for 100 epochs. Before training, the input images are first resized to 256×256 and then cropped randomly to 224×224. We used a batch size of 4.

As shown in Figure 1, we experimented with several inner-class translation: *summer* $\leftrightarrow$ *winter*, *cat* $\leftrightarrow$ *dog*, *horse* $\leftrightarrow$ *zebra*, *Ukiyo-e* $\leftrightarrow$ *photos*, *Picasso* $\leftrightarrow$ *photos*, *Van Gogh* $\leftrightarrow$ *photos* and *Monet* $\leftrightarrow$ *photos*. For inter-class translation, we present *Ukiyo-e* $\leftrightarrow$ *dog*, *Monet* $\leftrightarrow$ *dog*, *Picasso* $\leftrightarrow$ *zebra* and *Van Gogh to zebra*.

5.2 Inner-Class Style Transfer

Trained on images of Yosemite National Park in winter and summer, figure 3 displays one such inner-class style transfer. It can be seen that it does fairly well when the snow/grass area in the real image is rather small (top-most images). However, when there is too much snow or when water gets involved, performance decreases drastically (lower images).

We trained our model on images of cats and dogs. The result is shown in Figure 4. We can see that the mappings from cat to dog and from dog to cat are fairly good. However, it seems that, for cat to dog, all cats will be mapped to Samoyeds, and for dog to cat, all dogs are mapped to Ragdolls. This bias is caused by the fact that Samoyeds and Ragdolls are the most common ones in the training set.

For horses and zebras, we show some results in Figure 5. The quality of the learned mapping from horses to zebras is better than from zebras to horses. The reason is probably that it is hard to get rid of textures.

[1] https://www.kaggle.com/ikarus777/best-artworks-of-all-time

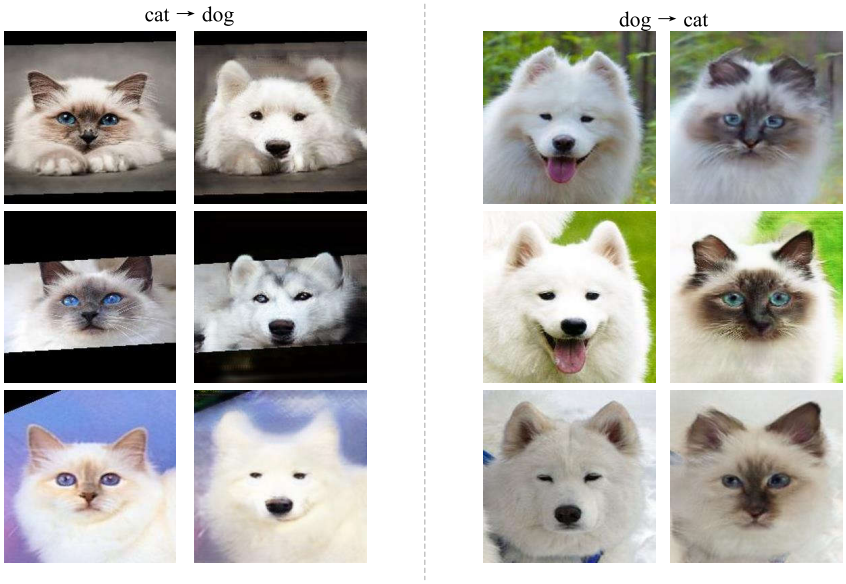


Figure 4: Cat to dog

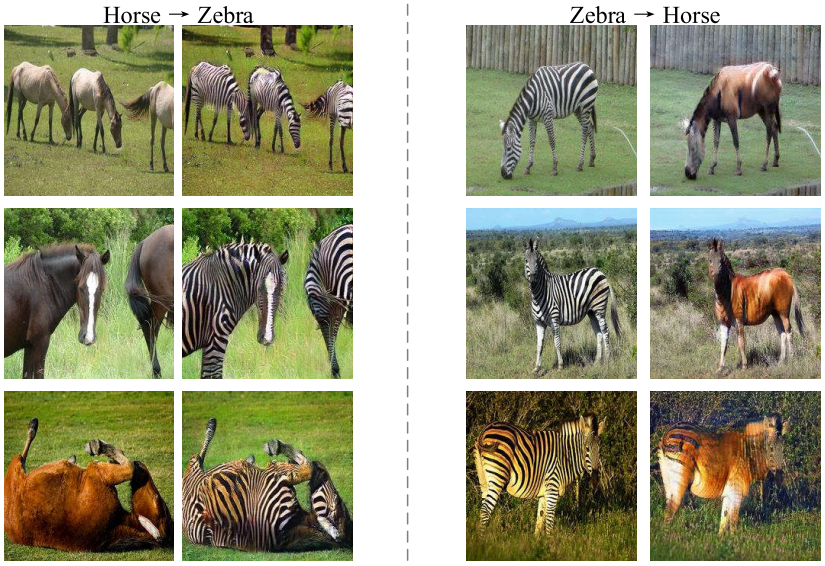


Figure 5: Horse to zebra

For the artworks, we show photos to Ukiyoe, Monet and Picasso in Figure 6, 8 and 7, respectively.

For Ukiyoe to photos, we can see that the quality of the mapping from photo to Ukiyoe is better, but from Ukiyoe to photo is worse. This is caused by the fact that Ukiyoes have strong painting attribute and different attribute from the real world. For example, in Figure 6, there are characters that are different from what a human looks like in the real world. This big gap makes the mapping from Ukiyoe to photo hard to learn.

For Picasso, we can see the strong personal style that makes the photos different from their original attributes. However, for Picasso's work, it is hard to map it to real photo. The reason is the same as Ukiyoes. The objects and figures in Picasso's work are different from what they are in the real world.

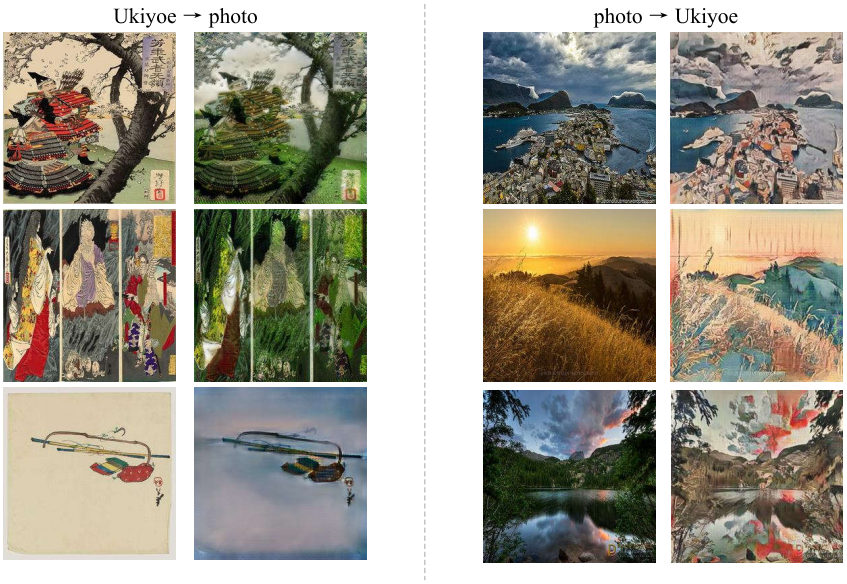


Figure 6: Ukiyo-e to photos

Figure 7: Picasso to photos

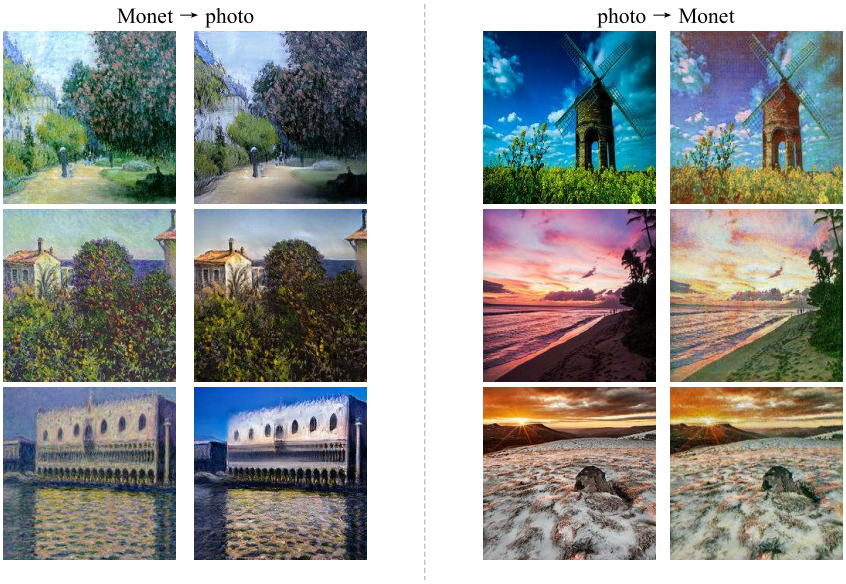


Figure 8: Monet to photos

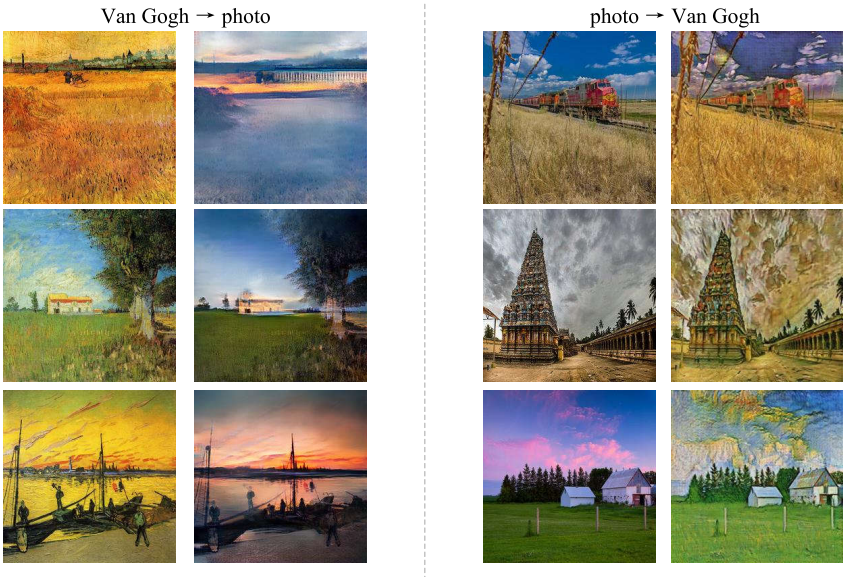


Figure 9: Van Gogh to photos

For Monet, however, we can see that the mappings are both fairly well in Figure 8. The reason is that Monet prefers to use landscapes as painting object, and the photos in the training set are almost landscapes. This makes sense since we have a rather small gap between those two domains.

For Van Gogh, the mappings are both working well. The reason is similar to Monet. Van Gogh loves drawing natural landscapes and the photos also contain many real landscapes. Also, we can see that Van Gogh has a obvious personal painting style that makes the photos change dramatically.

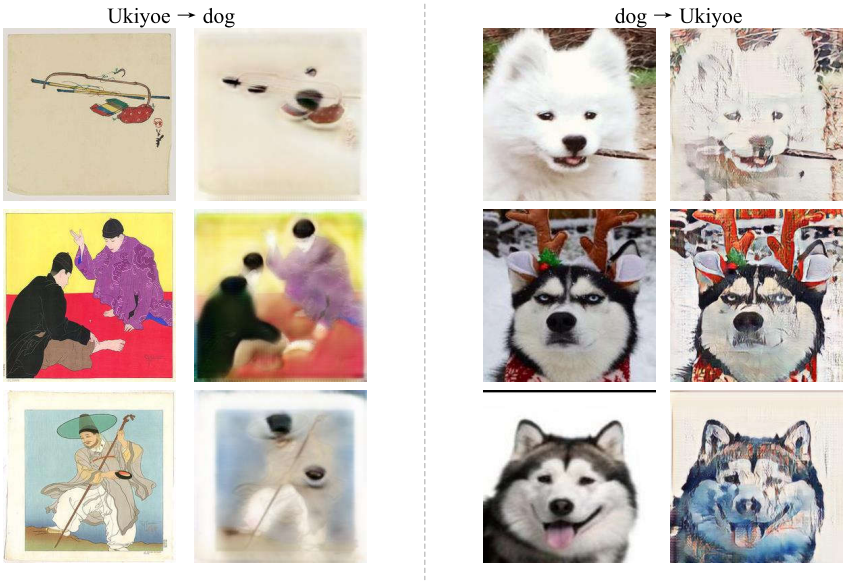


Figure 10: Ukiyo-e to dog

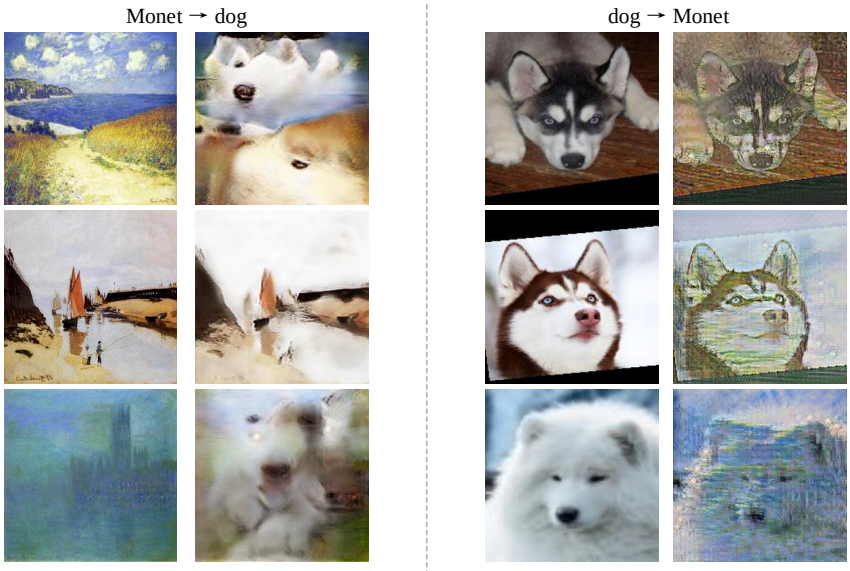


Figure 11: Monet to dog

5.3 Inter-Class Style Transfer

We also experimented with inter-class domains. We show our results of Ukiyoe to dog and Picasso to zebra in Figure 10 and 12.

For Ukiyoe to dog, an interesting phenomenon is that the network does not figure out what should be transferred to dogs. In Figure 10, the network is trying to map some objects to Samoyed's eyes, nose and mouth. For example, it maps some black dots to eyes and red bag to mouth. However, for dog to Ukiyoe, the mapping works well by transferring dogs to Ukiyoe-like style.

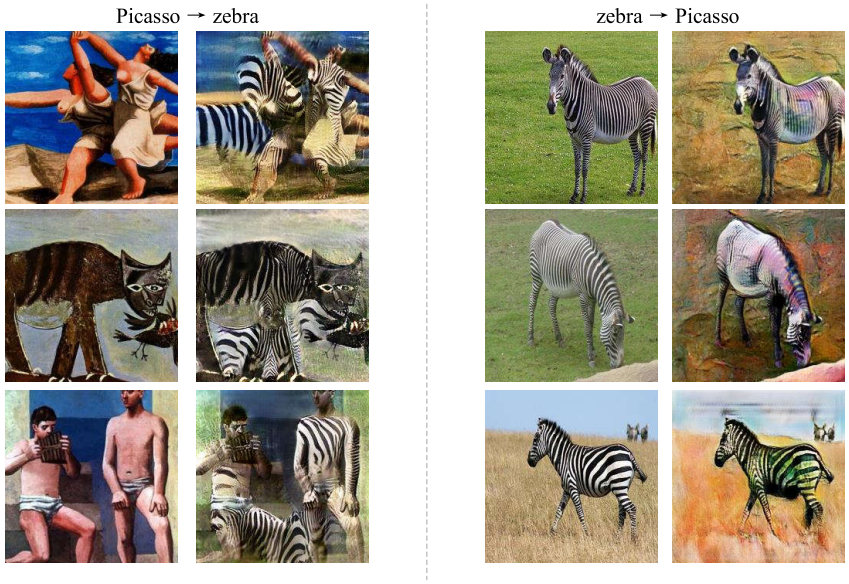


Figure 12: Picasso to zebra

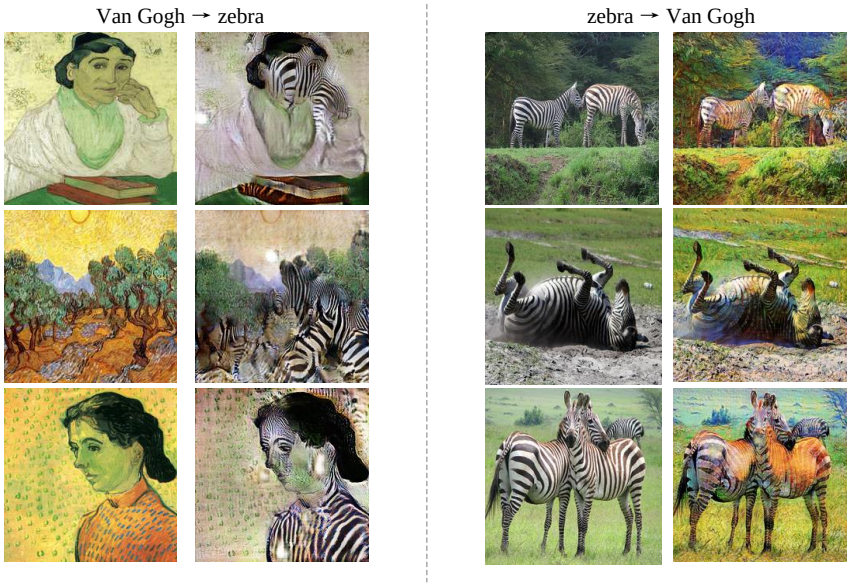
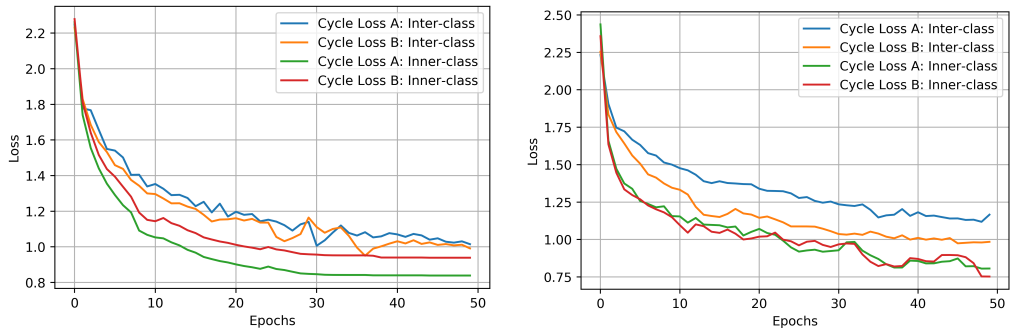


Figure 13: Van Gogh to zebra

For Picasso to zebra, the model adds zebra stripes to objects, this is similar to horse to zebra transfer. For zebra to Picasso, the images are added some Picasso's painting style.

The reason behind this observation is that, the gap between inter-class domains is larger than between inner-class domains. The qualitative results above show a significant difference. We also report the training cycle losses in Figure 14. As we can see from the loss curves, inner-class loss has a smoother and lower training loss, while inter-class loss converges slower, and has fluctuations during the training. This is one evidence of that the domain gap between inter-class is bigger.

As a result, the transfer between inter-class domains is hard to learn, since the domain gap is bigger than the inner-class domains.



(a) The losses of Van Gogh to photo (inner-class) and Van Gogh to zebra (inter-class)

(b) The losses of Picasso to photo (inner-class) and Picasso to zebra (inter-class)

Figure 14: A comparison between inner-class loss and inter-class loss.

6 Conclusion

In this project, we implemented CycleGAN from scratch and experimented its ability on different domains. For inner-class transfer, the learned mappings perform well for landscapes to artworks and seasons, but slight poor for animal transfer. Animals usually have more attributes than the pure landscapes. For inter-class transfer, it is easy to add painting styles to regular photos, but it is hard to make artworks look like real world scenes. The reason is that artworks usually contain objects that are highly different from real objects.

7 Contribution

Shujie Chen Shujie conducted part of the experiments of inner-class style transfer.

Felix Gabler Felix implemented the code for training and tests. He also conducted experiments of inner-class style transfer.

Yongchuang Huang Yonchuang helped with implementing the CycleGAN model, and did final project presentation.

Jingpei Lu Jingpei implemented the adversarial loss function and cycle consistency loss function. He also helped with writing reports and making the presentation slides.

Yiran Xu Yiran implemented generator and discriminator of the CycleGAN. He also conducted some experiments of inner-class and inter-class style transfer.

References

[1] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[2] Hsin-Ying Lee et al. “Diverse image-to-image translation via disentangled representations”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 35–51.

[3] Ming-Yu Liu and Oncel Tuzel. “Coupled generative adversarial networks”. In: *Advances in neural information processing systems*. 2016, pp. 469–477.

[4] Yusuf Aytar et al. “Cross-modal scene networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.10 (2017), pp. 2303–2314.

[5] Ashish Shrivastava et al. “Learning from simulated and unsupervised images through adversarial training”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2107–2116.

[6] Michael J. Wilber et al. “BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[7] Aaron Hertzmann et al. “Image analogies”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 327–340.

[8] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[9] Patsorn Sangkloy et al. “Scribbler: Controlling deep image synthesis with sketch and color”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5400–5409.

[10] Leon A Gatys et al. “Controlling perceptual factors in neural style transfer”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3985–3993.

[11] Levent Karacan et al. “Learning to generate images of outdoor scenes from attributes and semantic layouts”. In: *arXiv preprint arXiv:1612.00215* (2016).

[12] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[13] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[14] Emily L Denton, Soumith Chintala, Rob Fergus, et al. “Deep generative image models using a laplacian pyramid of adversarial networks”. In: *Advances in neural information processing systems*. 2015, pp. 1486–1494.

[15] Scott Reed et al. “Generative adversarial text to image synthesis”. In: *arXiv preprint arXiv:1605.05396* (2016).

[16] Michael Mathieu, Camille Couprie, and Yann LeCun. “Deep multi-scale video prediction beyond mean square error”. In: *arXiv preprint arXiv:1511.05440* (2015).

[17] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. “Generating videos with scene dynamics”. In: *Advances in neural information processing systems*. 2016, pp. 613–621.

[18] Jiajun Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in neural information processing systems*. 2016, pp. 82–90.

[19] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.