"Deep Fakes" using Generative Adversarial Networks (GAN)

Tianxiang Shen
UCSD
La Jolla, USA
tis038@eng.ucsd.edu

Ruixian Liu
UCSD
La Jolla, USA
rul188@eng.ucsd.edu

Ju Bai
UCSD
La Jolla, USA
jub010@eng.ucsd.edu

Zheng Li
UCSD
La Jolla, USA
zhl153@eng.ucsd.edu

Abstract

"Deep Fakes" is a popular image synthesis technique based on artificial intelligence. It is more powerful than traditional image-to-image translation as it can generate images without given paired training data. The goal of "Deep Fakes" is to capture common characteristics from a collection of existed images and to figure out a way of enduing other images with those characteristics, e.g. shapes and styles. Generative adversarial networks (GANs) provide us an available way to implement "Deep Fakes".

In this project, we use a Cycle-GAN network which is a combination of two GAN networks . The loss can be divided into 2 parts: total generator loss $\mathcal{L}_G$ and discriminator loss $\mathcal{L}_D$, where $\mathcal{L}_G$ includes a cycle-consistency loss $\mathcal{L}_{\text{cyc}}$ to ensure the images in input and output domains are mapped in a reasonable way. We calculate and back-propagate the 2 losses iteratively to train the Cycle-GAN.

We implement our Cycle-GAN for two objectives - object transfiguration and style transfer. In the former task we translate images of handbags and backpacks mutually and in the latter one we transform photos to other art styles such as Van Gogh oil painting and Ukiyo-e. Making use of PyTorch framework, the results of generated images are relatively satisfying.

1. Introduction

1.1. Background

Image-to-image translation has been researched for a long time by scientists from fields of computer vision, computational photography, image processing and so on. It has a wide range of applications for entertainment and design-assistance. The basic principle of image translation can be deemed as changing features of the input images in some specific ways while keeping other features unchanged. The method of feature change can be tuned manually or automatically. For automatic change, a process based on machine learning should be included for learning a general transfer method from a large training data set of numerous images to ensure the translation is reasonable for various inputs [1, 2, 3].

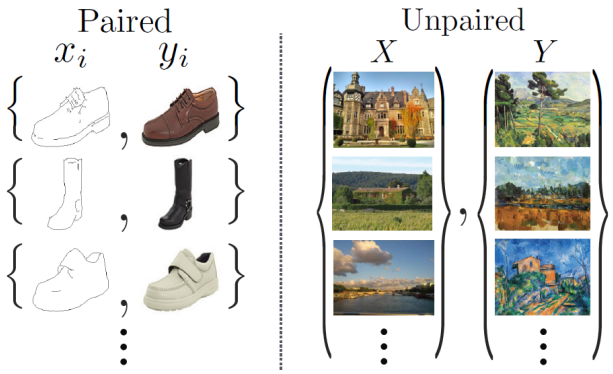

Figure 1. A comparison between paired training images and unpaired training images [4]

However, during the training period, many training models require paired training images. Yet obtaining paired training data can be difficult and expensive. For example, building image pairs manually may be resource-consuming, but undefined output will emerge without paired constraints. Inspired by Zhu et al.'s work [4], we implement a Cycle-GAN system that can learn to translate between images without paired training images as inputs(Figure 1, right). We test this Cycle-GAN system in two translation tasks, namely object transfiguration that translates images between handbags and backpacks and style transfer which transforms between real photographs and different art styles such as oil painting and Ukiyo-e.

1.2. Previous Works

Image-to-image translation has been investigated for a long time. Previously, it is usually carried out by manual or semi-automatic methods [5]. Such operations include enhancement of the contrast of pictures, image smoothing strategies, noise reduction techniques, and so on. In 2014, Goodfellow et al. developed image-to-image translation by

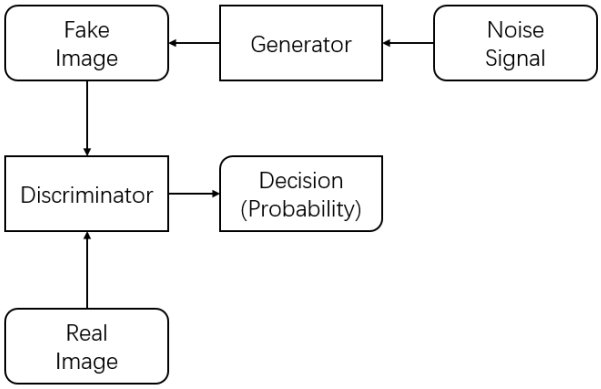

Figure 2. A block diagram of a GAN

using a generative adversarial network (GAN) significantly [6]. The GAN system consists of a generator that generates images from random noises and a discriminator that judges whether an input image is authentic or produced by the generator. The two components are functionally adversarial, and they play two adversarial roles like a forger and a detective literally. After the training period, the generator can produce fake images with high fidelity. In 2015, Radford et al. used a deep convolutional generative adversarial network (DCGAN) to implement a process of unsupervised representation learning and got a satisfactory result [7]. In this network, the generator and discriminator are both convolutional neural networks (CNNs). Later in 2017, Zhu et al. proposed a Cycle-GAN network to build an unpaired image-to-image translation [4]. The Cycle-GAN contains two GAN networks, and other than the loss in the traditional GAN network, it also included a cycle-consistency loss to ensure any input is mapped to a relatively reasonable output.

2. Physical and Mathematical framework

The framework we used in this project is a Cycle-GAN based on deep convolutional GANs.

2.1. Generative Adversarial Networks (GAN)

The basic module for generating fake images is a GAN. A block diagram of a typical GAN network is shown in Figure 2. A GAN network is consisted of a generator and a discriminator. During the training period, we use a data set X which includes a large number of real images x under a distribution of p_{data}. The generator G aims to produce images $G(z)$ which are similar to real images x where z are noise signals under a distribution of p_z. Meanwhile, the discriminator D aims to distinguish images generated by G from real images x. The probability that an input image $input$ determined by D as a real image rather than a fake image generated by G is denoted as $D(input)$. Obviously $D(input) \in [0, 1]$. We train D to maximize the probability that D assigns the correct labels to both real images and fake images generated by G, while G is being trained to minimize the probability that its outputs are determined by D as fake images, i.e. to minimize $1 - D(G(z))$. This two-player minimax game between D and G can be expressed as the following value function [6]:

$$\min_G \max_D \mathcal{V}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

After a sufficient training, the generator should be able to produce quite realistic images by using noise signals z. Meanwhile, the discriminator's capability of distinguishing fake images from real ones will also be strengthened.

2.2. Deep Convolutional GAN (DCGAN)

A convolutional GAN with a set of architecturally topological constraints for stability of training is called a DCGAN [7]. Both the generator and the discriminator of a DCGAN are CNNs. In our work, the generator of a DCGAN is a CNN which contains three down sampling layers, six residual layers and three up sampling layers (Figure 3), and the discriminator is another CNN which contains four convolutional layers (Figure 4).

2.3. Cycle-GAN

Although a DCGAN structure can fabricate images after a sufficient training period, the translation does not guarantee that an individual input x is mapped to an output y in a meaningful way, especially the mapping is not a bijection, i.e. all the inputs are mapped to the same output frequently, which is called a "mode collapse" [8]. Hence, Inspired by Zhu et al.'s work [4], we use a Cycle-GAN which is shown in Figure 5 for this project.

In a Cycle-GAN, there are two DCGANs, i.e. two generators and two discriminators. During the training period, two different sets of numerous images are sent as inputs to two DCGANs respectively. We denote these two domains as X and Y. For one of the DCGANs, input images x are members of domain X, and fake images generated from noise signals should be similar to images in domain Y rather than ones in domain X, which is different from a traditional GAN. We denote the generator of this GAN as G, and images generated by G as $G(x)$. We say that domain Y is the target domain of this GAN. Similarly, for the other GAN, input images are y which belong to domain Y, and its generator(denoted as F) should also generate images(denoted as $F(y)$) which are difficult to be distinguished from images in domain X. Hence, X is the target domain of this GAN. The discriminator of the GAN with generator G is denoted as D_Y since its goal is to distinguish fake images $G(x)$ from

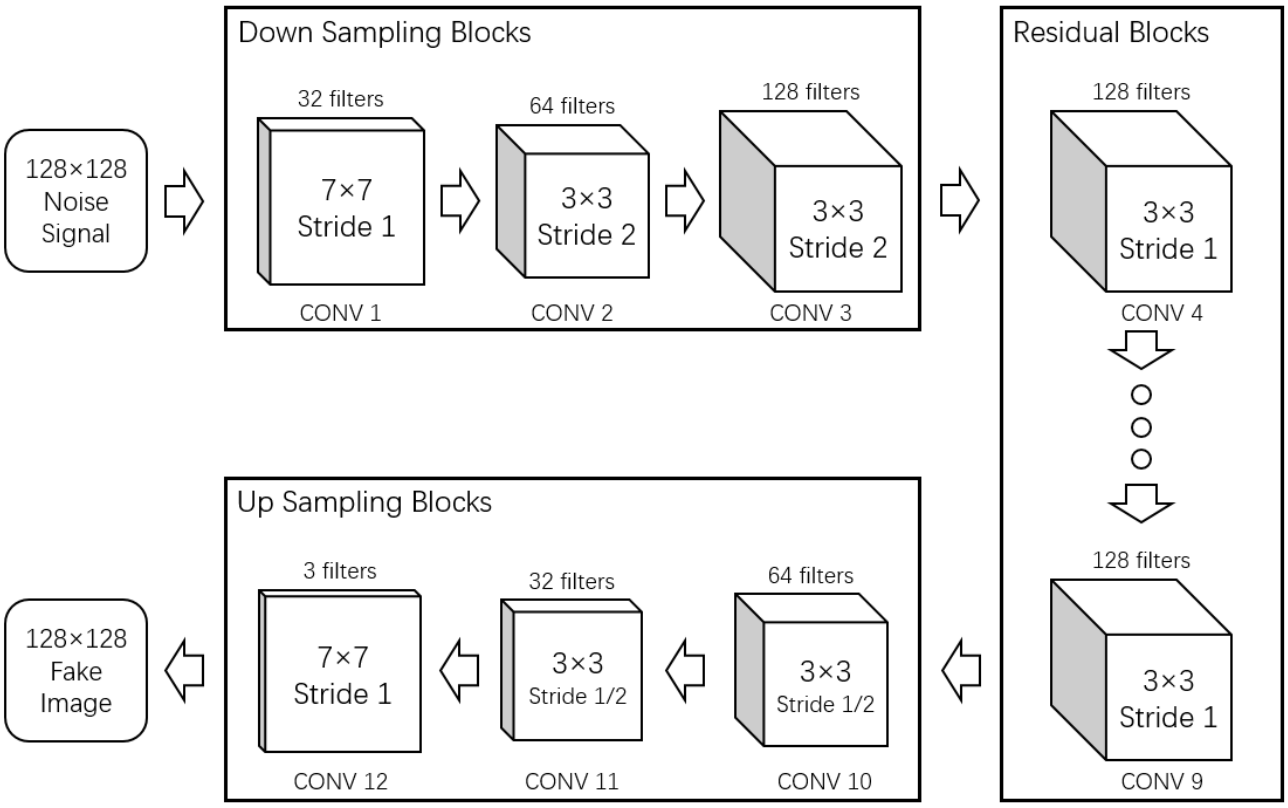

Figure 3. A block diagram of a generator in a DCGAN

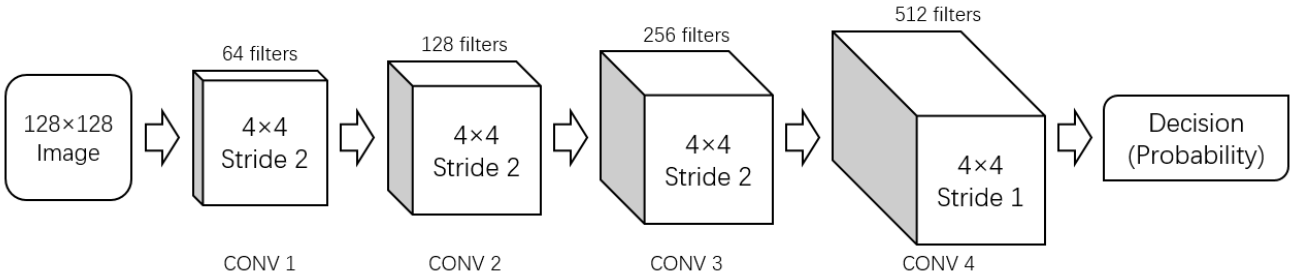

Figure 4. A block diagram of a discriminator in a DCGAN

real images in domain Y, and similarly the discriminator of the other GAN is denoted as D_X.

We introduce the cycle consistency loss $\mathcal{L}_{\text{cyc}}$ to ensure that if we translate an image from one domain to the other and then back to this domain again, finally the image should return to the place where it left [4], which means the mapping the input and the output is reasonable. The cycle-consistency losses contains two parts. One is called forward cycle-consistency loss that ensures $F(G(x)) \approx x$ (translate an input image x from domain X to domain Y via mapping G, and then back to domain X via mapping F), and the other is called backward cycle-consistency loss that ensures $G(F(y)) \approx y$.

Meanwhile, it is also helpful to introduce an additional loss to encourage mapping G and F to preserve color composition between inputs and outputs [4, 9]. We denote this loss as $\mathcal{L}_{\text{identity}}$ which aims to regulate the generator to be close to an identity mapping when real images in the target domain are provided to the generator as inputs[4].

Since there are two losses for each DCGAN, i.e. an adversarial loss and a discriminator loss, we merge the adversarial losses of the GANs together to obtain an adversarial

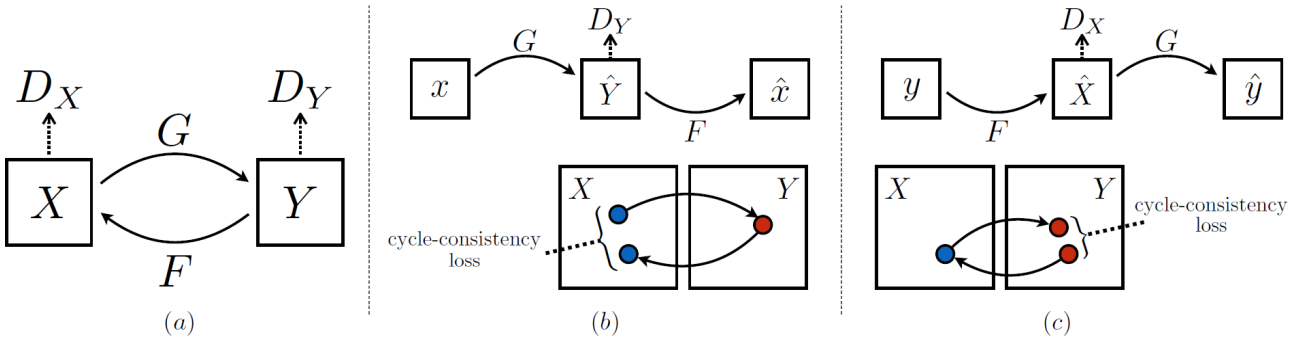

Figure 5. A demonstration of cycle-GAN [4]

Figure 6. Demonstration of results for handbag-backpack translation using Cycle-GAN. Left column for real images and right column for generated images

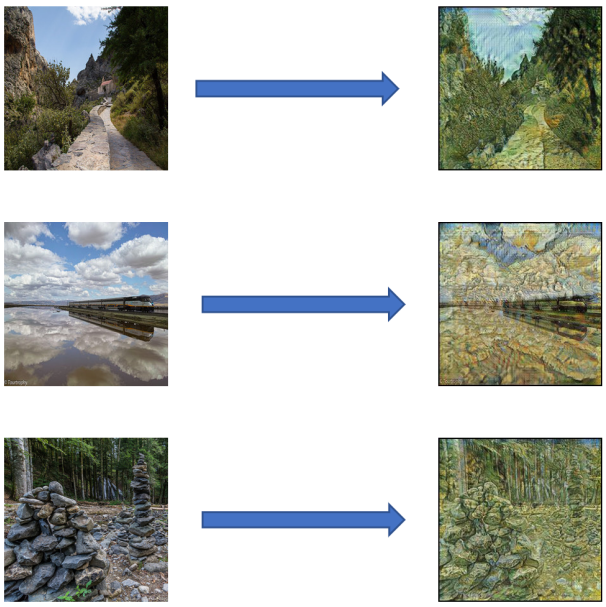

Figure 7. Demonstration of results for transferring photos to Van Gogh oil paintings.

loss for the whole cycle-GAN as $\mathcal{L}_{\text{GAN}}$, and we combine the discriminator losses together to get the discriminator loss for the whole network as $\mathcal{L}_D$. Summarizing, there are four losses for our cycle-GAN:

$$\begin{aligned}\mathcal{L}_{\text{GAN}} = &\mathbb{E}_{x\sim p_{\text{data}}(x)}[(1-D_Y(G(x)))^2] \\ &+\mathbb{E}_{y\sim p_{\text{data}}(y)}[(1-D_X(F(y)))^2]\end{aligned} \tag{2}$$

$$\begin{aligned}\mathcal{L}_D = &\mathbb{E}_{x\sim p_{\text{data}}(x)}[D_Y(G(x))^2] \\ &+\mathbb{E}_{y\sim p_{\text{data}}(y)}[D_X(F(y))^2] \\ &+\mathbb{E}_{x\sim p_{\text{data}}(x)}[(1-D_X(x))^2] \\ &+\mathbb{E}_{y\sim p_{\text{data}}(y)}[(1-D_Y(y))^2]\end{aligned} \tag{3}$$

$$\begin{aligned}\mathcal{L}_{\text{identity}} = &\mathbb{E}_{y\sim p_{\text{data}}(y)}[||G(y)-y||_1] \\ &+\mathbb{E}_{x\sim p_{\text{data}}(x)}[||F(x)-x||_1]\end{aligned} \tag{4}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}} = &\mathbb{E}_{x\sim p_{\text{data}}(x)}[||F(G(x))-x||_1] \\ &+\mathbb{E}_{y\sim p_{\text{data}}(y)}[||G(F(y))-y||_1]\end{aligned} \tag{5}$$

For these four losses, $\mathcal{L}_{\text{cyc}}$ and $\mathcal{L}_{\text{identity}}$ are ℓ_1-norm losses, meanwhile $\mathcal{L}_{\text{GAN}}$ and $\mathcal{L}_D$ are MSE losses. For simplicity, we combine $\mathcal{L}_{\text{cyc}}$, $\mathcal{L}_{\text{identity}}$ and $\mathcal{L}_{\text{GAN}}$ together as a total generator loss $\mathcal{L}_G$:

$$\mathcal{L}_G = \lambda_1\mathcal{L}_{\text{GAN}} + \lambda_2\mathcal{L}_{\text{identity}} + \lambda_3\mathcal{L}_{\text{cyc}} \tag{6}$$

where λ_1, λ_2 and λ_3 are coefficients that control relative importances of the three losses. We also divide $\mathcal{L}_D$ into two

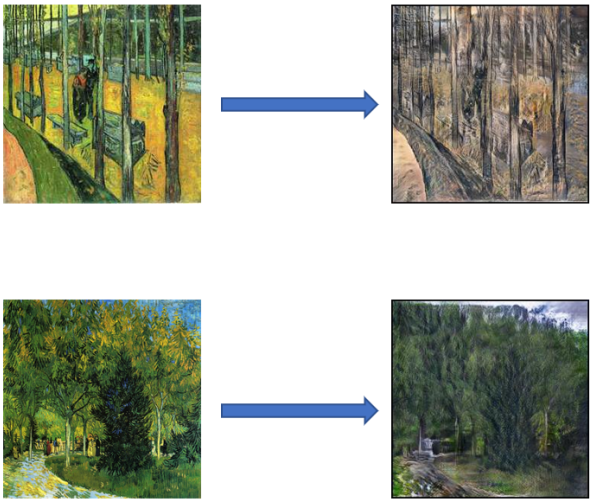

Figure 8. Demonstration of results for transferring Van Gogh oil paintings to photos.

sections, i.e. $\mathcal{L}_{D_X}$ and $\mathcal{L}_{D_Y}$ to express the losses of the two discriminators respectively:

$$\begin{aligned}\mathcal{L}_{D_X} &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[(1-D_X(x))^2] \\ &+ \mathbb{E}_{y\sim p_{\text{data}}(y)}[D_X(F(y))^2]\end{aligned} \tag{7}$$

$$\begin{aligned}\mathcal{L}_{D_Y} &= \mathbb{E}_{y\sim p_{\text{data}}(y)}[(1-D_Y(y))^2] \\ &+ \mathbb{E}_{x\sim p_{\text{data}}(x)}[D_Y(G(x))^2]\end{aligned} \tag{8}$$

In each epoch of the training period, we firstly calculate and back propagate $\mathcal{L}_G$, then calculate and back propagate $\mathcal{L}_{D_X}$ and finally calculate and back propagate $\mathcal{L}_{D_Y}$, which is shown in Algorithm 1, where E denotes the number of training epochs, N denotes the number of training images, and $learnRate_{\text{init}}$ denotes the initial learning rate. After first E_0 epochs, learning rate will start decreasing linearly until it equals to zero in the last epoch.

Algorithm 1 Algorithm to train a cycle-GAN

$learnRate \leftarrow learnRate_{\text{init}}$
for $e = 0 \rightarrow E-1$ **do**
  for $i = 0 \rightarrow N-1$ **do**
    Calculate $\mathcal{L}_G$
    Back-propagate $\mathcal{L}_G$
    Calculate $\mathcal{L}_{D_X}$
    Back-propagate $\mathcal{L}_{D_X}$
    Calculate $\mathcal{L}_{D_Y}$
    Back-propagate $\mathcal{L}_{D_Y}$
  end for
  if $e \geq E_0$ **then**
    $learnRate \leftarrow learnRate_{\text{init}} * (1 - (e - E_0)/(E - E_0))$
  end if
end for

3. Experiments and Results

3.1. Dataset

Our implementation of Cycle-GAN aims to solve two different problems, so we use two different sets of images for training.

For object transfiguration (handbag-backpack transformation), the images we used are collected from Google Image. We arrange 760 images as handbag inputs where 660 images for training and 100 images for testing, and 950 images as backpack inputs where 801 images for training and 149 images for testing. For style transfer (photo-Van Gogh oil painting/Ukiyo-e transfer), we make use of data sets from UC Berkeleys repository (available at UCB Dataset) to train our model.

3.2. Object Transfiguration

During training process, we set $\lambda_1 = 1$, $\lambda_2 = 5$ and $\lambda_3 = 10$, and the $\mathcal{L}_D$ is multiplied by a coefficient of 0.5. The losses while training are shown in Figure 9, where subfigures (a)-(e) show $\mathcal{L}_G$, $\mathcal{L}_D$, $\mathcal{L}_{\text{GAN}}$, $\mathcal{L}_{\text{identity}}$ and $\mathcal{L}_{\text{cyc}}$ respectively.

The results of the handbag-backpack translation are shown in Figure 6. The figures in the left column are real images while the figures in the right column are fake images produced by the generator.

3.3. Style Transfer

We use the same settings used in handbag-backpack translation for this task. And the losses while training is shown in Figure 10.

The results of the photo-Van Gogh oil painting translation are shown in Figure 7, 8. Similarly to Figure 6, the images in the left column are real and those in the right column are generated. However, the satisfactory part are all for the transferring from photos to oil paintings. The photos generated from oil paintings look not good because of the lack of realness.

Also, the Figure 11 shows the losses decreasing process during the training of the Cycle-GAN for transferring between photos and Ukiyo-e.

4. Result Discussion

The results of transfer photos to Ukiyo-e are demonstrated in Figure 12, which seems rather satisfactory. However, similar to transferring Van Gogh paintings into photos, the results of transferring Ukiyo-e to photos are not satisfactory. (Figure 13)

Figure 9, 10 and 11 show that all losses decrease continuously after each iteration and converge to constants at 200

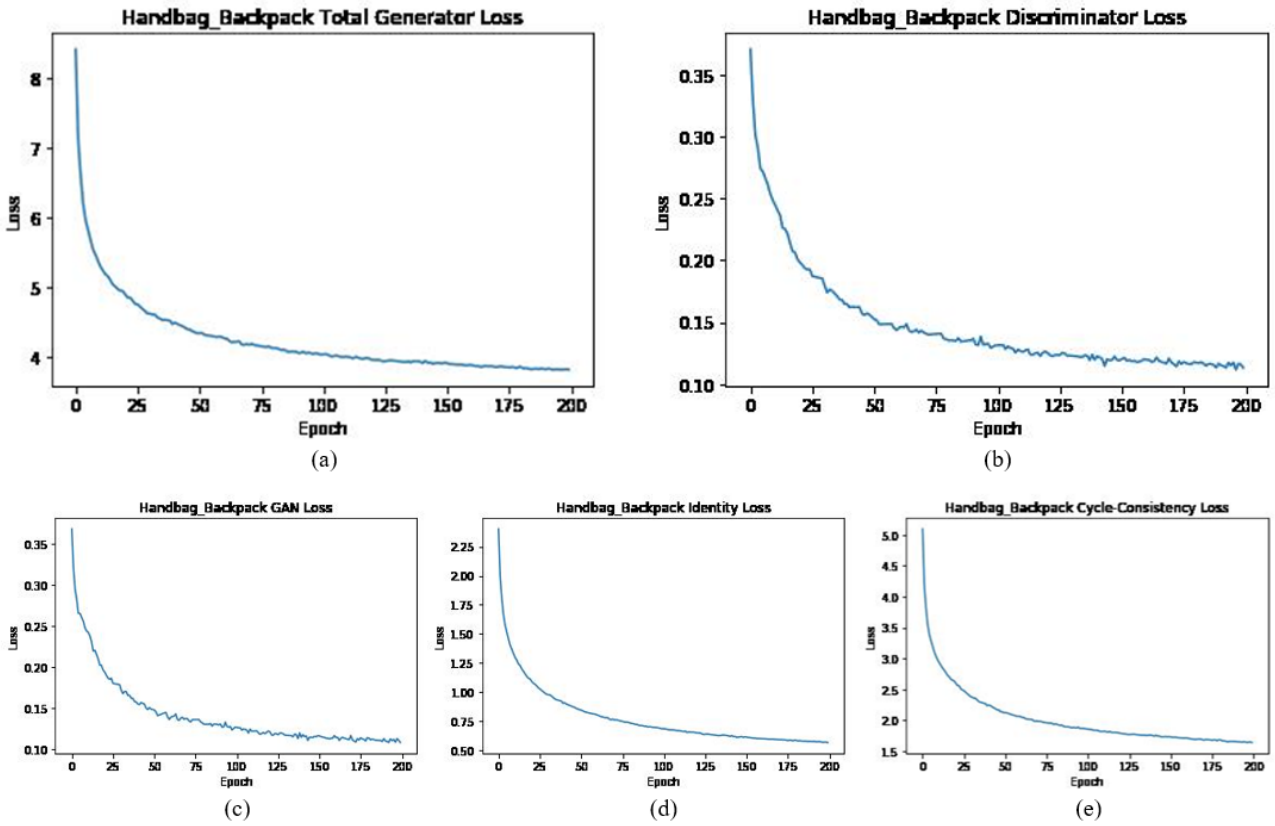

Figure 9. The losses decreasing process while training cycle-GAN for handbag-backpack transfiguration. (a)-(e) show $\mathcal{L}_G$, $\mathcal{L}_D$, $\mathcal{L}_{\text{GAN}}$, $\mathcal{L}_{\text{identity}}$ and $\mathcal{L}_{\text{cyc}}$ respectively

epochs when the algorithm is done.

4.1. Object Transfiguration

We use perceptual studies to evaluate the performance of the network, which is to let people distinguish real and fake images given a pair of input and output. From Figure 6, we can see that those handbags generated by the network in the right column are very similar to real bags. They are different from real backpacks in the left column in shape and style yet reserve some special characteristics. For example, bags in the second pair both have two vertical stripes and the third pair have similar zipper design. We showed our result in the presentation and most people cannot tell fake images from real ones.

Although our network produces compelling results most of the time, it may produce noisy images in some cases. This may be caused by the distribution characteristics of training datasets, or by the structure of the generator. There is still a little instability in this unsupervised system.

4.2. Style Transfer

We trained the Cycle-GAN with Van Gogh's oil paintings and tested some photos using this network. As shown in Figure 7, the output images,as expected, were in the style of oil painting, with colors and details kept. There were blurry and pale outcomes occasionally. And the situation is similar regarding the Ukiyo-e. The generated images are all in warm colors and are consist of a lot of very little blocks when observed in detail, which are just the typical features of Ukiyo-e.

However, when we try to transfer Ukiyo-e or oil paintings into photos, we did not get images like photos. Sometimes we may even get the original images. The reason for this is that GANs capture common characteristics of the dataset and generate images in this way. Both oil paintings and Ukiyo-e pictures have distinctive characteristics. The characteristics of Van Gogh oil paintings are that they use bright and hot colors extensively, as well as sporty, continuous, wave-like flowing strokes. And the Ukiyo-e pictures are always consist of warm colors such as red, yellow and light green and covered by tiny color blocks mainly because of the characteristics of the material that the Ukiyo-e pic-

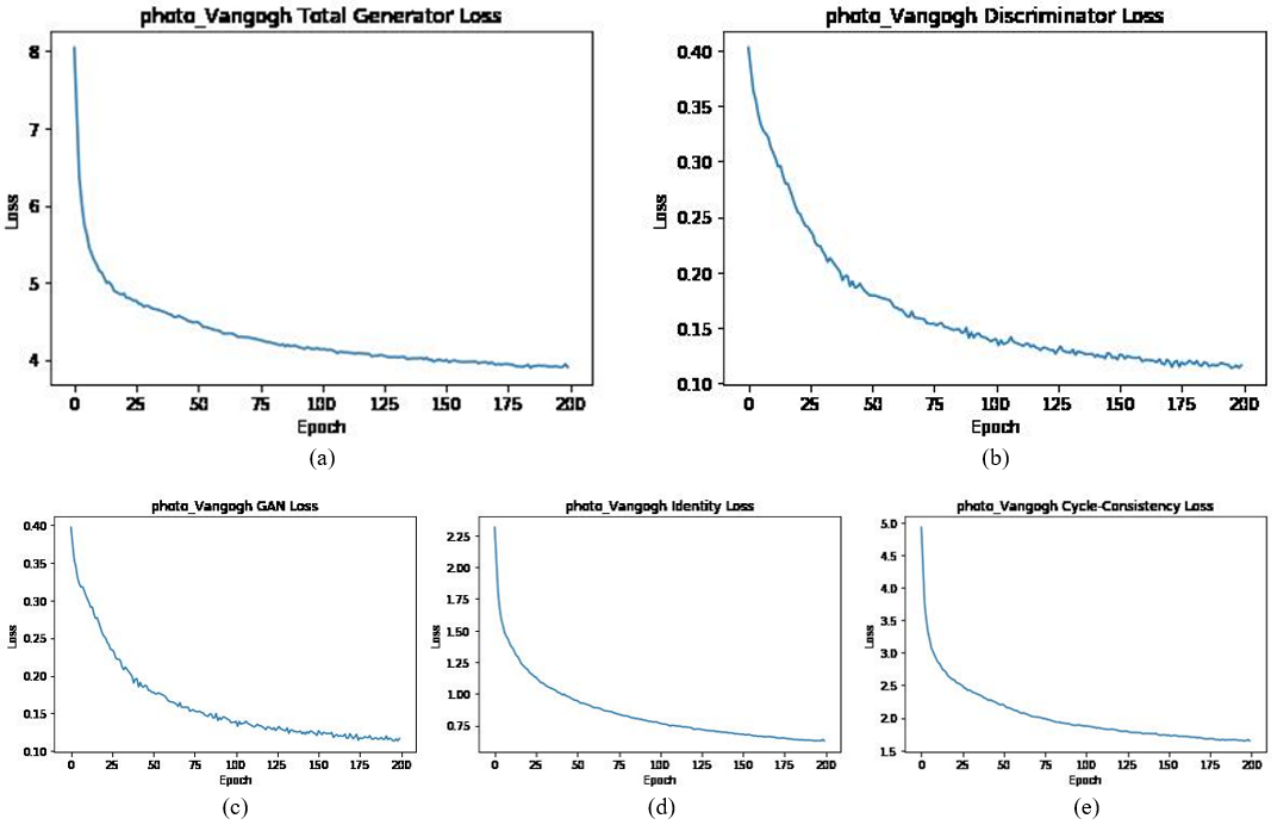

Figure 10. The losses decreasing process while training cycle-GAN for photos and Van Gogh oil paintings transferring. (a)-(e) show $\mathcal{L}_G$, $\mathcal{L}_D$, $\mathcal{L}_{\text{GAN}}$, $\mathcal{L}_{\text{identity}}$ and $\mathcal{L}_{\text{cyc}}$ respectively

tures are painted on. However, normal photos have nothing in common. The network cannot figure out a fixed pattern of photos to do image transfer and thus produces bad fakes.

5. Future Work

One drawback of GAN neural network is it needs multiple data to train it and the speed of the training process is rather slow. We are still finding the best optimizer functions and batch size for the GPU to balance the robustness and efficiency of our GAN networks.

GANs have many interesting applications in all kinds of industry apart from generating new objects and style transfer. One typical application is season transfer. We can train the network using photos of different seasons and turn it into a season transferring system. Likewise, our method can be implemented as a photo enhancement technique. It can generate photos with shallower depth of field given photos captured by DLSR camera.

Recently, GANs have modeled patterns of motion in video. They have also been used to reconstruct 3D models of objects from images. There will be an increasing number of improved GANs come into play in the near future.

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017. 1

[2] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 1

[3] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016. 1

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017. 1, 2, 3, 4

[5] Rafael C Gonzalez and Richard E Woods. Image processing. *Digital image processing*, 2, 2007. 1

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

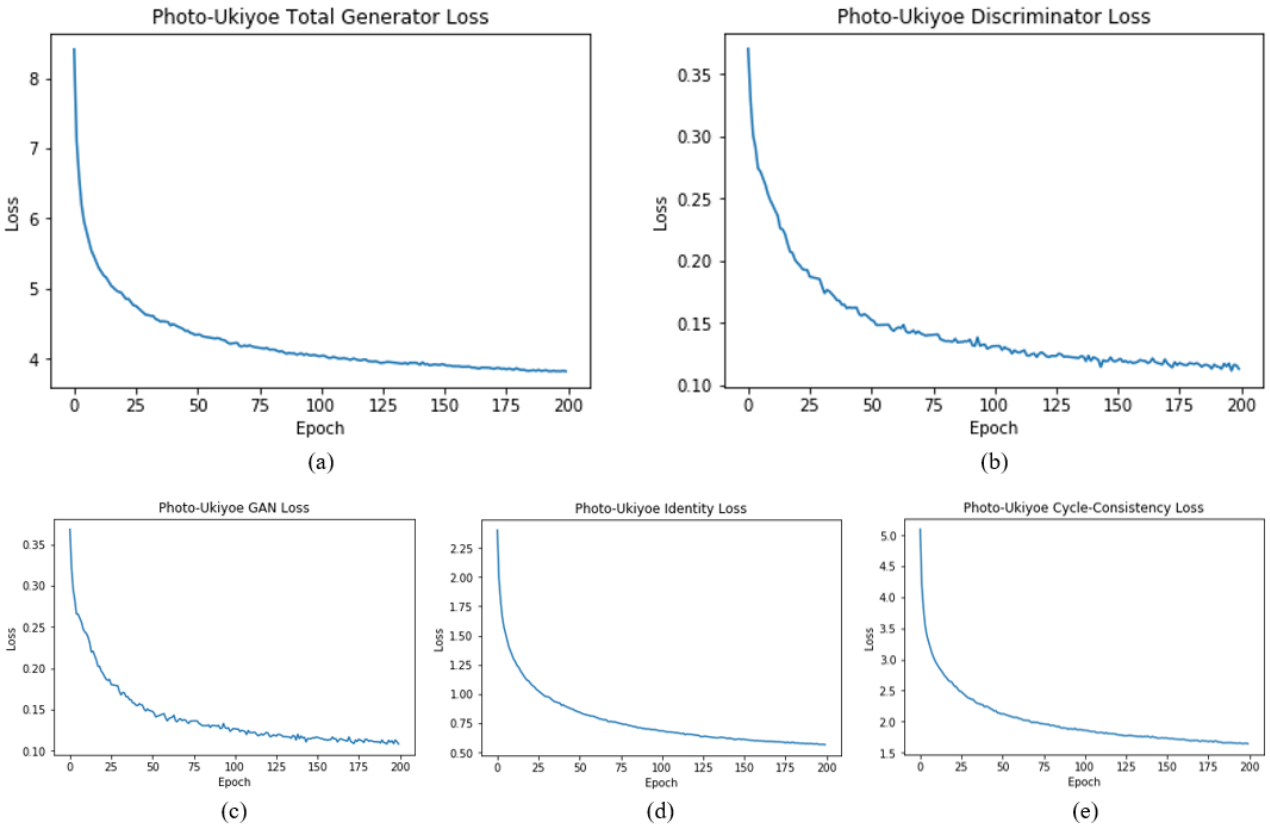

Figure 11. The losses decreasing process while training cycle-GAN for photos and Ukiyo-e transferring. (a)-(e) show $\mathcal{L}_G$, $\mathcal{L}_D$, $\mathcal{L}_{\text{GAN}}$, $\mathcal{L}_{\text{identity}}$ and $\mathcal{L}_{\text{cyc}}$ respectively

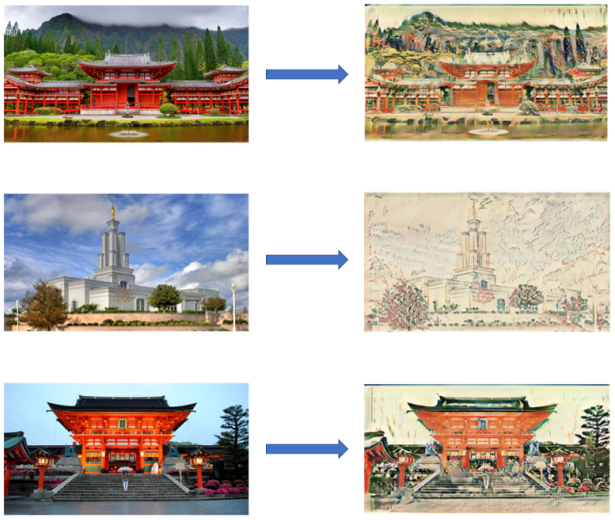

Figure 12. Demonstration of results for transferring photos to Ukiyo-e.

Figure 13. Demonstration of results for transferring Ukiyo-e to photos.

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional genera-

tive adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[8] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[9] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016. 3