AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Computer Science And Engineering

Cartoon-to-real: An Approach to Translate Cartoon to Realistic Images using GAN

by

15.01.04.147: Labiba Kanij Rupty

15.01.04.111: Arefeen Sultan

15.01.04.127: Nahid Pranto

15.01.04.133: Sayed Khan Shuvo

Supervisor: Mr. Mohammad Imrul Jubair(Asst. Prof.)

June 2019

AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Abstract

Mr. Mohammad Imrul Jubair(Asst. Prof.)
CSE

by
15.01.04.147: Labiba Kanij Rupty
15.01.04.111: Arefeen Sultan
15.01.04.127: Nahid Pranto
15.01.04.133: Sayed Khan Shuvo

We show some works to translate cartoon images to real world images using Generative Adversarial Network(GAN). Existing GAN-based image-to-image translation methods which are trained on paired datasets are impractical as the data is difficult to accumulate. Therefore, in this paper we exploit the ideas of *cycle-consistency* and *shared-latent space assumption* method for image translation which needs an unpaired dataset. By applying these methods using different framework(CycleGAN,UNIT,SingleGAN) we compare and show that our model is able to generate meaningful real world images from cartoon images.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many computer vision problems can be considered as an image-to-image translation problem, mapping an image from one domain to another. For example, colorization [1] [2] (grayscale $\rightarrow$ color), super resolution [3] [4] (low resolution $\rightarrow$ high resolution), and style transfer [5] [6] [7] (image $\rightarrow$ style image).

Gatys et al.[5] proposed artistic style transfer in his work. By using convolutional neural networks, the author translated any input image into the style of a painting image. Later many works were done in image-to-image translation tasks e.g.[6] [7] where most of them used convolutional neural networks to do the synthesis.

Later, Generative Adversarial Network(GAN)[8] was proposed which provided astounding results in many tasks, e.g, image generation [9] [10], image editing[11], text2image[12], image inpainting[13], image-to-image translation tasks[14] [15] etc. Isola et al.[15] proposed a GAN based approach to transfer input of one domain,x to another domain,y by training on paired image dataset. But accumulating such paired images of both domains is hard and ineffective. We might not get paired images to train for many specific tasks of image-to-image translation. Recently, for training unpaired images, where there is no correspondence between domain x and domain y images, many authors published incredible works such as, CycleGAN[14], DualGAN[16], DiscoGAN [17], UNIT[18] etc.

In unsupervised setting, adversarial loss alone can't solve the infinite mapping problem in the target domain. Zhu et al.[14] proposed that, by using cycle consistency loss, the network would be able generate images where the possible infinite mappings of target domain would be reduced.

Similar work was proposed in UNIT[18], where the space-latent assumption implied that

the two corresponding sets of different domains can be mapped to a same latent representation in a shared-latent space. To exploit this assumption, the authors proposed a framework based on generative adversarial network(GAN) and variational autoencoders(VAEs).
The authors in CartoonGAN[19] paper made an approach in cartoon stylization, transferring a set of real world images to cartoon domain. For this approach, no pairing was required during the training. In this paper, we work on - *Cartoon-to-Real* image translation task. During the training procedure, no pairing of images was required. We extracted cartoon images from different cartoon movies and real images from internet, i.e. flickr. Using our Cartoon-to-Real, we achieve significant result in translating the cartoon images to realistic ones.

1.1 Problem domain

In recent times, there have been many works on remaking cartoon movies into real live action films e.g. *The Lion King(2019)*, *Beauty and the Beast(2017)* etc. Translating the cartoon images into real world images is a tedious and tiresome work for the film industry. Meanwhile, existing image editing software/algorithms is more costly for bringing CGI effects into films. Therefore, techniques that can automatically transform cartoon images into real world images are very helpful and tremendous amount of time can be saved. Tools such as this also provide as an additional component in photoshop and image editing works.
In our work, we do this translation of cartoon images into real world images. For doing this task, we did not need any pairing between two sets of images, cartoon and real world images. We exploited the ideas of cycle consistency[14] and shared-latent space assumption[18] and provided effective results in cartoon-to-real images translation.

1.2 Motivation

CartoonGAN[19] paper proposed a GAN based approach on transforming real world images in cartoon domain. We became interested to transfer the opposite of CartoonGAN, that is, transforming cartoon images to real world images. We see that, many recent movies of cartoons is being made live action film so we were interested in such domain.

1.2.1 Why GAN

Generative adversarial networks provide state of the art results in image-to-image translation tasks. Recently CycleGAN[14], UNIT[18], SingleGAN[20] provides astounding results in this domain. CycleGAN exploits the concept of *cycle consistency loss* where the infinite mappings between two domains is reduced. Similar to *cycle consistency loss* UNIT[18] proposed another assumption known as shared latent space, where both domains have same shared latent representation. These GAN based approaches provides more effective and efficient results than any other approaches.

1.3 Contribution

We implemented cartoon to real world images using Generative Adversarial Network on our unpaired image dataset. For the cartoon domain, we've collected almost 3.1K images scrapped from various movies, e.g. *Pokemon*, *My Neigbour Totoro* and *Kikis Delivery*. We used flickr dataset for the real images domain. Images are resized to 128128 resolution. These images,real and cartoon domain images, have no correlations between them We achieved significant results in translating the cartoon images to realistic ones.

Chapter 2

Background & Related Work

2.1 Generative Adversarial Networks

Generative Adversarial Network(GANs)[8] has become a global phenomenon in deep learning algorithms. The algorithm uses two networks, Generators and Discriminators, in a minimax algorithm situation where both of them tries to outperform another in a significant task e.g. image generation [9] [10], image editing[11], text2image[12], image inpainting[13], image-to-image translation tasks[14] [15] etc. Full objective of GAN function is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

where Generator G tries to generate images, whereas discriminator D discriminates the output whether it is fake or real. In the objective function, Generator tries to maximize the value of $D(G(z))$ such that it can fool the discriminator,and thus the gap betwen real and fake becomes minimum. The discriminator tries to maximize the term of $[logD(x)]$ and for the 2nd term of $[log(1-D(G(z))]$, discriminator tries to minimize it to 0, which means that the discriminator tries to recognize if the image is generated or real.

2.2 Image-To-Image Translation

Recently, GAN based approach has given tremendous results in image-to-image translation tasks. CycleGAN[14] proposed a cycle consistency loss to reduce the infinite

mappings of input images to any distribution in the target domain. Adversarial loss alone can't solve the random permutation mappings of target distribution, rather it helps the input image to be translated into target domain.
In UNIT[18] framework, the authors proposed a shared-latent space assumption, which denotes that the pair of corresponding images in different domains can be mapped to a same latent representation in a shared-latent space. By using the combination of generative adversarial network(GAN) and variational autoencoders(VAEs) the authors achieved great results in image-to-image translation tasks. In SingleGAN[20] framework, the authors used *cycle consistency loss* [14], where one generator is needed instead of using two generators[14]. The authors were able to achieve significant results in single domain and multi domain translation.

2.3 Baseline Models

2.3.1 CycleGAN

While many researchers have produced groundbreaking results such as [15] ,[21], [22] on image-to-image translation using paired data, there hasn't been much successful research using unpaired data. To resolve this case, CycleGAN [14] has played an influencial role by presenting an approach which translates one image of domain X to another domain Y without any paired training data. This translation is based on an assumption that if an image, x_i from domain X can generate a new image a new image y_i of another domain Y, eventually, the generated image, y_i can be mapped to X by generating a new image $\hat{x}$ where $x_i = \hat{x}$. To sum up, if G is the generator which translates into domain Y and F is for the next translation, we can write it as following -

$$G(x_i) = y_i, where\, x_i \in X, y_i \in Y \tag{2.2}$$

$$F(y_i) = \hat{x}, where\, \hat{x} \in X \tag{2.3}$$

Let's break down the ideas that were used to make it a successful research and discuss them one by one.

2.3.1.1 Adversarial Loss for CycleGAN

Previously, we have known about *Goodfellow et al.*[8] and how it has revolutionized the future of AI. As mentioned in section 2.1, we know that GAN[8] architecture works are based on *Adversarial Loss* which is just an extension of *Binary Cross-Entropy Loss* [Appendix .1]. However, in the case of CycleGAN[14], although *Adversarial Loss* has been used, *Binary Cross Entropy Loss* is not used. The reason is more related to the training inconsistency of GAN[8]. From *Mao et al.*, it is known that using *Least Squares Loss* [Appendix .2] shows more stability in training for CycleGAN[14] than using *Binary Cross-Entropy Loss.* So, equation **??** of Adversarial loss turns into the following equation, where c is an image from domain C and r from R:

$$For\, Generator\, G, L_{adv} = \frac{1}{m}\sum_{i=1}^{m}(1 - D_r(G(c)))^2$$

$$For\, Generator\, F, L_{adv_{cyc}} = \frac{1}{m}\sum_{i=1}^{m}(1 - D_c(F(r)))^2$$

However, using only *Adversarial Loss* is not enough to get the best result. This loss is *under-constraint* as it only limits the output to be of a specific domain and fails to limit a closely related output with respect to input. From fig **??**, we can see its demostration. The researchers of CycleGAN[14] uses an addition loss *Cycle Consitency Loss* to limit the output to be closely related to the input.

2.3.1.2 Cycle Consistency Loss

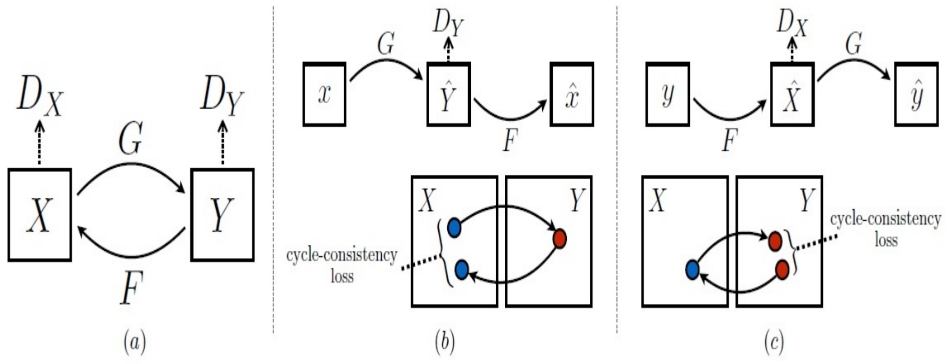

FIGURE 2.1: Two Mapping Functions with Cycle Consistency Loss in CycleGAN

The idea of *Cycle Consistency* goes way back. It is an idea of determining the transitivity of two images where second image is the reconstruction of the first image. This transitivity is here denoted as *Loss* in our case, where we use two *Cycle Consistency Loss* as *Forward & Backward Cycle Consistency Loss.*

For our *Cartoon-to-real* translation, if image, c of domain C is to be translated into domain R, through *Generator* G, there must be another *Generator* F to translate the newly translated image $G(c)$ into $\hat{c}$ with a view to reconstructing c. From figure ??, if this *Cycle Consistency Loss* is called *Forward - Cycle Consistency Loss*, the opposite is called *Backward - Cycle Consistency Loss.* It is defined using the following equations -

$$Forward\, Consistency\, Loss, L_{f\_cyc} = \frac{1}{m}\sum_{i=1}^{m}(F(G(c)) - c)$$

$$Backward\, Consistency\, Loss, L_{b\_cyc} = \frac{1}{m}\sum_{i=1}^{m}(G(F(r)) - r)$$

A question may arise on why using *Cycle Consistency Loss* solves the *under-constraint* issue. The intuition is that for a general mapping of two images, *Adversarial Loss* is great. However, it won't be able to specify the best image of the domain which should be mapped to the first image. On the other hand, *Cycle Consistency Loss* can do this job. Let's think of a scenario, where someone wants to translate *a garden image* to *Monet Painting* using GAN. To his surprise, he finds out that when used only *Adversarial Loss*, the machine translates the garden image with a random monet painting, kind of like figure ?? and on the other hand, when used *Cycle Consitency Loss*, it shows the same contents of the garden which seems to be painted by *Monet.* After some time, he finds out that, as *Cycle Consistency Loss* minimizes the reconstruction loss of the image, the machine is bound to choose an image which is pretty similar to the garden image, as its loss must be the least. So, we can say that, *Cycle Consistency Loss* binds the code to find out the best monet painting to be stylized.

So, the total loss is -

$$L(G, F, D_c, D_r) = L_{adv} + L_{f\_cyc} + L_{b\_cyc}$$

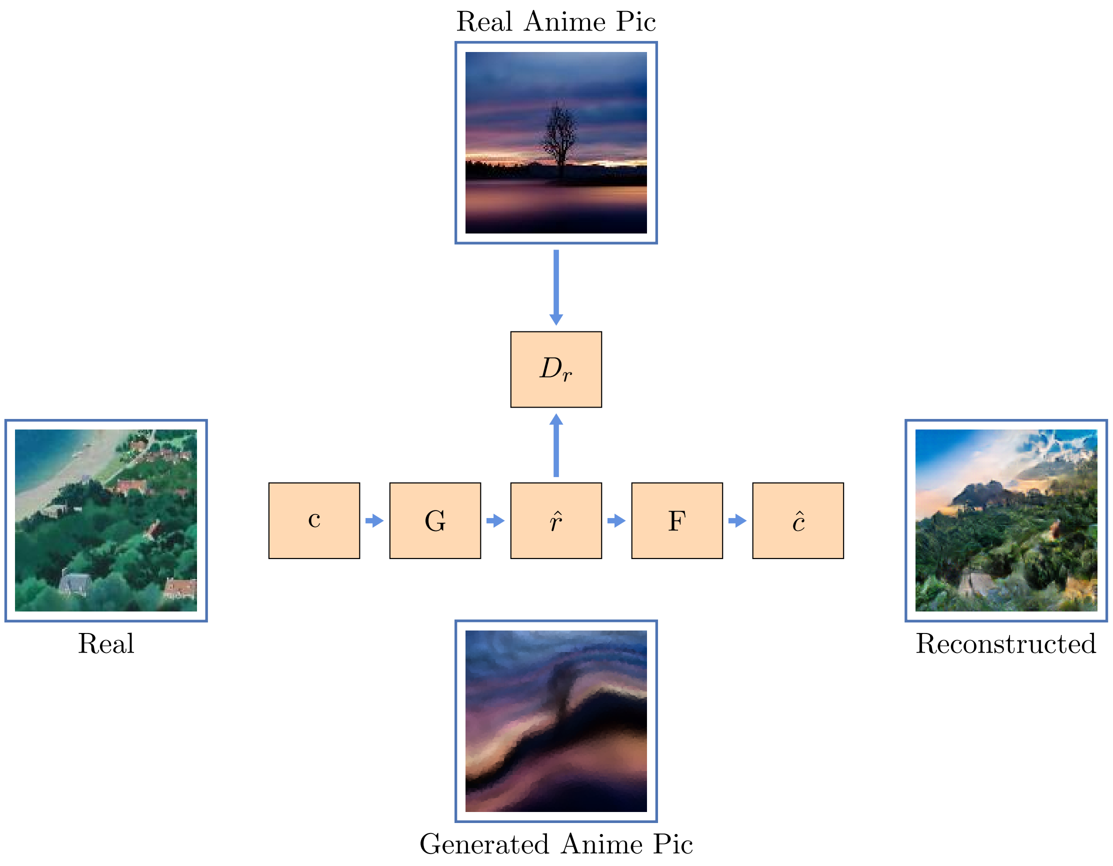

FIGURE 2.2: Architecture Of CycleGAN

2.3.1.3 Network Architecture & Training

CycleGAN[14] network consists of *two* **discriminator** networks and *two* **generator** networks. The **discriminator** networks are used as **PathGANs**. Each **Generator** has *three* sections: ENCODER, TRANSFORMER, DECODER [Appendix .3]. Let's take a deeper look into the architecture.

According to *Zhao et al.*[23], *L1 Loss* performs best for *Low frequency*[Appendix .4] details, such as *color-blotches, general tonal-distribution/contrast.* However, it fails at preserving *High Frequency*[Appendix .4] details, such as *crisp, edge, texture* etc. Due to the use of *L1 Loss* for *Cycle consistency loss*, the model tends to lose its texture and crisp details. To avoid this situation, **discriminator** network is forced to model *high frequency* structure. As a result, discriminator uses **PatchGAN** model which works as a fully collected layer checking a $N \times N$ patch of image every time whether that patch real or fake. In the case of CycleGAN, 70×70 pixel patch has been used. As texture and styles also work over patch, using **PatchGAN** preserve them. The architecture of **PatchGAN** is shown in Table 2.1.

TABLE 2.1: PatchGAN Architecture

Layers	Discriminators
1	CONV-(N64,K4,S2), LeakyRelu
2	CONV-(N128,K4,S2), InstanceNorm, LeakyRelu
3	CONV-(N256,K4,S2), InstanceNorm, LeakyRelu
4	CONV-(N512,K4,S2), InstanceNorm, LeakyRelu

We already know that each **Generator** consists of *three* architectures within itself. When the input is fed into the network, it directly reaches the encoder, which extracts the representation model of the inputs. It has *three* convolutional layers. After the final layer, it transforms the input into 256 channels and passes it to the *transformer* architecture. It is composed of a *Residual Block*[Appendix .5] which has 6 convolutional layers. The activated inputs are then expanded using **decoder** which uses *two deconvolutional layer*[Appendix. .6] to upsample the image and then a convolutional layer with 3 channels to output an **RGB** image. The architecture is shown in table 2.2.

TABLE 2.2: Generator Architecture of CycleGAN

Layers	Generators
1	CONV-(N64,K7,S1),InstanceNorm,Relu
2	CONV-(N128,K3,S2),InstanceNorm,Relu
3	CONV-(N256,K3,S2),InstanceNorm,Relu
4	residual-(N256,K3)
5	residual-(N256,K3)
6	residual-(N256,K3)
7	residual-(N256,K3)
8	residual-(N256,K3)
9	residual-(N256,K3)
10	DCONV-(N128,K3,S1/2),fractional,strided, InstanceNorm, Relu
11	DCONV-(N64,K3,S1/2),fractional,strided, IstanceNorm, Relu
12	CONV-(N3,K7,S1),InstanceNorm,Relu

GAN has a tendency to show *Mode Collapse*[Appendix .7] which makes training so hard. To avoid *Mode Collapse* in CycleGAN[14], last 50 generated images are kept in a pool to train the discriminator instead of just training with the last generated image. This technique was acquired from *Shrivastava et al.*[24].

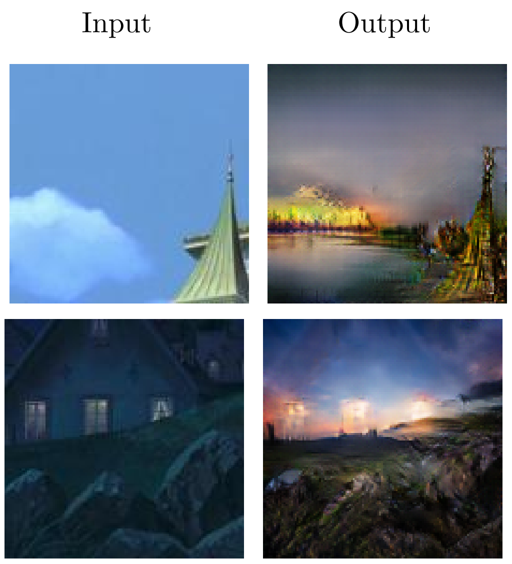

FIGURE 2.3: Comparison

2.3.1.4 Limitations

Although the output of generating cartoon images into real images look realistic, we can see from figure 2.3 that, some of the outputs fail to preserve contents. From figure **??**, CycleGAN[14] outputs a firing sea whereas input image contains only sky. This has been the case for various outputs which is the biggest limitation of CycleGAN[14] for our project.

2.3.2 UNIT

Researchers of UNIT[18] worked on an assumption known as shared latent space assumption, where a pair of corresponding images in different domains can be mapped to a same latent representation in a shared-latent space. The framework combined a variational autoencoder(VAE) and generative adversarial network (GAN). The adversarial training objective enforces the generator to transform corresponding images in two domains, while the variational autoencoders(VAE) make correlations between translated images with input images in the respective domain.

2.3.2.1 Network Architecture & Training

The following abbreviation for ease of presentation is used: N=Neurons, K=Kernel size, S=Stride size. The transposed convolutional layer is denoted as DCONV. The residual basic block is denoted as RESBLK.

Table 2.3: Network Architecture of Encoders and Generators MNIST → USPS

Layers	Encoders
1	CONV-(N64,K5,S2), BatchNorm, LeakyReLU
2	CONV-(N128,K5,S2), BatchNorm, LeakyReLU
3	CONV-(N256,K8,S1), BatchNorm, LeakyReLU
4	CONV-(N512,K1,S1), BatchNorm, LeakyReLU
5	CONV-(N1024,K1,S1)
Layers	Generators
1	DCONV-(N512,K4,S2), BatchNorm, LeakyReLU
2	DCONV-(N256,K4,S2), BatchNorm, LeakyReLU
3	DCONV-(N128,K4,S2), BatchNorm, LeakyReLU
4	DCONV-(N64,K4,S2), BatchNorm, LeakyReLU
5	DCONV-(N3,K1,S1), TanH

Table 2.4: Network Architecture of Discriminators MNIST → USPS

Layers	Discriminators
1	CONV-(N20,K5,S1), MaxPooling-(K2,S2)
2	CONV-(N50,K5,S1), MaxPooling-(K2,S2)
3	FC-(N500), ReLU, Dropout
4a	FC-(N1), Sigmoid
4b	FC-(N10), Softmax

Table 2.5: Network Architecture of Discriminators SVHN → MNIST

Layers	Discriminators
1	CONV-(N64,K5,S1), MaxPooling-(K2,S2)
2	CONV-(N128,K5,S1), MaxPooling-(K2,S2)
3	CONV-(N256,K5,S1), MaxPooling-(K2,S2)
4	CONV-(N512,K5,S1), MaxPooling-(K2,S2)
5a	FC-(N1), Sigmoid
5b	FC-(N10), Softmax

2.3.2.2 Limitations

In UNIT[18] outputs, the images tend to be soft on the surfaces of the outputs. In real world images, the surfaces are not that smooth as it is in UNIT outputs. Real world images tends to be less sharpen on the edges and more crisp on the surface.

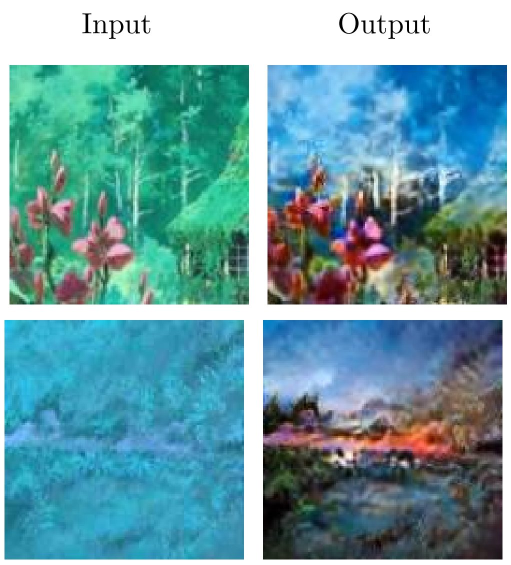

FIGURE 2.4: Comparison

2.3.3 SingleGAN

We have already known about two models which consist of multiple generators and discriminators. However, using multiple generators might be ineffective and inefficient. To reduce inefficiency, researchers of SingleGAN [20] worked on producing unpaired image translation using a **Single Generator**. Basically, it is a multi-domain image-to-image translation using *Single Generator* and multiple *Generative Adversarial* Learning Objects.

Intuitively, while translating an image of one domain to another, there exists some common features, such as content, edge of the first domain which will retain its position even after translation. From figure 2.5, we get an idea how this works where 3 of domain 1 and domain 2 share common features which at this point is clearly the edge.
Researchers of singleGAN use this intuition for their work. They generalize the model by sharing these features. For specific domain mapping, an domain distribution, retrieved by auxiliary information, is used .
For cartoon to real translation, if the auxiliary information is Z, C is cartoon domain and R is real domain, we can define the functions as -

$$x_C^{fake} = G(x_R, Z_c)$$

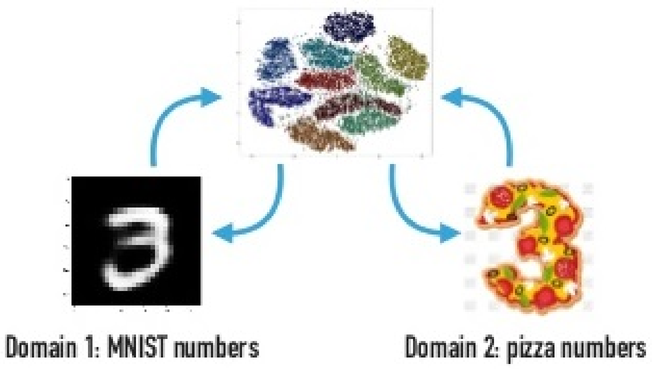

FIGURE 2.5: In this figure, 3 of domain 1(*MNIST Dataset) and pizza-looking* 3 *of domain* 2 *share common features.*

$$x_R^{fake} = G(x_C, Z_R)$$

Here, domain code is retrieved using *Central Biasing Instance Normalization (CBIN)*[25] with the help of auxiliary information.

2.3.3.1 Loss Functions

SingleGAN uses *two* loss functions. These are *Adversarial Loss* and *Cycle Consistency Loss*.

As it gets to use a single generator to translate one image to another, it uses *one Adversarial Loss* for each domain translation. In our case, where we have *two* domains C and R, the loss will be as followiing -

$$L_{adv}(G, D_C) = \mathbb{E}_{x_C}[log(D_C(x_C))] + \mathbb{E}_{x_R, Z_C}[log(1 - D_C(G(x_R, Z_C))]$$

$$L_{adv}(G, D_R) = \mathbb{E}_{x_R}[log(D_R(x_R))] + \mathbb{E}_{x_C, Z_R}[log(1 - D_R(G(x_C, Z_R))]$$

As mentioned in 2.3.1.1 and 2.3.1.2, the same problem may also happen in SingleGAN condition. Due to being *unconstraint*, an additional loss, Cycle Consistency loss is used. For our case, the loss will be -

$$L_{cyc_{sing}} = \mathbb{E}_{x_C}[||x_C - G(G(x_C, Z_R), Z_C)||_1] + \mathbb{E}_{x_R}[||x_R - G(G(x_R, Z_C), Z_R)||_1]$$

So, the final loss is -

$$L = L_{adv}(G, D_C) + L_{adv}(G, D_R) + L_{cyc_{sing}}$$

2.3.3.2 Network Architecture & Training

Generator G uses the ResNet[26] structure with an encoder-decoder framework, which contains two stride-2 convolution layers for downsampling, six residual blocks and two stride-2 transposed convolution layers for upsampling. All the normalization layers are replaced except upsampling layers. For the discriminators D, two discriminators[27] are used to discriminate the real and fake images in different scales.

2.3.3.3 Limitations

Unfortunately, of all models, **SingleGAN** performed the worst. The color scheme of the output is totally faded. We can see examples from 2.6

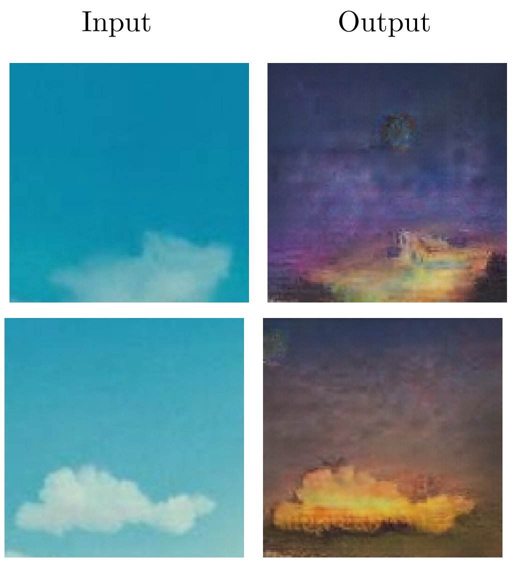

Figure 2.6: Comparison

Chapter 3

Progress & Experiments

So far, we have studied on CycleGAN[14], UNIT[18] and SingleGAN[20] for image-to-image translation tasks. We developed two unpaired datasets to train our network. For the cartoon domain, we've collected almost 3.1K images scrapped from various movies, e.g. *Pokemon*, *My Neigbour Totoro* and *Kikis Delivery*. We used flickr dataset for the real images domain. Images are resized to 128128 resolution. For implementation we used PyTorch and for hardware we used Nvidia GTX 1060. We compare our results through different architectures i.e. CycleGAN[14], UNIT[18] and SingleGAN[20].
In some images, *cycle consistency loss* was unable to preserve content representation of input image. However, UNIT[18] was able to maintain content of input image in the output. Still it has some flaws, as UNIT tends to smoothen the surface of output images, where in real world surfaces are tend to be crisp and less sharpen on the edges. In SingleGAN[20], output images colors become faded and the content is not preserved most of the time.

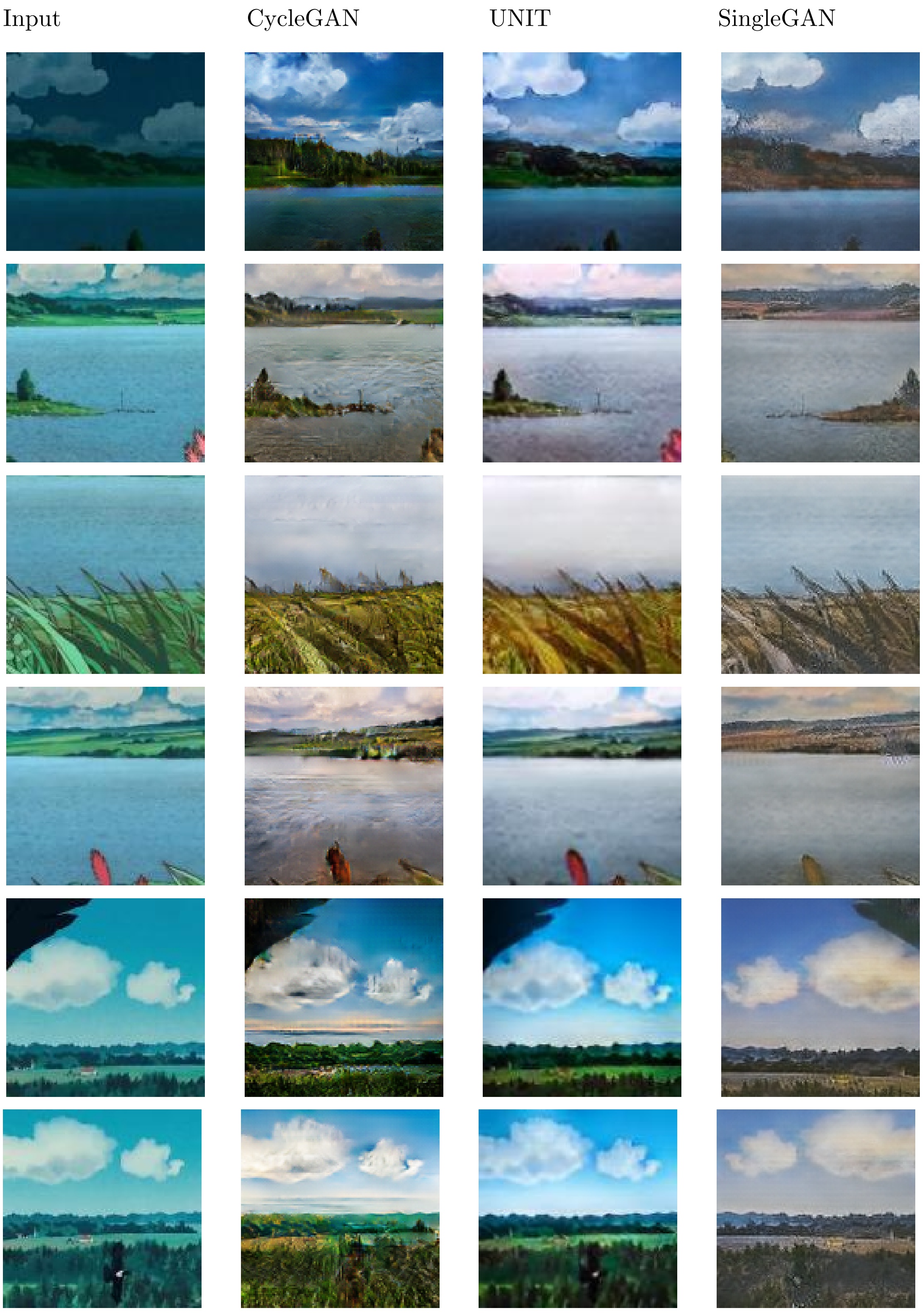

Figure 3.1: Comparison Between CycleGAN, UNIT and SingleGAN

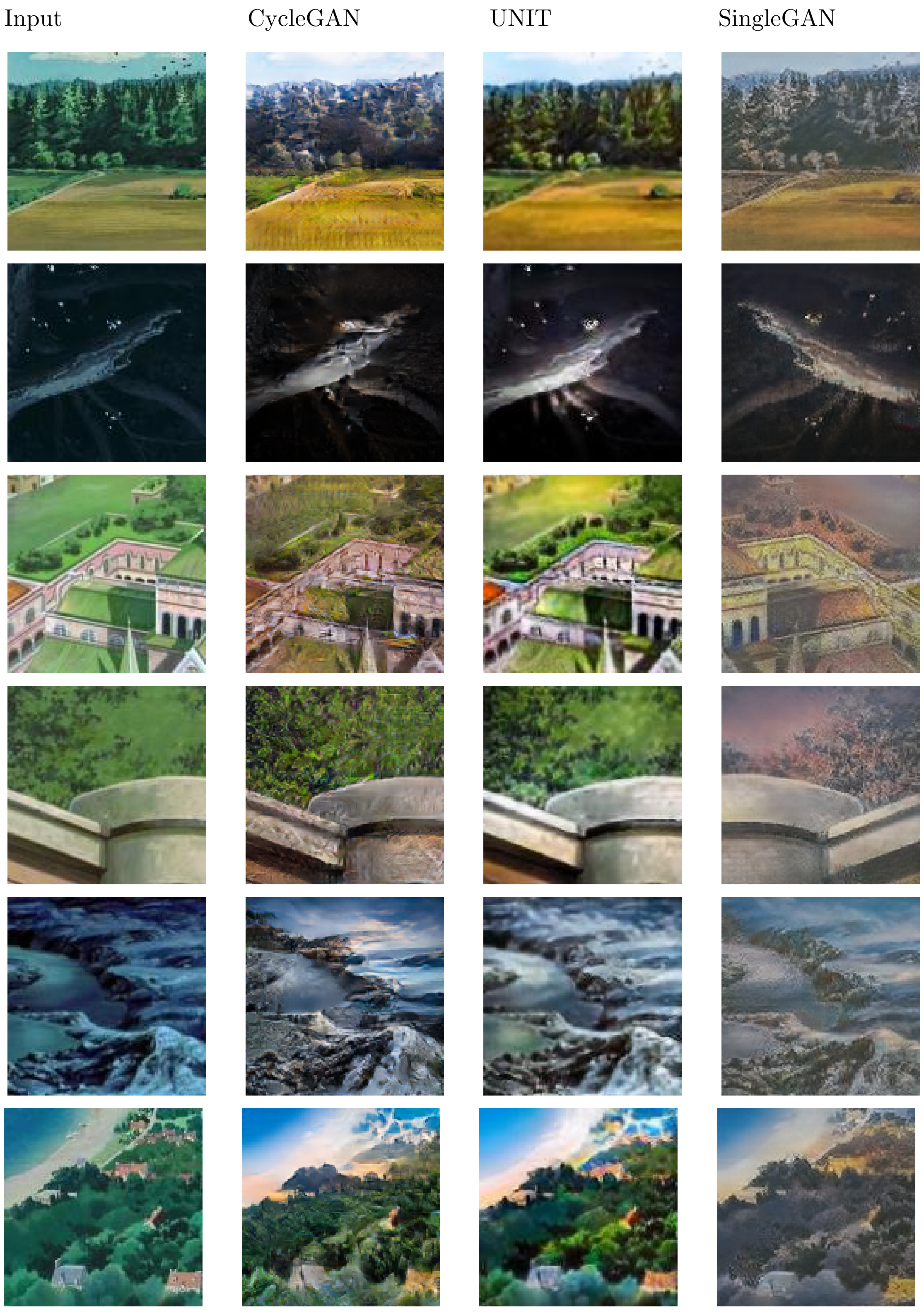

FIGURE 3.2: Comparison Between CycleGAN, UNIT and SingleGAN

Chapter 4

Future Plans & Conclusion

After the experiments and results we achieved from Chapter 3, we noted some observations. Those are described below -

From the output of Figure 3.1, 3.2, it is seen that **CycleGAN** gives pretty realistic output. It seems almost real at the first glance. However, after taking a close look from Figure 2.3, we came to conclusion that, the content of the input image is not quite retrieved in **CycleGAN** outputs. In fact, it translates skies to seas and seas to skies.

Again, for **SingleGAN**, we notice that, this model doesn't produce much satisfactory result. From Figure 2.6 and even from 3.1, 3.2, we see that **SingleGAN** doesn't excel at all. The color is too faded and the contents are not retrieved perfectly.

Lastly, for **UNIT**'s case, we see that the model is succeeded in retrieving every pixel of the input perfectly. Almost no output can be seen where it varies too much from input. However, according to Figure 2.4, as much perfect content-wise as it is, it fails in making outputs more realistic. They seem pretty cartoonish due to the smoothness of the surface.

4.1 Final Decision

From the above observations, we decided to emphasize more on **CycleGAN** and **UNIT** as they complement each other. We are excluding **SingleGAN** as it fails to produce any output close to our satisfaction. Assuming that making the output sharper will improve the result, we planned $four$ options. These are discussed below -

4.1.1 Preprocessing Input

Previously, we used the raw dataset as our input. As our output lacks sharpness, we assume that adding more crisp in the input might give us a better realistic output. On this case, we want to add more details on **UNIT** model and see how that impacts our result.

4.1.2 Using Sharpening Loss

Another idea to add more crisp in the output is to use sharpening loss. We got this idea from **CartoonGAN** [19]. **CartoonGAN** uses an *edge loss* function to create more cartoonish images as they observed that cartoons tend to have sharper edge than real images. We decided to implement idea on **UNIT** model. Instead of using edge loss function, we will use *Sharpening Loss* Function.

4.1.3 PatchGAN as Discriminator

We know from Section 2.3.1.2 that, **CycleGAN** uses *PatchGAN* as discriminator to reduce loss of high frequency details of images such as crisp, texture. We may guess why *CycleGAN* perfroms better in case of high frequency details. So, we decided to add **PatchGAN** discriminator into **UNIT** model as an option to improve our result.

4.2 Conclusion

In this report, we have discussed about our topic in-depth. We showed our experiments based on our project. We made a brief overview of the models of the experiments and their architecture and how they work. Finally, we analyzed all these experiments and developed a plan to improve our project. Our goal is to produce a result which will as real as it can get.

Appendix A

Appendix A

Appendices

.1 Binary Cross Entropy Loss

A criterion to measure the Binary Cross Entropy between the target and the output. The loss can be described as:
$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right],$
where N is the batch size.

.2 Why Least Squares Loss more suitable for GAN

for $\mathcal{L}_{adv}$(Adversarial loss), the negative log likelihood objective is replaced by a least-squares loss. This loss is more stable during training and generates higher quality results.

.3 Encoder - Decoder Network

From [28], Generally, the encoder encodes the input sequence to an internal representation called 'context vector' which is used by the decoder to generate the output sequence. The lengths of input and output sequences can be different, as there is no explicit one on one relation between the input and output sequences.
To sum up, an encoder is a network such as *CNN, RNN, MLP*, which extracts a feature vector from raw inputs and then outputs the feature vecor. And decoder is also a network, which takes that output feature vector as input and produces an output which represents the input in the best way possible.

.4 Low & High Frequency Image

Frequency means the rate of change per pixel. *A high frequency* image is the one which keeps changing a its pixel in a high rate. A striped picture which has black and white stripes will have the highest frequency, since 0 is the lowest pixel and then it changes to 255 which is the highest within a very short time. Again, *a low frequency* image is the one whose pixel values rarely change. For example, a single colored image will have 0 frequency.

.5 Residual Block

After convolutional nueural network used by *AlexNET* won the **ImageNET**, the popularity of **CNN** started to rise. It was assumed that using more layers give better results. However, using more layers produced another problem, *Vanishing Gradient*. It is when after running of few layers, as it goes deeper, the gradient tends to be *zero*. To avoid this problem, after sometime a new model called *ResNET* was created. The main idea of it is to create blocks for which gradient won't be calculated. So, chance of vanishing gradient is decreased. This blocks are called **Residual Blocks**.

.6 Deconvolutional Layers

Deconvolutional layer is a term which is often used to denote a reverse convolution. It is used for image upsampling to get back the original image size. It is done in the following manner - by taking each pixel of the input, multiply them with a 3×3 kernel to get a weighted output, and then it has to be inserted into the output image. When the outputs overlap, it is summed.

.7 Mode Collapse

One of the common problems of GAN is mode collapse problem. In this problem, the generator produces less diversified samples. So there is a possible chance that the generator is generating similar images without any varieties.

Bibliography

[1] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *CoRR*, abs/1603.06668, 2016. URL http://arxiv.org/abs/1603.06668.

[2] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016. URL http://arxiv.org/abs/1603.08511.

[3] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016. URL http://arxiv.org/abs/1609.04802.

[4] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. *CoRR*, abs/1704.03915, 2017. URL http://arxiv.org/abs/1704.03915.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. URL http://arxiv.org/abs/1508.06576.

[6] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. *CoRR*, abs/1601.04589, 2016. URL http://arxiv.org/abs/1601.04589.

[7] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *CoRR*, abs/1705.01088, 2017. URL http://arxiv.org/abs/1705.01088.

[8] 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.

[9] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

[10] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015. URL http://arxiv.org/abs/1506.05751.

[11] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. *CoRR*, abs/1609.03552, 2016. URL http://arxiv.org/abs/1609.03552.

[12] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016. URL http://arxiv.org/abs/1612.03242.

[13] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. URL http://arxiv.org/abs/1604.07379.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251, 2017. doi: 10.1109/ICCV.2017.244. URL https://doi.org/10.1109/ICCV.2017.244.

[15] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8798–8807, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Wang_High-Resolution_Image_Synthesis_CVPR_2018_paper.html.

[16] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *CoRR*, abs/1704.02510, 2017. URL http://arxiv.org/abs/1704.02510.

[17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *CoRR*, abs/1703.05192, 2017. URL http://arxiv.org/abs/1703.05192.

[18] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. URL http://arxiv.org/abs/1703.00848.

[19] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9465–9474, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Chen_CartoonGAN_Generative_Adversarial_CVPR_2018_paper.html.

[20] Xiaoming Yu, Xing Cai, Zhenqiang Ying, Thomas H. Li, and Ge Li. Singlegan: Image-to-image translation by a single-generator network using multiple generative adversarial learning. *CoRR*, abs/1810.04991, 2018. URL http://arxiv.org/abs/1810.04991.

[21] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6836–6845, 2017. doi: 10.1109/CVPR.2017.723. URL https://doi.org/10.1109/CVPR.2017.723.

[22] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *CoRR*, abs/1612.00215, 2016. URL http://arxiv.org/abs/1612.00215.

[23] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. Computational Imaging*, 3(1):47–57, 2017. doi: 10.1109/TCI.2016.2644865. URL https://doi.org/10.1109/TCI.2016.2644865.

[24] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2242–2251, 2017. doi: 10.1109/CVPR.2017.241. URL https://doi.org/10.1109/CVPR.2017.241.

[25] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL http://arxiv.org/abs/1611.07004.

[28] farizrahman4u/seq2seq, Nov 2018. URL https://github.com/farizrahman4u/seq2seq. [Online; accessed 27. Nov. 2018].

[29] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[30] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[31] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2813–2821, 2017. doi: 10.1109/ICCV.2017.304. URL https://doi.org/10.1109/ICCV.2017.304.