Face Style Transfer and Removal with Generative Adversarial Network

by

Qiang Zhu

B.Sc., University of Science and Technology of China, 2017

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

© Qiang Zhu 2020

SIMON FRASER UNIVERSITY
Spring 2020

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
# Approval

<table>
<thead>
<tr>
<th>Name:</th>
<th>Qiang Zhu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree:</td>
<td>Master of Science</td>
</tr>
<tr>
<td>Title:</td>
<td>Face Style Transfer and Removal with Generative Adversarial Network</td>
</tr>
<tr>
<td>Examining Committee:</td>
<td>Chair: Qianping Gu Professor School of Computing Science</td>
</tr>
<tr>
<td></td>
<td>Ze-Nian Li Senior Supervisor Professor School of Computing Science</td>
</tr>
<tr>
<td></td>
<td>Mark S. Drew Supervisor Professor School of Computing Science</td>
</tr>
<tr>
<td></td>
<td>Manolis Savva External Examiner Assistant Professor School of Computing Science</td>
</tr>
<tr>
<td>Date Defended:</td>
<td>May 4, 2020</td>
</tr>
</tbody>
</table>
Abstract

Style transfer plays a vital role in image manipulation and creates new artistic works in different artistic styles from existing photographs. While style transfer has been widely studied, recovering photo-realistic images from corresponding artistic works has not been fully investigated. And all previous work considers style transfer and removal as separate problems.

In this thesis, we present a method to transfer the style of a stylized face to a different face without style and recover photo-realistic face from the same stylized face image simultaneously. Here, style refers to the local patterns or textures of the stylized images. Style transfer gives a new way for artistic creation while style removal can be beneficial for face verification, photo-realistic content editing or facial analysis. Our approach contains two components: the Style Transfer Network (STN) and the Style Removal Network (SRN). STN renders the style of the stylized image to the non-stylized image, and the SRN is designed to remove the style of a stylized photo. By applying the two networks successively to an original input photo, the output should match the input photo. The experiment results in a variety of portraits and styles demonstrate our approach’s effectiveness.

**Keywords:** Style Transfer, Style Removal, Generative Model, Neural Network, Deep Learning
Acknowledgements

First of all, I’d like to express my deepest gratitude and respect to my supervisor Dr. Ze-Nian Li for his unreserved support, encouragement, and patience throughout my graduate study. His guidance is essential to complete this work and also very instructive for my learning. Without his guidance, this thesis cannot be completed. I believe that his positive attitude and enthusiasm for the career will inspire mine in the future as well.

Then I want to extend my gratitude to Dr. Mark S. Drew, the member of my supervisory committee, for his insightful comments and helpful suggestions. Also, I’d like to appreciate Dr. Manolis Savva to be the external examiner and Dr. Qianping Gu to be the chair of my thesis defense.

Last but not least, I appreciate my family and friends for their love, encouragement, and support.
# Table of Contents

 Approval ii 

 Abstract iii 

 Acknowledgements iv 

 Table of Contents v 

 List of Figures vii 

 1 Introduction 1 
  1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 
  1.2 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 
  1.3 Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 

 2 Previous Work 4 
  2.1 Generative model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4 
  2.1.1 Generative model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4 
  2.1.2 Discriminative models . . . . . . . . . . . . . . . . . . . . . . . . . . . 5 
  2.1.3 Generative Adversarial Networks . . . . . . . . . . . . . . . . . . . . . 5 
  2.1.4 Conditional GAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6 
  2.1.5 PatchGAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7 
  2.1.6 CycleGAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8 
  2.2 Different loss functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9 
  2.2.1 Mean Absolute Error (MAE) / $L_1$ Loss . . . . . . . . . . . . . . . . 9 
  2.2.2 Mean Squared Error (MSE) and L2 loss . . . . . . . . . . . . . . . . . 10 
  2.2.3 Perceptual / Feature loss . . . . . . . . . . . . . . . . . . . . . . . . . 10 
  2.2.4 Gram Matrix / Style loss . . . . . . . . . . . . . . . . . . . . . . . . . 11 
  2.3 Style removal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12 
  2.4 Style transfer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13 
  2.4.1 Style Transfer Introduction . . . . . . . . . . . . . . . . . . . . . . . . 13 
  2.4.2 Slow Neural Style Transfer . . . . . . . . . . . . . . . . . . . . . . . . 13 
  2.4.3 Per-Style-Per-Model Fast Neural Method . . . . . . . . . . . . . . . . 14
3 Face Style Transfer and Removal with Generative Adversarial Network

3.1 Outline ................................................................. 17
3.2 Networks ............................................................... 18
3.2.1 Generator Network ............................................. 18
3.2.2 Discriminative Network ................................. 18
3.3 Identity preservation ............................................ 19
3.4 Formulation .......................................................... 19

4 Experiments ............................................................. 22
4.1 Datasets ............................................................... 22
4.2 Evaluation metric ................................................ 22
4.3 Experimental Results ............................................ 23
4.3.1 Qualitative Evaluation ..................................... 23
4.3.2 Quantitative Evaluation .................................... 25
4.3.3 Limitations ....................................................... 26

5 Conclusion ............................................................... 38

Bibliography ............................................................. 39
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>The architecture of our framework contains two parts: a style transfer network $G$ and a style removal network $F$ which are learned simultaneously. $G$ learns to render the style of $y^a$ to $x$ while $F$ learns to remove the style of $y^a$. Adversarial discriminators $D_Y$ aims to distinguish between the real stylized faces $y^a$ from domain $Y$ and samples generated by $G$, and vice versa for $F$ and $D_X$. And $D_S$ is used to determine if two pairs of faces are stylized with the same style. The results in the first state are used as input to generate images in the second state. Then by comparing the output of the second state with the input, we aim to preserve identity and style consistency.</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Generative Model [2]</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Training GANs: Two-player game [2]</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Conditional Generative Adversarial Network [24]</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>CycleGAN architecture [39]</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>System overview. We train an image transformation network to transform input images into output images. We use a loss network pre-trained for image classification to define perceptual loss functions that measure perceptual differences in content and style between images. The loss network remains fixed during the training process. [17]</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Examples of style transfer.</td>
<td>13</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Samples of the synthesized dataset with [17]. (a) Original real face image. (b)-(h) The stylized faces of (a) from Wave, Udnie, the shipwreck of the minotaur, Scream, Candy, Lady, Water, which have been used for training our network. (i)-(m) The stylized faces of (a) from Cup Head, La Muse, Starry Night, Rain Princess, Mosaic, which have not been used for training.</td>
<td>27</td>
</tr>
</tbody>
</table>
Figure 4.2 Samples of the synthesized dataset with [22]. (a) Original real face image. (b)-(d) The stylized faces of (a) from Mosaic, Starry-night, the shipwreck of the minotaur, which have been used for training our network. (e)-(h) The stylized faces of (a) from Candy, Wave, Rain-princess, and La Muse which have not been used for training.

Figure 4.3 Results of style transfer. We compare with style transfer work. (a) Original images. (b) Stylized images with reference style [22]. (c) Our method. (d) Zhu et al.’s method [39]. (e) Isola et al.’s method [16].

Figure 4.4 Results of style transfer. We compare with style transfer work. (a) Original images. (b) Stylized images with reference style [17]. (c) Our method. (d) Zhu et al.’s method [39]. (e) Isola et al.’s method [16].

Figure 4.5 More results of style transfer. (a) and (d) Original images. (b) and (e) Stylized images with reference style [17]. (c) and (f) Our results.

Figure 4.6 More results of style transfer. (a) and (d) Original images. (b) and (e) Stylized images with reference style [17]. (c) and (f) Our results.

Figure 4.7 Results of style removal. We compare with style removal work. (a) Stylized images with [22]. (b) Ground-truth original images. (c) Zhu et al.’s method [39]. (d) Isola et al.’s method [16]. (e) Shiri et al.’s method [28]. (f) Our method.

Figure 4.8 Results of style removal. We compare with style removal work. (a) Stylized images with [17]. (b) Ground-truth original images. (c) Zhu et al.’s method [39]. (d) Isola et al.’s method [16]. (e) Shiri et al.’s method [28]. (f) Our method.

Figure 4.9 More results of style removal. (a) and (d) Ground-truth original images. (b) and (e) Stylized images with [17]. (c) and (f) Our results.

Figure 4.10 More results of style removal. (a) and (d) Ground-truth original images. (b) and (e) Stylized images with [17]. (c) and (f) Our results.

Figure 4.11 Style Transfer User Study. Performance comparison results between our method and other methods.

Figure 4.12 Style Removal User Study. Performance comparison results between our method and other methods.

Figure 4.13 Limitations of Style Transfer. (a) Original images. (b) Stylized images with reference style [22]. (c) Our results.

Figure 4.14 Limitations of Style Removal. (a) Stylized images with [22]. (b) Ground-truth original images. (c) Our results.

Figure 4.15 Limitations of Style Transfer. (a) Original images. (b) Stylized images with reference style [17]. (c) Our results.

Figure 4.16 Limitations of Style Removal. (a) Stylized images with [17]. (b) Ground-truth original images. (c) Our results.
Chapter 1

Introduction

1.1 Background

Painting is a popular art form that has attracted many people over the years. Since the last century, the artistic theory behind attractive artwork has not only attracted the attention of artists but also attracted the attention of many computer science researchers.

In the field of computer vision, researchers usually treat style transfer as a generalized texture synthesis problem, which is to extract the texture from the source and transfer it to the target \([8, 6, 10, 9]\). However, these methods have a common limitation, which is that they often fail to capture image structures effectively since they only use low-level image features.

Recently, inspired by the success of Convolutional Neural Networks (CNNs), Gatys et al. \([12]\) first explored how to utilize a CNN to apply famous painting styles to natural images. Through plenty of experimental results, they demonstrated that a CNN is able to extract style information of famous artwork as well as content information of arbitrary photograph.

After that, a variety of methods have been proposed for style transfer. These CNN based methods leverage a pre-trained network to extract deep features and match their gram matrix statics to recombine the style of artistic work and the content of a given photo \([7, 5, 15, 17, 21, 31, 32, 36]\).

Though the style transfer problem has been investigated widely, the reversed problem style removal has not been fully explored. Style removal is basically to remove the style of the stylized images and restore the original images, which could be beneficial for face photo-realistic content editing or face verification. And all previous work \([28, 17, 22]\) considers style transfer and removal as separate problems.

1.2 Motivation

Style transfer and style removal can also be treated as a domain adaptation problem which is to learn a mapping between the source image domain and target image domain. Many
researchers have used generative adversarial networks (GAN) [13] for the mappings of two image domains and have achieved appealing results in style transfer [19], image editing [38] and image generation [27]. Isola et al. introduced a "pix2pix" framework that learns a mapping between the input and the output images utilizing a conditional generative adversarial network [14]. To learn the mapping, Zhu et al. proposed CycleGAN which adopted a generative network with a cycle consistency loss to make the distribution of the mapped images cannot be distinguished from that of real images in the target domain.

Inspired by the success of photo-realistic style transfer built on generative adversarial networks (GANs), we adopted an unsupervised learning method based on the CycleGAN architecture of Zhu et al. [39] and Conditional GAN of Isola et al. [16]. We consider a problem that has asymmetric forward and backward functions. This type of forward function takes the source image as well as reference stylized image as input, while the backward function requires only the stylized image as input.

This thesis describes a way to apply the style of an example photo of a person to a photo of a different person and restore photo-realistic face from the identical stylized face photo at the same time. In our work, we employ cycle architecture together with variants of cycle-consistency loss to transfer the style of the reference stylized face to a new face and remove the style of the stylized face at the same time. Identity preservation is required in both tasks. In the style transfer task, reconstructing images from deep features may lead to extreme distortions which can result in identity loss. Generally, stylized faces can have diverse facial expressions and the vital facial details are distorted or lost completely. And there can be artistic effects such as profile edges or texture and color changes. These artifacts lead to a partial loss of identity-related information. As a result, it is challenging to recover identity-consistent photo-realistic face images from stylized faces.

This work has been published in International Conference on Machine Vision Applications (MVA) 2019 [40]. And in this thesis, we further utilize a new way to generate the dataset and also conduct a quantitative comparison with other methods.

1.3 Outline

Our idea is to train two separate networks simultaneously: one that transfers style and another that removes style. Each of them is trained with an adversary network. Consecutively applying both networks should keep the identity of the source photo. The overview of our method is as Figure 1.1.

The thesis is organized as follows:

- Chapter 2 gives an introduction of Generative models, multiple loss functions, as well as several algorithms of style transfer and style removal.
• Chapter 3 explains our method towards designing style transfer and removal network based on generative models as well as the training strategies of the network.

• Chapter 4 presents the experiments of style transfer and style removal, and compare with other methods both qualitatively and quantitatively.

• Chapter 5 gives a summary of our method.

Figure 1.1: The architecture of our framework contains two parts: a style transfer network $G$ and a style removal network $F$ which are learned simultaneously. $G$ learns to render the style of $y^\alpha$ to $x$ while $F$ learns to remove the style of $y^\alpha$. Adversarial discriminators $D_Y$ aims to distinguish between the real stylized faces $y^\alpha$ from domain $Y$ and samples generated by $G$, and vice versa for $F$ and $D_X$. And $D_S$ is used to determine if two pairs of faces are stylized with the same style. The results in the first state are used as input to generate images in the second state. Then by comparing the output of the second state with the input, we aim to preserve identity and style consistency.
Chapter 2

Previous Work

2.1 Generative model

2.1.1 Generative model

Generative models represent a type of statistical model that is capable of generating new data instances. For example, new images of animals that look similar to real animals could be generated by a generative model. Formally, assume we have a set of data samples $X$ and a set of labels $Y$, a generative model will capture a joint probability $p(X, Y)$ that is $p(X|Y)p(Y)$, or it will just capture $p(X)$ if there are no labels. A generative model could tell you the probability of a given sample and it contains the distribution of the data itself. For example, a typical generative model is a model that predicts the next word in a sequence, since it could assign a probability to a sequence of words.

Given training data, our goal is to generate new samples from same distribution. So we have training data generated from some distribution $p_{data}(x)$, and we want to learn a model $p_{model}(x)$ to generate samples from the same distribution. As shown in Figure 2.1 [2], we want to learn a $p_{model}(x)$ that is similar to $p_{data}(x)$.

![Figure 2.1: Generative Model](image)
2.1.2 Discriminative models

Discriminative models aim to distinguish different kinds of data instances. For example, a discriminative model can distinguish dogs from cats. Formally, the probability that they try to capture is the conditional probability \( p(Y|X) \). Instead of telling you whether a given instance is likely, a discriminative model just tells you how likely a label is to apply to the instance.

2.1.3 Generative Adversarial Networks

A generative adversarial network (GAN) [13] is employed to sample for complex, high-dimensional training distribution by sampling from a simple distribution, e.g. random noise and learning a neural network transformation to the training distribution. GANs are generative models that are able to generate new data instances that are similar to training data. For example, GANs could produce images that look like a human face, even if the face does not belong to any real person.

GAN has two parts:

- Generator Network: attempt to deceive the discriminator by generating realistic images.
- Discriminator network: attempt to distinguish real images from fake images.

**Minimax objective function:**

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]  \hspace{1cm} (2.1)

In this function:

- \( D(x) \) represents the discriminator’s probability estimation that real data instance \( x \) is real.
- \( \mathbb{E}_x \) is the expected value of all real data instances.
- \( G(z) \) is the generator’s output when noise \( z \) is given.
- \( D(G(z)) \) represents the discriminator’s probability estimation that a fake instance \( x \) is real.
- \( \mathbb{E}_z \) is the expected value of all random inputs to the generator (actually, the expected value of all generated fake instances \( G(z) \)).
- Discriminator \( (\theta_d) \) aims to maximize the objective so that \( D(x) \) is near 1 (real) and \( D(G(z)) \) is near 0 (fake).
• Generator \((\theta_g)\) aims to minimize the objective so that \(D(G(z))\) is near 1 (discriminator is fooled into treating the generated \(G(z)\) to be real).

• The generator can’t directly affect the \(\log(D(x))\) term in the function. As a result, minimizing the loss is equivalent to minimizing \(\log(1 - D(G(z)))\) for the generator.

Training GANs is a two-player game, as shown in Figure 2.2 [2]. GAN training proceeds in alternating periods:

• The discriminator is trained for one or more epochs. **Gradient ascent** on discriminator is:

\[
\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

(2.2)

• The generator is trained for one or more epochs. **Gradient ascent** on generator is:

\[
\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))
\]

(2.3)

Steps 1 and 2 are repeated to continue training the generator and discriminator networks.

After training, we could use the generator network to generate new images.

![Figure 2.2: Training GANs: Two-player game [2]](https://www.overleaf.com/project/5df1ca899f011e000143c1dd)

2.1.4 Conditional GAN

In an unconditional generative model, there is no way to control the modes of the generated data since the input is random noise. However, by applying additional information to the model, you can guide the data generation process. Such conditions can be based on data of
different modalities or on class labels. Conditional GANs trained on labeled datasets, letting you specify labels for each generated instance. For example, an unconditional MNIST GAN will generate random numbers, whereas a conditional MNIST GAN could generate different shapes of fonts for a specified number.

As illustrated before, GANs learn a mapping function between random noise vector $z$ and target image $y$, $G : z \rightarrow y$. On the contrary, conditional GANs [24] learn a mapping $G : \{y,z\} \rightarrow x$, where $y$ is the input image, $z$ is the random noise vector and $x$ is the target image, as shown in Figure 2.3.

If we apply some additional information $y$ to both the generator and the discriminator, the generation adversarial network can be easily extended to a conditional model. $y$ can be any kind of auxiliary information like class labels or other forms of data. We can achieve the conditioning by feeding $y$ as an additional input layer into both the discriminator and generator.

**Minimax objective function:**

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{y \sim p(y), x \sim p(x)} \log D_{\theta_d}(y, x) + \mathbb{E}_{y \sim p(y), z \sim p(z)} \log(1 - D_{\theta_d}(y, G(y, z))) \right] \quad (2.4)$$

The generator $G$ aims to generate outputs that look similar to “real” images $\{x\}$. The adversarially trained discriminator $D$ aims to not only distinguish between "real" images $\{x\}$ and generated "fake" images $G(y, z)$, but also detect if $y$ and $x$, $y$ and $G(y, z)$ are matched or not.

### 2.1.5 PatchGAN

Isola et al. exploited a PatchGAN discriminator architecture, which only punishes the structure at the scale of patch [16]. After dividing an image into $N \times N$ patches, the discriminative operation is performed for each patch. This discriminator attempts to classify whether each patch in the image is fake or real. They run this discriminator across the image convolutionally and provide the final output of $D$ by averaging all responses of each patch.

It can be found that when $N = 1$, it is equivalent to pixel-by-pixel operation. When the image size is 256 and $N = 256$, the operation is on one entire image. They prove that $N$ that is much smaller than the entire size of the image could still generate high-quality outputs. And the experiment shows that when $N = 70$, the result is the best.

PatchGAN has advantages since the smaller PatchGAN runs faster, has fewer parameters, and can be employed for any large images. In fact, one of the biggest advantages is that you can generate large images by training on small images because you are working on the small patches anyway. For example, your original image is $256 \times 256$, and we work on every $70 \times 70$ blocks when training the model. If we want to convert a $1000 \times 1000$ image, we only need to divide $1000 \times 1000$ image into multiple $70 \times 70$ blocks, then each block can be converted separately [1].
2.1.6 CycleGAN

CycleGAN [39] is actually a $X \rightarrow Y$ one-way GAN plus a $Y \rightarrow X$ one-way GAN. Two GANs share two generators and then each has a discriminator, so in total there are two discriminators and two generators. A one-way GAN has two losses, and a CycleGAN adds up to four losses.

The objective is to learn mappings between two domains $X$ and $Y$. The model contains two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$. And there are two adversarial discriminators $D_X$ and $D_Y$, where $D_X$ is designed to discriminate between images $\{x\}$ and translated images $\{F(y)\}$, $D_Y$ tries to distinguish between $\{y\}$ and $\{G(x)\}$.

**Loss functions:**

**Adversarial loss** is used to match the distribution of generated images to the data distribution in the target domain.

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

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)]$$

$$+ \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))]$$
where $G$ tries to produce 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 $y$. Similarly, the adversarial loss for the mapping $F : Y \rightarrow X$ and its discriminator $D_Y$ is:

$$
L_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log (1 - D_X(F(y)))]
$$

Cycle consistency loss is employed to forestall the learned mappings $G$ and $F$ from contradicting one another.

we believe that the learned mapping functions should be cycle-consistent: as shown in Figure 2.4, for each image $x$ that is from domain $X$, the output of the translation cycle should be close to the original image, i.e. $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. This is called forward cycle consistency. Similarly, for each image $y$ that is from domain $Y$, $G$ and $F$ also have backward cycle consistency: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. We could express this behaviour as cycle consistency loss:

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

Figure 2.4: CycleGAN architecture [39]

### 2.2 Different loss functions

#### 2.2.1 Mean Absolute Error (MAE) / $L1$ Loss

The loss function on per-pixel is used as a measure to understand the differences between images on the level of the pixel. The loss function aims to measure the difference between the pixel values of two images.

$L1$ loss is expressed as below [37]:

$$
L_1(I^1, I^2) = \frac{1}{N} \sum_{i,j} |I^1_{i,j} - I^2_{i,j}|
$$

(2.5)
where $I_{i,j}^1$ is the $i,j$th pixel of the processed image, and $I_{i,j}^2$ the $i,j$th pixel of the ground truth image.

Although this per-pixel loss function may seem complicated at first, it is a very simple concept. In short, the loss function can find the sum of all absolute differences between each pixel. That is to say, each pixel value is actually measured along with the other values to produce a complete representation of the pixel loss of the image. From a human viewer’s perspective, MAE may generate an image that appears to have a higher quality for the image enhancement task.

### 2.2.2 Mean Squared Error (MSE) and L2 loss

If $I_{i,j}^1$ is the $i,j$th pixel of the processed image, and $I_{i,j}^2$ the $i,j$th pixel of the ground truth image, the MSE loss \cite{3} is defined by:

$$MSE(I^1, I^2) = \frac{1}{N} \sum_{i,j} (I_{i,j}^1 - I_{i,j}^2)^2$$

(2.6)

The comparison is performed on each of the three channels of the RGB image. MSE is used to compare the distance between the target image’s pixels and the generated image’s pixels. MSE takes the average of each pixel’s difference and squares it.

And $L2$ loss is just the square root of MSE loss:

$$L_2(I^1, I^2) = \sqrt{\frac{1}{N} \sum_{i,j} (I_{i,j}^1 - I_{i,j}^2)^2}$$

(2.7)

The goal of $L2$ loss is to use the least squared deviation to minimize the total of the squared differences between the target image and the generated image. With $L2$ loss, it is meaningful to discuss the distance between two images.

Using the MSE metric is likely to make the network find a blur filter through training because that is the way to have the easiest solution and the lowest loss to converge and minimize the loss. Although the loss is minimized, the perceived quality of the generated image is poor from the perspective of attracting human viewers. For example, the loss of an image that has salt and pepper noise may be lower than that of some generated images which could be closer to the ground truth images from human perception.

### 2.2.3 Perceptual / Feature loss

In \cite{17}, Johnson et al. proposed a feed-forward neural network architecture to transfer a style to any target images, as shown in Figure 2.5. And they employed feature/perceptual loss to measure high-level semantic and perceptual differences between images, which enforces the output image $\hat{y}$ to be similar to the target image $y_c$ perceptually but does not require them to match exactly. This is intuitive since images that have similar content will have similar
feature representations in the higher layers of a network. They utilize a pre-trained loss network $\phi$ used for image classification, which means that the feature loss functions are also deep CNN. In their experiments $\phi$ is the 16-layer VGG network [29] which is pre-trained on the ImageNet dataset.

They encourage the output image $\hat{y}$ and the target image $y_c$ to have similar feature representations which are calculated by the loss network $\phi$, rather than enforce them to match on the level of pixels.

When we use network $\phi$ to process the image, let $\phi_j(x)$ be the activations of the $j$th layer; $\phi_j(x)$ will be a feature map of shape $C_j \times H_j \times W_j$ if $j$ is a convolutional layer. Then the perceptual loss is the normalized and squared Euclidean distance between the feature maps:

$$l_{\phi,j}^{\text{feat}}(\hat{y}, y_c) = \frac{1}{C_j H_j W_j} \left\| \phi_j(\hat{y}) - \phi_j(y_c) \right\|^2_2 \quad (2.8)$$

Figure 2.5: System overview. We train an image transformation network to transform input images into output images. We use a loss network pre-trained for image classification to define perceptual loss functions that measure perceptual differences in content and style between images. The loss network remains fixed during the training process. [17]

In this way, the perceptual loss function is able to know what features are in the target image and can evaluate how well the features of the output image match these features, instead of just comparing pixel differences. As a result, the model that is trained with this loss is capable of generating much finer details in the output images.

### 2.2.4 Gram Matrix / Style loss

Gatys et al. [11] proposed to make use of gram matrix to represent a style w.r.t specific content. Based on the CNN activations in each layer of the loss network (Figure 2.5), they compute the correlations between the responses of different filters to build a style representation. These feature correlations are defined by the Gram matrix.
A set of vectors’ gram matrix is basically a matrix such that each value of the matrix is the inner product of two vectors. And the gram matrix here is basically the multiplication of the activation matrix and the activation matrix’s transpose. The intuition of adopting a gram matrix is that they want to capture the statistics of features.

They take the activations from each of the channels and flatten them into a one-dimensional vector, and then take the dot products of these vectors with each other to construct a gram matrix. The dot product indicates the degree of correlation between each combination of channels. If one channel indicates bright colors and the other channel indicates texture, then a dot product will indicate cells with bright colors also tend to have texture.

So they define gram matrix $G_l \in \mathbb{R}^{N_l \times N_l}$ as below, where $G_{ij}^l$ is the inner product between the vectorized filter activations $i$ and $j$ in layer $l$:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

(2.9)

The style loss of layer $l$ is given by:

$$E_l = \frac{1}{4N_l^2M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

(2.10)

and the total loss is:

$$L_{\text{style}}(y_s, \hat{y}) = \sum_0^L w_l E_l$$

(2.11)

Here, $A_l^l$ is the gram matrix style representation of the original image $y_s$ and $G_l^l$ is the representation of the generated image $\hat{y}$ in layer $l$. $N_l$ is the number of feature maps and $M_l$ is the size of the flattened feature map in layer $l$. $w_l$ is the weight given to the style loss of layer $l$.

This encourages the trained model to generate output images that have correct context and style, and appear closer to the target and looks more convincing. Gram matrix loss has been widely used to represent style loss between the target and generated images. If two images have similar gram matrices, they will have the same style. Gram matrix loss is useful for style transfer because by flattening the filter maps, we remove the spatial relationship information from the comparison.

### 2.3 Style removal

Different from style transfer, style removal is to digitally remove the style of the stylized image [28]. Fatemeh and Xin [28] proposed a neural network for style removal. Photo-realistic images are restored from stylized ones by using De-stylization Neural Network. The style
removal network of them consisting of convolutional layers, fully connected layers, and de-convolutional layers. To extract structures from stylized images, the convolutional layer is designed. Consecutively, the extracted feature maps are transferred into the corresponding original images’ feature maps by the fully-connected layer. Finally, real images are generated from the transferred feature map by the deconvolutional layers. Their discriminator consisting of fully connected layers and convolutional layers is used to encourage the restored images to be indistinguishable with real images. Their network learns a mapping from stylized feature maps to realistic feature maps. In our work, we simultaneously perform both tasks and we demonstrate that better results can be gained by improving the processes of transfer and removal in turn.

2.4 Style transfer

2.4.1 Style Transfer Introduction

Given a content image and a style image, we aim to render the style of the given style image to the given content image to produce a new image, as shown in Figure 2.6. Here, content is the semantic information of an image and style is local patterns or textures of an image. This is called style transfer [11, 12].

The traditional style transfer methods render the style of the style image to the content image to produce a new stylized image. Most recent style transfer approaches utilize Convolutional Neural Networks (CNNs), but use different loss function and diverse optimization method [12, 5, 17, 18, 31]. Gatys et al. [12] first proposed to employ a CNN for style transfer which is to use optimization to match the gram matrix correlation statistics. Johnson et al. [17] speed up the process by training a feed-forward network using perceptual loss functions.

2.4.2 Slow Neural Style Transfer

The slow neural method [11, 12, 20, 26, 18] depends on online image optimization.

The idea contains the following steps. Given the content and style images, the first step is to extract the corresponding content and style information. And then recombine the style

Figure 2.6: Examples of style transfer.
and content information to get a stylized representation. The last step is to reconstruct a stylized output based on the stylized representation iteratively.

Gatys et al. [12] first proposed a neural style transfer algorithm. They find that a deep CNN is able to extract the content information of an image by reconstructing representations from intermediate layers of VGG-19 network [29]. So they penalize the difference between the feature representations of content and stylized images to reconstruct the content part of the newly stylized image and build the style part by matching the gram matrix correlation statistics.

The cost function has two terms: a style loss term and a perceptual content loss term. So the total loss function is a weighted combination of content and style losses, basically, it represents the problem that the content of the stylized image is required to be similar to the content of the content image and the style of the stylized image should be similar to the style of the style image.

Then they minimize this loss by iteratively optimizing the input image. They initialize the input with a blank gray canvas and start changing the pixel values to minimize the loss. And Adam optimizer is used to minimize this loss.

Though their method can produce appealing results, it takes such a long time to generate a stylized image. Since the CNN function inevitably loses some low-level information, it does not perform well in maintaining the consistency of fine structures and details. In addition, due to the limitations of Gram-based style representations, it fails for photo-realistic synthesis.

2.4.3 Per-Style-Per-Model Fast Neural Method

Johnson et al. [17] and Ulyanov et al. [32] first proposed an off-line neural method based on model optimization. The idea of these two methods is similar. They first train a feed-forward style transfer network given a specific style and then in the test phase, the trained network could generate a stylized output in a single forward pass. The difference lies in the architecture of their network. Johnson et al.’s design is based on the network proposed by Radford et al. [25] and they also employ fractionally-stride convolutions as well as residual blocks. And Ulyanov et al. design their generator network using a multi-scale architecture.

The objective function of their methods is similar to that of Gatys et al. [11], which shows that their methods are also parametric methods based on summary statistics. The algorithms of Ulyanov et al. and Johnson et al. are able to achieve style transfer in a real-time manner.

Shortly after [32, 17], Ulyanov et al. [34] proposed instance normalization (IN) to improve the stylization quality of the outputs. IN is basically to employ normalization to every single image instead of a batch of images, which is equivalent to batch normalization (BN) if we set the batch size to be 1. Compared with BN, using IN in style transfer network achieves better visual results and displays a faster convergence. Since IN is capable of
normalizing the content image’s style to the desired style [15], we could learn the objective much easier as the rest of the network only needs to handle the content loss.

2.4.4 Multiple-Style-Per-Model Fast Neural Method

Compared with slow neural style transfer methods, the PSPM method’s speed of generating stylized images is two orders of magnitude faster. However, for each specific style, a separate feed-forward generator needs to be trained, which is not flexible and time-consuming. However, the paint strokes of many paintings are similar and just have different color palettes. So it is redundant to train a separate feed-forward network for each style. Therefore, MSPM [4, 7, 36] is proposed. By incorporating different styles into one single model, MSPM has more flexibility compared with PSPM.

Chen et al. [4] proposed to bind each style with a small number of parameters. By learning the style and content information utilizing separate network components, they successfully decouple style and content explicitly. More specifically, they individually learn different styles by using "StyleBank" layers, which are mid-level convolution filters. They bind each style to a set of parameters of the 'StyleBank' layer. Each style is bound to a small number of parameters in the 'StyleBank' layer. Content information is learned by the rest of the network and it is shared by different styles. By fixing the content parts of the network, and training only a “StyleBank” layer for a new come style, their method also supports incremental training flexibly. The advantage of their methods is that it makes easier to learn a new style. Nevertheless, the limitations of neural style transfer algorithms still exist. For example, brush strokes are lack of semantics, details, depth, and variations.

2.4.5 Arbitrary-Style-Per-Model Fast Neural Method

The goal of Arbitrary-Style-Per-Model Fast Neural Method [5, 22, 15] is one-model-for-all. That is to say, transferring arbitrary artistic styles using one single trained model.

By using several feature transformations, Li et al. [22] tries to transfer any artistic style in a learning-free manner. Similar to [15], their encoder is composed of the first few layers of the pre-trained VGG network and train the corresponding decoder. But they used whitening and coloring transformations (WCT): \( I = Dec(WCT(F(I_c), F(I_s))) \) to take place of the AdaIN layer [15] that is in the middle of the encoder and decoder. They noticed that whitening transformation is able to preserve the content structure and remove style-related information. As a result, when encoder feeds content activations \( F(I_c) \) into whitening transformation, it is capable of removing the original style from the input content image and a filtered representation containing only content information will be returned. Then they combine the filtered content representation into the style patterns included in \( F(I_s) \) by applying a coloring transformation. And by decoding the transformed features, we could obtain the styled result \( I \). Their algorithm does not have generalization capabilities limitations as they use a learning-free manner to transfer artistic styles. However, their
algorithm is not very effective at generating fine strokes and sharp details. It still doesn’t take preserving variations and depth information in brush strokes into consideration.
Chapter 3

Face Style Transfer and Removal with Generative Adversarial Network

3.1 Outline

Style transfer and removal can also be posed as a domain adaptation problem [39, 16, 28]. Inspired by these methods which can transfer an image from a source image domain to have appearance similarity with images in a target domain, we introduce a way to transfer style to a face photo, where the style is from an example face of another person. At the same time, the style of the stylized face could also be removed. Two networks (Figure 1.1) are used: one to transfer style and another one to remove the style. The style transfer network takes a source face image and a stylized face image as input, while the style removal network only requires the stylized face image as input. Both transform networks should preserve the identity of the source face image. The style removal network is utilized to help maintain the identity in the style transfer process. And we also employ the identity-preserving and the pixel-level Euclidean loss functions to constrain the recovered faces to lie on the manifold of faces without style while preserving its identity. Finally, an adversarial loss is leveraged to ensure that we can obtain satisfying visual results.

To sum up, our main contributions are:

- A transform network that can transfer the style of a stylized face photo to a source face photo and recover photo-realistic face from the same stylized face image.
- We unite an adversarial loss, a cycle-consistent loss, an identity-preserving loss, and a pixel-level similarity loss to transfer style and recover face.
3.2 Networks

We aim to apply the style of a person’s face to a different person and infer an identity-preserving and photo-realistic face from a stylized face at the same time. In order to achieve this purpose, a framework which contains a Style Transfer Network and Style Removal Network is used, both of them made up of a generator network and discriminative network. The style transfer network is encouraged to extract the feature of the stylized face and apply it to a different person’s face, and the style removal network is trained to recover faces that origin from the latent space of original faces. And the discriminative network is trained to distinguish between generated faces and original faces.

3.2.1 Generator Network

For generators $G$ and $F$, a reasonable architecture choice is a traditional encoder-decoder network, which will gradually down-samples and encode the input image into a compact hidden code, and then gradually up-samples the hidden code to restore an image which has the same resolution compared with the input image.

Style transfer network $G$ learns to extract the style of $y^\alpha$ and renders it to $x$ while preserving the identity of $x$. While the style removal network $F$ learns to remove the style of the same photo $y^\alpha$ maintaining its identity. The autoencoder of $F$ learns a deterministic mapping for converting Images from portraits’ space to some latent space through the encoder and learns a mapping to transform images from latent space to real faces’ space through the decoder. In this way, the encoder is able to extract the feature representations of the stylized faces and converts them into feature maps in the latent real face domain, and then the decoder reconstructs photo-realistic faces from these feature representations.

We adopt Johnson et al.’s [17] architecture for our generative networks, which have shown remarkable results for super-resolution and neural style transfer. Their network includes two stride-2 convolution blocks, a few residual blocks [14], and two stride-$\frac{1}{2}$ fractionally-stride convolutions blocks. We utilize 6 blocks for $128 \times 128$ training images and we also use instance normalization [33] similar to Johnson et al. [17].

3.2.2 Discriminative Network

In order to obtain attractive visual effects and capture global structures, we adopt a discriminator that enforces the generated faces and the original faces to lie in the identical latent space. Our discriminators follow the architecture design of the discriminator in pix2pix [16]. The discriminator will distinguish the entire image instead of looking at image patches since faces contain unique global structures. The discriminative loss (also called adversarial loss) encourages the distribution of the generated faces to be similar to that of the original faces. This loss is also used to update the parameters of the generator (parameters of the discriminator and generator are updated alternately).
3.3 Identity preservation

So as to restrict the restored faces to share more similarity of features with the ground-truth faces. An identity-preserving loss and pixel-wise $L^2$ loss are employed to enforce the restored faces and their ground truth original face counterparts to have appearance similarity. The pixel-wise $L^2$ loss encourages intensity-based similarity between the restored faces and their ground truth faces.

Motivated by the ideas of Johnson et al. [17] and Gatys et al. [11], an identity-preserving loss is also utilized. Specifically, we measure the Euclidean distance between the feature maps of the restored and the ground truth counterparts. The feature maps are obtained from the ReLU layer activations of the VGG-19 network [29]. The VGG network could capture useful facial features because it is pre-trained on the ImageNet dataset which is a very large image dataset.

As a result, by encouraging similarities of feature representations between generated and real faces, we are capable of preserving identity information. Finally, we consider pixel-level $L^2$ loss, identity-preserving loss and adversarial loss as our ultimate loss functions for training our network.

3.4 Formulation

Let $X$ and $Y$ be the no-style and with-style image domains and training samples $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_j\}_{j=1}^M$ where $y_j \in Y$ are given. $Y^\alpha \in Y$ stands for a sub-domain of $Y$ that consists of images of a particular style $\alpha$. While $y_j \in Y^\alpha$, it is denoted as $y^\alpha_j$. And the data distribution of $X$ and $Y$ are denoted as $x \sim p(x)$ and $y \sim p(y)$.

As illustrated in Figure 1.1, the framework consists of two networks: $G : X \times Y^\alpha \rightarrow Y^\alpha$ and $F : Y \rightarrow X$. The two networks $G$ and $F$ are trained at the same time, $G$ is designed to render a style and $F$ is used to remove the style. We feed network $G$ with an image of a face with style, $y^\alpha \in Y^\alpha$, and a picture of a different face without style, $x \in X$. Style transfer network $G$ learns to extract the style of $y^\alpha$ and renders it to $x$ while preserving the identity of $x$. While the style removal network $F$ learns to remove the style of the same photo $y^\alpha$ maintaining its identity. Note that if $G$ and $F$ work successfully, the style of the output of $G$ could be transferred to the output of $F$ which will double the number of training samples. And if $G$ and $F$ can maintain identity, we can attain two images that look like the two input images. Thus, based on the above analysis, we have the following losses.

**Adversarial loss.** An adversarial loss is utilized to encourage the results of $G$ to be indistinguishable from the real stylized samples from domain $Y$, the loss is defined as:

$$L_G(G, D_Y) = E_{y \sim p(y)}[\log D_Y(y)]$$

$$+ E_{x \sim p(x), y \sim p(y)}[\log(1 - D_Y(G(x, y)))]$$
Where G is encouraged to generate faces $G(x, y^\alpha)$ indistinguishable from the real samples while $D_Y$ aims to distinguish between the reference stylized faces $y^\alpha$ from domain $Y$ and the translated faces $G(x, y^\alpha)$. A similar adversarial loss is also introduced to force $F$ to generate images that look similar to the non-stylized reference faces from domain $X$:

$$L_F(F, D_X) = E_{x \sim p(x)}[\log D_X(x)] + E_{y^\alpha \sim p(y)}[\log(1 - D_X(F(y^\alpha)))]$$

**A variant of Cycle Consistent loss.** The learned mapping functions should have cycle-consistency property. For every image $y^\alpha$ in domain $Y$ and $x$ in domain $X$, our image generation network should be capable of bringing $x$ and $y^\alpha$ back to the input image. This is to say, if we transfer style to $x$ and then remove the style immediately, we could obtain the image $x$ exactly. And if we stylize face $x$ with the style of face $y^\alpha$, and then transfer the same style of the result $G(x, y^\alpha)$ back to the style-removed face $F(y^\alpha)$, the result $G(F(y), G(x, y^\alpha))$ should look similar to the input face $y^\alpha$. These constraints help preserve the identity of $x$ as well as ensure the successful transfer of style $y^\alpha$. So the cycle consistent loss function is defined as:

$$L_{cyc}(G, F) = E_{x \sim p(x), y^\alpha \sim p(y)}[\|F(G(x, y^\alpha)) - x\|_1] + E_{x \sim p(x), y^\alpha \sim p(y)}[\|G(F(y^\alpha), G(x, y^\alpha)) - y^\alpha\|_1]$$

**Style loss.** Inspired by [16], an assisting discriminator $D_S$ is also employed to determine if two pairs of faces are stylized with the same style. When training the model, we need to feed $D_S$ with two style pairs, one is fake style pairs ($y^\alpha, G(x, y^\alpha)$) and another one is real style pairs ($y^\alpha, the same style \alpha rendered to another face$).

$$L_S(G, D_S) = E_{y^\alpha \sim p(y)}[\log D_S(y^\alpha, x_r)] + E_{x \sim p(x), y^\alpha \sim p(y)}[\log(1 - D_S(y^\alpha, G(x, y^\alpha)))]$$

Where $x_r$ is a synthetic ground-truth generated by the current style transfer algorithm [22, 17]. Among examples of different faces stylized with the same style, it can provide the discriminator $D_S$ a clue about what should be classified as positive examples.

**MSE loss for $F$.** The style-removed face $F(y^\alpha)$ is enforced to be indistinguishable from its ground-truth $y^\alpha_r$. The pixel-wise MSE loss function between $F(y^\alpha)$ and $y^\alpha_r$ is as follows:

$$L_{MSE}(F) = E_{y^\alpha \sim p(y)}[\|F(y^\alpha) - y^\alpha_r\|^2]$$

**Identity loss for $F$.** To maintain the identity of the style-removed faces, we encourage the style-removed face $F(y^\alpha)$ and the ground-truth face $y^\alpha_r$ to have similar feature representations which are computed by VGG-19 pre-trained network [29]. The identity-preserving loss $L_{id}$ is expressed as:
\[ L_{id}(F) = \mathbb{E}_{y^o \sim p(y)}[\|\phi(F(y^o)) - \phi(y^o_r)\|^2] \]

Here \( \phi(.) \) denotes the activations of the layer ReLU3-3 of the VGG-19 [29] pre-trained network while processing some input image.

**Total loss.** The loss \( L \) is defined as:

\[ L = \lambda_G L_G + \lambda_F L_F + L_{cyc} + \lambda_S L_S + L_{MSE} + L_{id} \]

\( \lambda_G, \lambda_F, \lambda_S \) are the weights to balance different losses.

**Training Details.** CycleGAN [39] is used to pre-train \( F \). CycleGAN could remove most of the style of a face. We alternately train \( G \) and fine-tune \( F \) after \( F \) is initialized by CycleGAN. \( G \) is trained ten times more frequently than \( F \) because \( G \) is much harder to train. For the first 100 epochs, we set \( \lambda_G = \lambda_F = \lambda_S = 0.1 \). For the second 100 epochs, we trust the discriminators more and set \( \lambda_G = \lambda_F = \lambda_S = 0.5 \).
Chapter 4

Experiments

4.1 Datasets

To train our network, we need four separate datasets, two containing faces without style and another two containing faces with a wide variety of styles. We utilize the CelebA [23] dataset to generate such datasets. Firstly, we randomly select 1.4K source faces from the dataset and then resize them to get $128 \times 128 \times 3$ RGB images. These images are used as real ground-truth faces $y_{\alpha}$. To generate stylized faces, we retrain the fast style transfer [17] network for seven diverse styles. And finally we harvest 1.4K training pairs for $\{y_{\alpha}, y_{\alpha}^c\}$ pairs. Then using other 1.4K real faces and the same seven styles, we obtain 1.4K training pairs for $\{x_r, x\}$ pairs in the same way.

To test our network, we utilize 1K real faces to generate 6K testing reference stylized images for $\{y_{\alpha}\}$ from six various styles (La Muse, Starry Night, Mosaic, Rain Princess, Cup Head, Candy) and choose other 6K real faces for $\{x\}$. As shown in Figure 4.1, we applied different styles to a single source face producing vivid stylized results. Note that it is more challenging to restore realistic faces from the Cup Head, Mosaic, and Candy styles because the facial details are distorted and excessively smoothed during the styling process, as shown in 4.1. We also employ the approach of Li et al. [22] to produce stylized faces, as shown in Figure 4.2. There is no overlap between the training and testing datasets.

4.2 Evaluation metric

Except for qualitative evaluation, a quantitative evaluation is also conducted. And we report the Structural Similarity Index (SSIM) [30], which is a perceptual measure that quantifies the degradation of image quality due to processing such as data transmission losses or data compression. SSIM is based on the image’s visible structures.

SSIM is a complete reference measure that compares two images, one is the original image and another is the processed image which is usually compressed. SSIM is a well-known metric in the video field, but it could also be used for photograph applications.
For two images $x$ and $y$, the SSIM metric compares their corresponding pixels and the neighborhoods using the following comparison metrics: luminance ($I(x,y)$), contrast ($C(x,y)$), and structure ($S(x,y)$):

$$I(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$C(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$S(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

(4.1) (4.2) (4.3)

Where $\mu_x$, $\mu_y$, $\sigma_x$, and $\sigma_y$ are the mean and standard deviations of pixel intensity of an image patch that is centered at either $x$ or $y$. Following the idea of [35], they choose a padding size of 5 pixels and obtain $11 \times 11$ image patches. For image patches that are centered at $x$ and $y$, the sample correlation coefficient between corresponding pixels is defined by $\sigma_{xy}$. For numerical stability, they also add constants $C_1$, $C_2$, and $C_3$ to the comparison functions. Then the SSIM score is given by the combination of these comparison functions:

$$SSIM(x,y) = I(x,y)^\alpha C(x,y)^\beta S(x,y)^\gamma$$

(4.4)

### 4.3 Experimental Results

Below we compare our methods with other approaches both quantitatively and qualitatively. So as to perform comparison fairly, we use our training dataset to train the approaches [39, 16, 28] for style transfer and removal tasks. And we used both Johnson et al.'s [17] algorithm and Li et al.'s [22] algorithm to generate the training dataset.

#### 4.3.1 Qualitative Evaluation

**Style Transfer Comparison.** Figure 4.3 and Figure 4.4 shows the style transfer results. Our network is able to transfer a wide variety of artistic styles across diverse source faces preserving the identity of them. We also compare our method with two different previous work [39, 16]. CycleGAN [39] is an unsupervised approach that uses unpaired dataset for the image-to-image translation task. It utilizes generative networks to make the mapped images and the real samples in the target domain have the same data distribution. To use it for style transfer, we employed a collection of stylized faces with the same style and a collection of source faces to train the network. The learned mapping function takes one source face as input and transforms it into the specific stylized face domain. CycleGAN employs a patch-based discriminator. Because the patch-based discriminative network aims to distinguish whether an image patch is from generated faces or the original stylized faces,
it fails to take global structures of styles into account. As shown in Figure 4.3(d), the generated stylized faces are blurry and the styles are not smoothly applied to the whole face. As shown in Figure 4.4(d), the mouth, nose, and eyes of the stylized faces are distorted and preservation of facial identity is lost. We also compare our work with [16], a conditional generative adversarial network, called pix2pix. For style transfer task, we use a paired stylized faces and source faces to train the network. Since pix2pix also utilizes a patch-based discriminator, it doesn’t consider global structures of styles. From the fifth column of Figure 4.3 and 4.4, one can see that the stylized faces are fuzzy and distorted. Thus, their network fails to generate attractive results. Compared with previous work, we obtain more appealing results and perform better in preserving the facial identity of the source face photo, as shown in Figure 4.3(c) and 4.4(c). We also show more style transfer results in Figure 4.5 and Figure 4.6.

**Style Removal Comparison.** In Figure 4.7 and Figure 4.8, we compare our results with three different methods by using our training dataset to train the three approaches. We train CycleGAN [39] using a set of stylized faces and a set of source faces. The network learns a mapping between two different domains. The learned mapping function takes one stylized face as input and transforms it into the real face domain. The network fails to capture the global structure of faces because it utilizes a PatchGAN discriminator. As shown in Figure 4.7(c) and 4.8(c), the recovered faces are distorted and the style that overlaps with the hairs was not fully removed. We compare our work against [16]. Since it also employs a patch-based convolutional neural network to discriminate the image patch between a source face and a stylized face, their method fails to catch the global appearance of faces. For style removal task, we use a paired stylized faces and real faces to train the network. From the fourth column of Figure 4.7 and 4.8, one can see that the details of the face were fuzzy and it loses the identity consistency with respect to the source face. Shiri and Xin [28] introduce a de-stylization approach that exploits only a simple auto-encoder and a standard discriminative network to restore original images. And they utilize only a pixel-wise loss in the generator. They employ a large-scale dataset to train their approach, but it could not produce authentic facial details. As seen in Figure 4.7(e) and 4.8(e), while the method can produce acceptable results, the recovered faces are distorted and lose some details of hairs and color consistency. Compared with the above approaches, the results of our method demonstrate better preservation of face identity and are more consistent with the source faces in colors, as shown in Figure 4.7(f) and 4.8(f). We can observe that the restored faces from different stylized images resemble each other, which indicates the consistency of our method with respect to different artistic styles. We also show more style transfer results in Figure 4.9 and Figure 4.10.
4.3.2 Quantitative Evaluation

**Style Transfer User Study.** We conducted a user study to do a pairwise comparison of style transfer results between our method and the method of Zhu et al. [39] and of Isola et al. [16]. We choose 25 groups of source photos, reference stylized photos and the results generated by different methods (refer to an example in Figure 4.4). For each group, we put the source photos and the reference stylized photos at the first two columns, and then randomly arrange the generated results to the remaining columns. Then we asked 12 people to rank each group based on the style similarity and realistic with respect to the reference stylized photos. Figure 4.11 shows the ranking statistics for this study, which indicates the percentage of each rank for a method. The subjects tended to rate higher for the generated faces that have more pleasing visibility and similarity. The measure demonstrates the superiority of our method compared with other methods.

**Style Removal SSIM Comparison.** To evaluate the recovery performance, we calculate the Structural Similarity Index (SSIM) perceptual metric scores on the whole test dataset including seen (the first row of Figure 4.7 and Figure 4.8) and unseen (the remaining rows) styles. SSIM is a perceptual measure that is based on the image’s visible structures, so we use it here to quantifies the degradation of image quality. Table 4.1 shows that our method outperforms other methods on both seen and unseen styles. Figure 4.7 and Figure 4.8) visually demonstrated the performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Seen Styles</th>
<th>Unseen Styles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isola [16]</td>
<td>0.626</td>
<td>0.623</td>
</tr>
<tr>
<td>Zhu [39]</td>
<td>0.620</td>
<td>0.615</td>
</tr>
<tr>
<td>Shiri [28]</td>
<td>0.793</td>
<td>0.781</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.859</strong></td>
<td><strong>0.845</strong></td>
</tr>
</tbody>
</table>

**Style Removal User Study.** We conducted a user study to do a pairwise comparison of style removal results between our method and the method of Shiri et al. [28], of Zhu et al. [39] and of Isola et al. [16]. We randomly choose 25 stylized faces and use these methods to restore the original faces. Then we get 25 groups of original faces and the recovered faces with different methods (refer to an example in Figure 4.8). For each group, we put the original face at the first column, and then randomly arrange the recovered faces to the remaining columns. Then we asked 12 people to rank each group based on the identity fidelity and visual quality compared with the corresponding ground truth images. Figure 4.12 indicates the ranking statistics for this study, which shows the percentage of each rank for a method. The subjects tended to rate higher for the restored faces that have more pleasing visibility and preserve identities better. The measure indicates that our method outperforms other methods, which is consistent with the above numerical evaluations.
4.3.3 Limitations

However, one limitation of our method is that the network may result in artifacts for faces that have large pose variations. We note that the numbers of images of large pose variations (side-view samples of human faces) and small pose variations are unbalanced. E.g., there are more faces of small pose variations. This makes our synthesized dataset unbalanced. Hence, the facial features of side-view samples of human faces are not fully represented in our dataset. As shown in Figure 4.13, Figure 4.14, Figure 4.15, and Figure 4.16, the style transfer result has some distortions and the style removal network also fails to produce satisfying results. In the future, we would like to explore how to solve large pose variations. For example, we would like to attempt to collect more side-view images to balance the dataset. And we will also try to quantify the performance of style transfer and removal for side-view images on testing datasets that have different percentages of the side-view faces and the faces that have small pose variations.
Figure 4.1: Samples of the synthesized dataset with [17]. (a) Original real face image. (b)-(h) The stylized faces of (a) from Wave, Udnie, the shipwreck of the minotaur, Scream, Candy, Lady, Water, which have been used for training our network. (i)-(m) The stylized faces of (a) from Cup Head, La Muse, Starry Night, Rain Princess, Mosaic, which have not been used for training.
Figure 4.2: Samples of the synthesized dataset with [22]. (a) Original real face image. (b)-(d) The stylized faces of (a) from Mosaic, Starry-night, the shipwreck of the minotaur, which have been used for training our network. (e)-(h) The stylized faces of (a) from Candy, Wave, Rain-princess, and La Muse which have not been used for training.

Figure 4.3: Results of style transfer. We compare with style transfer work. (a) Original images. (b) Stylized images with reference style [22]. (c) Our method. (d) Zhu et al.’s method [39]. (e) Isola et al.’s method [16].
Figure 4.4: Results of style transfer. We compare with style transfer work. (a) Original images. (b) Stylized images with reference style [17]. (c) Our method. (d) Zhu et al.’s method [39]. (e) Isola et al.’s method [16].
Figure 4.5: More results of style transfer. (a) and (d) Original images. (b) and (e) Stylized images with reference style [17]. (c) and (f) Our results.
Figure 4.6: More results of style transfer. (a) and (d) Original images. (b) and (e) Stylized images with reference style [17]. (c) and (f) Our results.
Figure 4.7: Results of style removal. We compare with style removal work. (a) Stylized images with [22]. (b) Ground-truth original images. (c) Zhu et al.’s method [39]. (d) Isola et al.’s method [16]. (e) Shiri et al.’s method [28]. (f) Our method.
Figure 4.8: Results of style removal. We compare with style removal work. (a) Stylized images with [17]. (b) Ground-truth original images. (c) Zhu et al.’s method [39]. (d) Isola et al.’s method [16]. (e) Shiri et al.’s method [28]. (f) Our method.
Figure 4.9: More results of style removal. (a) and (d) Ground-truth original images. (b) and (e) Stylized images with [17]. (c) and (f) Our results.
Figure 4.10: More results of style removal. (a) and (d) Ground-truth original images. (b) and (e) Stylized images with [17]. (c) and (f) Our results.
Figure 4.11: Style Transfer User Study. Performance comparison results between our method and other methods.

Figure 4.12: Style Removal User Study. Performance comparison results between our method and other methods.
Figure 4.13: Limitations of Style Transfer. (a) Original images. (b) Stylized images with reference style [22]. (c) Our results.

Figure 4.14: Limitations of Style Removal. (a) Stylized images with [22]. (b) Ground-truth original images. (c) Our results.

Figure 4.15: Limitations of Style Transfer. (a) Original images. (b) Stylized images with reference style [17]. (c) Our results.

Figure 4.16: Limitations of Style Removal. (a) Stylized images with [17]. (b) Ground-truth original images. (c) Our results.
Chapter 5

Conclusion

In this thesis, we present an approach for transferring style from a reference face to a different face that doesn’t have style and for removing style of the same stylized face to recover the original face at the same time. Our method consists of two networks: the Style Removal Network and the Style Transfer Network. And each network contains a generator and a discriminator, which generates results and forces the generated results to be similar to the target images respectively. We combine multiple loss functions including adversarial loss, a cycle-consistent loss, an identity-preserving loss, and a pixel-level similarity loss to train the two networks together, which allows them to strengthen each other. Our network can extract the style of the reference face and apply it to a source face. At the same time, it can also de-stylize stylized portraits successfully. Compared with other work, the experiments demonstrate that our method produces more convincing results. And we believe this approach could be applied to other similar applications, like applying the style of a color T-shirt to a different white T-shirt.
Bibliography


