Identity-Preserved Face Beauty Transformation with Conditional Generative Adversarial Networks

Zhitong Huang

A Thesis
In The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

August 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Zhitong Huang**

Entitled: **Identity-Preserved Face Beauty Transformation with Conditional Generative Adversarial Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Jinqiu Yang

_____ Examiner
Dr. Yuhong Yan

_____ Supervisor
Dr. Ching Y. Suen

Approved by _____
Chair of Department or Graduate Program Director

_____ 2020

_____
Mourad Debabbi, Interim Dean
Faculty of Engineering and Computer Science

Abstract

Identity-Preserved Face Beauty Transformation with Conditional Generative Adversarial Networks

Zhitong Huang

Identity-preserved face beauty transformation aims to change the beauty scale of a face image while preserving the identity of the original face. In our framework of conditional Generative Adversarial Networks (cGANs), the synthesized face produced by the generator would have the same beauty scale indicated by the input condition. Unlike the discrete class labels used in most cGANs, the condition of target beauty scale in our framework is given by a continuous real-valued beauty score in the range [1 to 5], which makes the work challenging. To tackle the problem, we implement a new triple structure similar to [1], in which the conditional discriminator is divided into a normal discriminator and a separate face beauty predictor. We also develop another new structure called Conditioned Instance Normalization to replace the original concatenation used in cGANs, which makes the combination of the input image and condition more effective. Furthermore, Self-Consistency Loss is introduced as a new parameter to improve the stability of training and quality of the generated image. In the end, the objectives of beauty transformation and identity preservation are evaluated by the pretrained face beauty predictor and state-of-the-art face recognition network. The result shows that certain facial features could be synthesized by the generator according to the target beauty scale, while preserving the original identity.

Acknowledgments

I would like to sincerely appreciate my supervisor Prof. Ching Yee Suen for the continuous support of my master study and research with his patience, motivation, and knowledge. I also would like to thank Concordia University and my faculty of Engineering and Computer Science for providing such an excellent environment to study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Face beauty (or facial attractiveness) has drawn the interest and attention of computer vision and pattern recognition researchers. The topic of face beauty remains arguable because of its subjective properties. However, in recent research [2]–[5], it has been proven that face beauty could be correctly predicted by computer to a certain extent.

Generative Adversarial Networks [6] (GANs) have succeeded in many applications such as style transfer [7]–[9] and image-to-image translation [10]–[12] and become one of the most popular generative models. As one of the extensions, conditional Generative Adversarial Networks [13] (cGANs) allow control of the output through conditioning on some labels, which motivates us to implement face beautification with cGANs. In our work, the inputs to cGANs are the original face image and the target beauty scale indicated by a real-valued beauty score.

The most important consideration of face beautification is to maintain the original identity of the input face because face beautification would become meaningless if the input face is turned into another identity. Xu et al. [14] introduce a mature architecture for identity-preserved face aging. The problem domain of face aging is very similar to ours of face beautification. Therefore, we decide to use IPCGANs [14] as the basic architecture of our work.

The conditioning labels in most applications of cGANs including IPCGANs are formed in

one-hot vectors such as the class labels in [1] and age groups in [15]. However, in the face beauty dataset, the beauty scale of a face is labeled as a continuous real-valued beauty score. One way to address the problem is to discretize the beauty score and divide the face images into separate groups. A significant problem of this method is that the images in two adjacent groups could be very similar when they are close to the threshold. Another problem is the unbalanced data of different groups due to the Gaussian-like distribution of beauty score. Therefore, we decide to use the original continuous labels of beauty score as the input condition feeding to cGANs. However, in our initial experiments, we find that the switch to continuous input condition causes instability in training and degrading of image quality. To address these problems, we introduce modifications to current cGANs including conditioned instance normalization and self-consistency loss.

The choice to use continuous labels also inspires us to enlarge our topic from face beautification to face beauty transformation where the beauty scale of a face can be transformed into arbitrary scales indicated by the input condition. The objective of beauty transformation is not limited to beautify the input face, but to generate a sequence of face images corresponding to different beauty scales indicated by the given beauty score.

To sum up, the main contributions of our work are:

(1) We develop a framework of automatic face beauty transformation with cGANs, where the identity of the input face image is preserved. The experimental results show that certain facial features could be synthesized according to different target beauty scales.

(2) We implement a new framework called conditioned instance normalization to combine the input condition with the input face more effectively. Such an operation improves the quality of generated face images.

(3) We introduce the self-consistency loss which stabilizes the training process and improves the image quality.

Chapter 2

Literature Review

Style Transfer

As one of the most popular applications of GANs, style transfer aims to apply the color and contrast style of a given image (style image) onto another one (content image), of which the content remains unchanged. Correspondingly, the beauty features of a face image could be interpreted as a style of color and contrast, such as makeup, skin color, and texture. Therefore, we could consider beauty transformation as a kind of style transfer and the techniques used in style transfer should also work in beauty transformation. Despite the similarity, the target style in style transfer is extracted from the style image, whereas the target beauty scale in beauty transformation is directly indicated by a real-valued beauty score. This is also the reason why we name our work as beauty transformation instead of beauty transfer, where the beauty scale of the input face image is transformed to a target scale rather than transferred from another face image. Considering the similarities and differences, we inspect instance normalization [8] and its derivatives conditional instance normalization [16] and adaptive instance normalization [9], from which we derive conditioned instance normalization for beauty transformation.

IPCGANs

In IPCGANs [14], a framework of identity-preserved face aging is implemented with cGANs,

in which the input face is transformed into different age groups indicated by a one-hot condition. Our framework uses a similar structure consisting of generator, discriminator, face beauty predictor (FBP) (face age estimator in IPCGANs) and face recognition network (FRN). However, continuous real-valued condition of beauty score is used in our work instead of discrete one-hot label. In our experiments, we find that feeding condition of continuous value to cGANs makes the training extremely difficult and unstable. Therefore, we replace the conditional discriminator with a normal discriminator which only focuses on deciding if an input image is real or fake. Furthermore, we introduce the conditioned instance normalization to replace the traditional concatenation in cGANs. Another difficulty is the lack of data. Compared with various sources of huge face age datasets, face datasets labeled with beauty score is rare and usually very small (less than 10 K images). To relieve the problem, we further introduce the self-consistency loss to improve the quality of generated face and stability of training.

Triple-GAN

Li et al. [1] point out the problem of existing cGANs in semi-supervised learning (SSL), where the discriminator of a two-player structure plays two conflicting roles of discriminator and classifier. To relieve the problem, a triple structure is introduced, where the conditional discriminator is divided into two components: a discriminator and a classifier [1]. In our experiments, we find the triple structure performs better than the traditional architecture of cGANs in aspects of training stability and quality of generated images.

Beholder-GAN

To solve the problem of lack of face beauty datasets, Beholder-GAN [17] labels CelebAHQ [18] dataset with CNNs pretrained with face beauty dataset SCUT-FBP5500 [19]. However, CelebAHQ mainly consists of wild faces (with different poses and expressions, etc.), whereas SCUT-FBP5500 contains almost all frontal face portraits. Additionally, there are 80% of Asian faces and merely 20% of Caucasian faces in SCUT-FBP5500 dataset, while CelebAHQ contains various faces world-wide. Therefore, we believe that the original dataset labeled with beauty score should provide more accurate information of beauty. In our work, only SCUT-FBP5500 dataset is applied throughout the whole training, which

makes cGANs difficult to produce realistic faces. By applying a small dataset, we could also research on how to train a stable cGANs with limited number of samples. Another significant difference is that the identity of the input face is not preserved during the beautification process in Beholder-GAN.

BeautyGAN

The objective of BeautyGAN [20] is to perform instance-level and identity-preserved transfer of makeup style from a reference face image to one without makeup. Compared with BeautyGAN where only the makeup style of the face image is considered, our work examines the general features of a face image in aspects of beauty including makeup, texture, facial features, and illuminance, etc. Nevertheless, the result of our work shows that the makeup of a face also plays a very significant role in beauty transformation, which coincides with the idea of BeautyGAN.

Cycle-Consistency Loss

Zhu et al. [12] introduce cycle-consistency loss in their framework of image-to-image translation where the content of an image is required to be consistent after a cycle of forward and backward translation. The similar concept could be introduced to beauty transformation where a cycle is defined as a process when an image is transformed to target beauty scale and then transformed back to its original scale. However, in our experiments, we find this approach unstable. To stabilize the training, we introduce the self-consistency loss where the input face image is required to stay unchanged after being transformed to its original ground truth beauty scale. With the self-consistency loss, we could obtain generated face images of better quality.

Chapter 3

Background

In this chapter, we will review the fundamental concepts of Generative Adversarial Networks and Instance Normalization, from which we derive our framework and new methodology.

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a framework for training a generative model which can produce realistic samples close to real data distribution [6]. Typically, there are two components in GANs playing against each other: (1) a discriminator which learns the real data distribution and differentiates fake samples from real samples; and (2) a generator which focuses on generating real-like fake samples that could confuse the discriminator and let it believe the generated samples come from the real data distribution. In this adversarial game, the discriminator, which has the knowledge of real data distribution, guides the generator to produce good fake samples towards real data distribution.

The adversarial loss of the minimax game is described in Eqn. (1) listed below:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p_z}[\log(1 - D(G(z)))]. \text{ [6]} \tag{1}$$

where $x \sim p_{data}$ is data sampled from real data distribution, $z \sim p_z$ is sampled from Gaussian noise.

Eqn. (1) shows that the discriminator is trained to maximize log-likelihood of x from real data distribution and negative log-likelihood of $G(z)$ generated by generator, whereas the generator is trained to minimize negative log-likelihood of $G(z)$. The first term in Eqn. (1) lets the discriminator learn the distribution of real data, and the second term is where the "adversarial game" happens: while D tries to maximize the term, G minimizes the term in opposite.

3.1.1 Conditional GANs

From the original GANs, Conditional GANs (cGANs) can be introduced by adding conditional constraints to both the generator and discriminator [13]. Then Eqn. (1) can be rewritten as:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x|y)] + E_{z\sim p_z}[\log(1 - D(G(z|y)|y))]. \tag{2}$$

The extra constraints could be any conditions such as class of objects, identity, and age of faces, and in our work, beauty score of face images. With the constraint of extra information, the discriminator is required to not only judge whether a sample is from real data distribution, but also decide whether the sample meets the condition, e.g., whether an image belongs to a specific class. In the meantime, the generator is also conditioned by the same extra information, which allows the generator to produce samples according to the condition. In our work, the conditional generator is capable of generating face images which meet with the target beauty score.

3.1.2 Conditional GANs with real data input

In previous sections, the generator produces samples from scratch with Gaussian noise input which is adequate when the generated sample is only required to be in a certain distribution, e.g., a group or a class of objects. However, in some cases, the generator

needs to produce specific samples, such as a specified identity in our work, which is almost impossible to achieve when the input is a random noise. In age-cGAN [15], an encoder is applied to extract the identity information from a face image and uses this information as input to the generator. Then the generator is trained to reconstruct the face of the same identity based on the encoded identity information. However, in more recent papers [8], [10], [14], an encoder-to-decoder structured network [21] is applied in the generator, which enables a real data input to the generator and an end-to-end training of the generator. Then Eqn. (2) can be rewritten as:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x|y)] + E_{x\sim p_{data}}[\log(1 - D(G(x|y)|y))]. \tag{3}$$

Note that here real data $x \sim p_{data}$ is input to G instead of random noise $z \sim p_z$.

3.1.3 Least Squares Loss

As mentioned in [22], the original objective function of GANs may lead to unstable learning, which is mainly caused by the vanishing gradients. Specifically, the sigmoid function used at output of the discriminator and the logarithm function used in loss function both have a "flat" region where the value of gradient is small. To address this problem, the least squares loss function in LSGANs [22] is applied and Eqn. (3) can be written as:

$$\begin{aligned} \min_D V(D) &= \frac{1}{2}E_{x\sim p_{data}}[(D(x|y) - a)^2] + \frac{1}{2}E_{x\sim p_{data}}[(D(G(x|y)|y) - b)^2]. \\ \min_G V(G) &= \frac{1}{2}E_{x\sim p_{data}}[(D(G(x|y)|y) - c)^2]. \end{aligned} \tag{4}$$

Note that the output of the last fully connected layer in discriminator is no longer passed through sigmoid function which means $D(x|y)$ is in range $(-\infty, +\infty)$ rather than $(0, 1)$.

In practice, a is usually set to $a = 1$ for real samples and b is set to $b = 0$ for fake samples. Accordingly, c is set to $c = a = 1$, which enables the generator to produce samples similar to the real samples. Compared with log-likelihood loss in Eqn. (3) where generated samples are trained towards positive infinity, the least squares loss could motivate the generator to produce samples closer to real data by pushing the generated samples towards the real

samples [22].

3.2 Instance Normalization

Instance normalization (IN) [8] is a kind of contrast normalization which helps to improve the quality of image generation. As shown in Figure 3.1, each channel of the feature tensor $x \in \mathbb{R}^{B\times C\times W\times H}$ is normalized to have zero mean and unit variance:

$$y_{tkij} = \frac{x_{tkij} - \mu_{tk}}{\sqrt{\sigma_{tk}^2 + \epsilon}} \tag{5}$$

where x_{tkij} is the $tkij$-th element of $x \in \mathbb{R}^{B\times C\times W\times H}$. i and j are the spatial dimensions, k denotes the channel of the feature maps and t represents the t-th image in the batch. ϵ is a small number added to avoid square root of zero.

The statistics of mean and variance is calculated separately for each channel of each feature map:

$$\begin{aligned} \mu_{tk} &= \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H} x_{tkij} \\ \sigma_{tk}^2 &= \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}(x_{tkij} - \mu_{tk})^2 \end{aligned} \tag{6}$$

where μ_{tk} is the mean and σ_{tk}^2 is the variance of the k-th channel of the t-th image.

After the normalization, a set of learnable affine parameters is applied to the normalized feature, which allows the feature to have new mean and variance:

$$y_{tkij} = \gamma\left(\frac{x_{tkij} - \mu_{tk}}{\sqrt{\sigma_{tk}^2 + \epsilon}}\right) + \beta \tag{7}$$

where γ and β are the learnable affine parameters corresponding to the new variance and mean respectively.

In our experiments, we find that instance normalization improves the quality of the generated images and also accelerates the convergence of the generator.

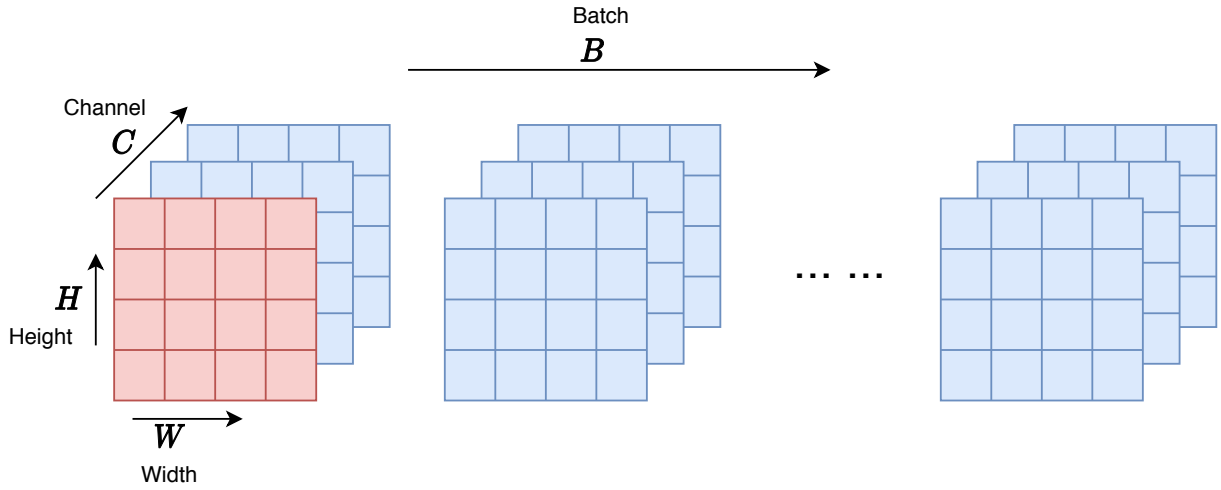

Figure 3.1: Instance Normalization. A batch of features with size $B \times C \times W \times H$, where W and H represent the spacial dimensions, C is the number of channels and B is the batch size. The statistics of instance normalization is calculated within each channel of the feature maps (the red region indicates one of them).

3.2.1 Conditional Instance Normalization

Conditional Instance Normalization (CIN) is a technique used in style transfer, which enables a single generator to transfer multiple styles. Instead of learning only one set of affine parameters, there is a group of parameters learned in conditional instance normalization [16], where each pair of parameters in the group corresponds to a specific style. As shown in Figure 3.2, different pairs of affine parameters from (γ_1, β_1) to (γ_n, β_n) corresponding to different styles are trained. The mean and variance of the input feature are swapped to new mean and variance defined by the affine parameters. During testing, one can get generated images with different styles by choosing different pairs of affine parameters.

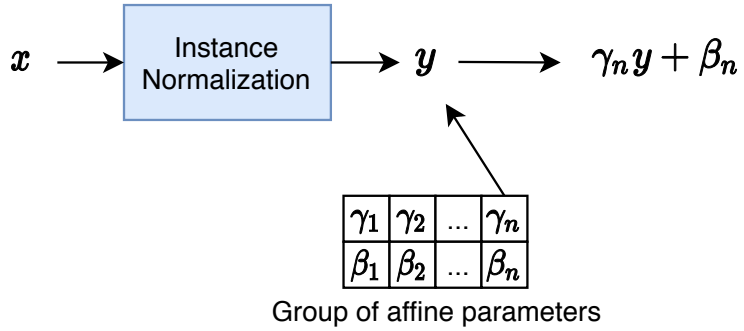

Figure 3.2: Conditional Instance Normalization. x is the input feature, y is the normalized feature with zero mean and unit variance and $\gamma_n y + \beta_n$ is the new feature with new mean of β_n and variance of γ_n.

3.2.2 Adaptive Instance Normalization

To transfer the image to an arbitrary style, Huang and Belongie [9] introduce a new framework named Adaptive Instance Normalization (AdaIN). As shown in Figure 3.3, the input image and style image are fed to the same encoder which extracts high level features. Then the feature x of the input image is normalized to zero mean and unit variance through IN. After that, the mean and variance (γ_s, β_s) of the feature x_s of the style image are applied to the normalized feature y. This operation swaps the mean and variance of the input feature to those of the style feature. With the stylized feature $\gamma_s y + \beta_s$, a decoder with reversed structure as the encoder could generate an output image with content of the input image and style of the style image.

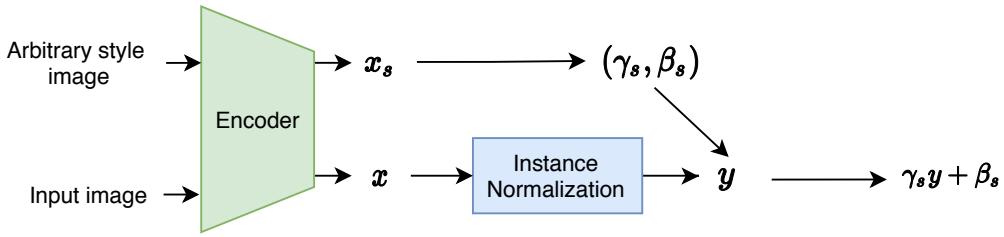

Figure 3.3: Adaptive Instance Normalization. x and x_s are the extracted features of the input image and style image respectively. (γ_s, β_s) are the channel-wise mean and variance of x_s. y is the normalized feature of x with zero mean and unit variance. $\gamma_s y + \beta_s$ is the new feature with mean of β_s and variance of γ_s.

3.3 Cycle-Consistency Loss

In an image-to-image translation framework [12], a generator G is trained to transfer the images from style X to Y and another generator F is supposed to do the inverse mapping: $Y \rightarrow X$. To enhance the capability of the generators and improve the quality of the generated images, cycle-consistency loss is introduced to maintain the consistency of G and F:

$$\mathcal{L}_{cyc}(G, F) = ||F(G(x)) - x||_1 + ||G(F(y)) - y||_1 \tag{8}$$

where $||.||_1$ is the L1 norm.

As in Eqn. (8), the image after the cycles through two generators ($x \rightarrow G(x) \rightarrow F(G(x))$ and $y \rightarrow F(y) \rightarrow G(F(y))$) should be consistent with the original image.

In starGAN [11], the idea of cycle-consistency loss is extended to conditional GANs with only one generator:

$$\mathcal{L}_{rec} = ||G(G(x, c'), c)||_1 \tag{9}$$

where c is the original ground truth label of x and c' is the label of another domain. As in Eqn. (9), an image is translated to another domain c' and then brought back to its original domain c. The generator is required to maintain the content of the image in this cycle.

Chapter 4

Methodology

In this Chapter, we will first present an overview of our framework and then the overall architecture and procedures will be introduced. After that, each individual component and procedure will be described in detail. At last, we will introduce the Conditioned Instance Normalization and Self-Consistency Loss.

4.1 Overview

The beauty transformation is implemented with a Conditional Generative Adversarial Networks (cGANs) [13], which can generate images constrained on some conditions. In our framework, the input is the original face image sampled from real face dataset, and the output is a synthesized image conditioned on a target beauty score which indicates the target beauty scale we want the generator to produce. Meanwhile, the generator is also expected to generate realistic face images and preserve the identity of the input face, i.e. the output face shares the same identity as the input face.

4.2 Overall Architecture

As shown in Figure 4.1, a similar architecture used in IPCGANs [14] for face-aging is applied in our framework. The input face $x_u \in \mathcal{D}$, sampled from a real face dataset $\mathcal{D}$, is first

forwarded to the generator which consists of four parts: (1) Down-sampling Convolutional Layers which extract high-level features of input face; (2) Conditioned Instance Normalization block which combines the extracted feature and target beauty score y_u; (3) Residual Blocks [23]; and (4) Up-sampling Transposed Convolutional Layers which up-samples the feature conditioned on target beauty score y_u to the same size as the input face x_u.

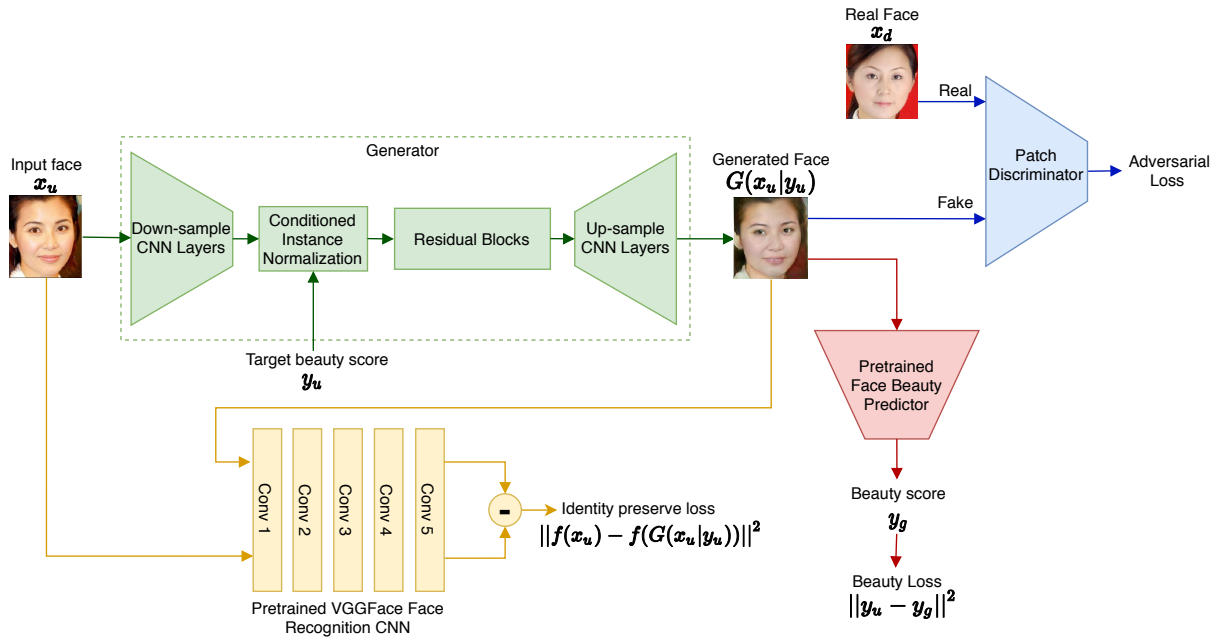

Figure 4.1: Overall structure. Input face x_u is forwarded to the generator which produces output face $G(x_u, y_u)$ conditioned on target beauty score y_u. The generated face is then fed to three other networks to compute the corresponding losses.

The generated face $G(x_u, y_u)$ is then input to the discriminator whose task is to distinguish between fake images (generated faces) and real images($x_d \in \mathcal{D}$). Through Adversarial Loss, the generator is trained to "fool" the discriminator by letting it believe the generated image is real [6]. Thus, the generator could produce images which are considered to be from the same distribution as real images by the discriminator.

Furthermore, the generated face is fed to a Face Recognition Network (FRN) which is pretrained to recognize the identity of human face. In this network, the feature of the intermediate layer, which is supposed to extract high-level feature related to the identity of a face, is selected to be the identity feature. Both the generated face and the original input face is forwarded to the network, after that, the mean-squared difference between their

identity features is defined as Identity-Preserve Loss. By minimizing the Identity-Preserve Loss, the generator is trained to generate a face with same identity as the original input face.

Finally, a pretrained Face Beauty Predictor (FBP) is used to label the beauty score of the generated face. Recall that the generated face $G(x_u, y_u)$ is conditioned on a target beauty score y_u, therefore, the labeled beauty score y_g of the generated face is supposed to be equal to the target beauty score y_u. Thus, the Beauty Loss could be defined as the mean-squared difference between the labeled beauty score y_g of the generated face and desired target beauty score y_u. By minimizing this Beauty Loss, the generator could output a face which has the same beauty score as the target.

Now, by combining all the three losses mentioned above, the generator could transform the input face into a face with the desired target beauty score while preserving its identity and making it look realistic.

4.3 Conditional GANs Module

In cGANs introduced in Section 3.1.3, the generator is trained to output $G(x|y)$ based on the input x and conditioned on y. Meanwhile, the discriminator is optimized to decide whether a pair of sample x and condition y is real or fake, i.e., whether the sample x comes from the data distribution conditioned on y. This is feasible when the condition y is a group or class label and the discriminator only needs to perform work similar to a classifier, i.e., determine whether the input sample x belongs to the class y. However, in our work, the condition y is a continuous real-valued beauty score instead of a class label, which makes it difficult to determine whether the input sample x comes from the distribution specified by y because every single sample in the real data distribution has a different value of y.

To tackle this problem, we apply the structure introduced in Triple-GAN [1]. The generator is kept unchanged but the task of discriminator is decomposed into two parts: (1) whether the sample comes from the real data distribution; and (2) whether the sample matches the condition. The first task could be accomplished with a normal discriminator without

condition and the second task is fulfilled with a separated face beauty predictor (see detail in Section 4.4).

Therefore, the adversarial loss for our cGANs is:

$$\begin{aligned} \mathcal{L}_D &= E_{x\sim p_{data}}[(D(x)-1)^2] + E_{x\sim p_{data}}[(D(G(x|y)))^2]. \\ \mathcal{L}_G &= E_{x\sim p_{data}}[(D(G(x|y))-1)^2]. \end{aligned} \tag{10}$$

Both D and G are trained to minimize their own loss $\mathcal{L}_D$ and $\mathcal{L}_G$ respectively. Note that we directly use the MSE without multiplying with $\frac{1}{2}$.

4.4 Face Beauty Predictor Module

The main objective of our work is to transform the input face to the target beauty score indicated by the input condition. At the input side of the generator, the target beauty score is input as condition so that the generator could output face $G(x|y)$ conditioned on beauty score y. To ensure the beauty score of the output face equals to the target, a Face Beauty Predictor (FBP) is introduced. FBP is a pretrained deep convolutional network which can predict the beauty score of a given face and the parameters of FBP are fixed during the whole training process. The performance of FBP is measured in terms of root mean squared error, mean absolute error, and Pearson correlation as shown in Table 4.1.

RMSE	**MAE**	**PC**
0.2272	0.1753	0.9437

Table 4.1: Performance of pretrained face beauty predictor.

To ensure the beauty score of the output face meets the target, the generated face is fed to the FBP and the beauty loss is defined as the mean square error (MSE) between the predicted beauty score and the target:

$$\mathcal{L}_{beauty} = ||FBP(G(x|y)) - y)||^2 \tag{11}$$

where y is the target beauty score and $FBP(.)$ is the predicted beauty score.

With this beauty loss, the generator could get essential information related to face beauty from the FBP and thus generate faces with the desired beauty score.

4.5 Identity-Preserve Module

Another significant objective of our work is to maintain the identity of the input face, i.e., the identity of the input face and the generated face should be the same. As stated in [14], the identity information should be encoded in the intermediate layers of a face recognition network (FRN). Therefore, to preserve the identity of the input face, a pretrained FRN is introduced to extract the identity features of both the input face and the generated face. Then the identity preserve loss is defined as the mean square error (MSE) between the identity features:

$$\mathcal{L}_{identity} = ||h(G(x|y)) - h(x)||^2 \tag{12}$$

where $h(.)$ stands for the intermediate features of the FRN. We choose the 5th convolutional layer as the identity features.

4.6 Conditioned Instance Normalization

In normal cGANs, the input image and condition are combined through simple concatenation. As shown in Figure 4.2, assume that the input image is of size $w \times h \times c$, and the input condition is of size $1 \times 1 \times c'$. The input condition y is first reshaped to $w \times h \times c'$ through simple replication and then concatenated with the input image x. After that, the concatenated input of size $w \times h \times (c + c')$ is fed to the generator which produces the output image $G(x|y)$.

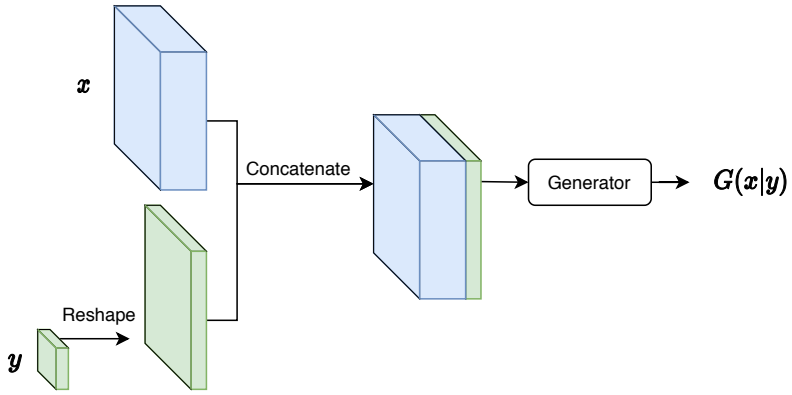

Figure 4.2: Concatenation in cGANs. x is the input image of size $w \times h \times c$, y is the input condition of size $1 \times 1 \times c'$ and $G(x|y)$ is the generated image.

Through the concatenation, the generator could generate an image based on both the input image x and condition y. However, this operation is inefficient because the input condition only has one single value in each channel and after the replicate reshape, each channel will be filled with the same single value. In Appendix A, a detailed proof will be given to show that the condition only acts as a bias term after the concatenation.

Therefore, to combine the input image and condition more efficiently, we introduce a new framework named Conditioned Instance Normalization (CondIN) based on the ideas of CIN (details in Section 3.2.1) and AdaIN (details in Section 3.2.2). The experiments in [9], [16] show that tuning the affine parameters of IN could control the style of the generated image, which proves that the mean and variance of the extracted feature of an image could represent the style of that image. Therefore, if we consider beauty scale as kind of style of face images, we could change the beauty scale of a face image by tuning the mean and variance of its intermediate feature. In CIN and AdaIN, the affine parameters are either selected from a group set or extracted from a style image. However, in our work, the input condition is a single real-valued number of beauty score. To map the beauty score to the affine parameters, we use a network which consists of three fully connected layers.

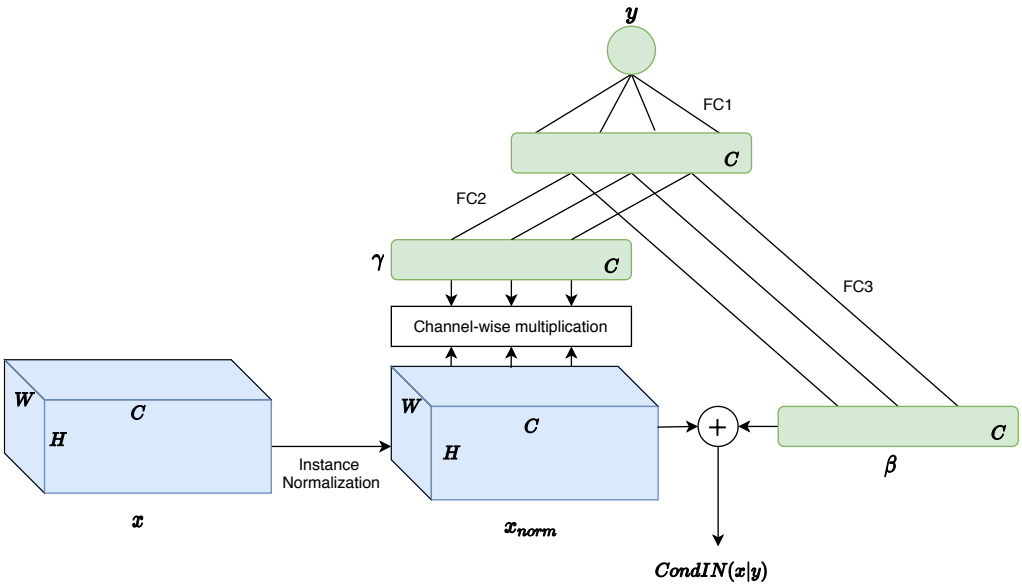

Figure 4.3: Conditioned Instance Normalization. x is the feature of size $C \times W \times H$ and x_{norm} is the normalized feature. y is the single value of beauty score. FC1, FC2, and FC3 denote the fully connected layers. γ and β are the channel-wise mean and variance to be aligned to x_{norm}. $CondIN(x|y)$ is the feature conditioned on y through CondIN.

As shown in Figure 4.3, the intermediate feature x of the input face image is first normalized to have zero mean and unit variance through IN. Meanwhile, the input condition y of beauty score is mapped to γ and β with the same size as the channel size C through three fully connected layers FC1, FC2, and FC3. After that, channel-wise multiplication and addition are applied to align the new variance γ and mean β to the normalized feature x_{norm}. Finally, we could output a feature with new mean and variance conditioned on the beauty score y.

The choice of the activation function of the three fully connected layers is also important. We choose sigmoid function for FC1 to allow more nonlinearity for the mapping. As for FC3 which outputs the mean of the feature, we use the raw output without activation function to enlarge the range of the output mean. To stabilize the training process, sigmoid function is applied to FC2 as activation function which limits the range of the output variance. After that the output variance is multiplied by 2 because of two reasons: (1) the output variance should have a larger range of $(0, 2)$ instead of $(0, 1)$; (2) the weights and bias in fully connected layer are initialized with small values, thus the output after sigmoid should

be around 0.5 and after multiplied by 2, the output variance is around 1.0 at the beginning of training which increases the stability.

Finally, the formula of CondIN is:

$$CondIN(x) = \gamma \odot IN(x) \oplus \beta \tag{13}$$

where $\odot$ and $\oplus$ are channel-wise multiplication and addition. γ and β are the mapped mean and variance from the condition:

$$\begin{aligned} \gamma &= 2\delta\left(g_2[\delta(g_1[y])]\right) \\ \beta &= g_3[\delta(g_1[y])] \end{aligned} \tag{14}$$

where $\delta(.)$ is the sigmoid function. $g_1[.]$, $g_2[.]$, and $g_3[.]$ are the fully connected layers FC1, FC2, and FC3 respectively.

Compared with the normal way to combine the feature and condition through concatenation where the condition acts as a bias term which affects only the mean of the feature (see details in Section 4.6 and proof in Appendix A), CondIN allows the input condition to control both the variance and mean of the feature. Therefore, our framework CondIN makes the combination of the feature and condition more efficient.

4.7 Self-Consistency Loss

In conditional GANs module (Section 4.3), the generator is trained to generate faces which are realistic and close to the real face distribution by minimizing the adversarial loss. Furthermore, by minimizing the identity preserve loss in Section 4.5, the identity of the input face is maintained. However, in our experiments, the quality of the generated faces is unsatisfactory with blurs and artifacts. To further improve the quality of the generated faces, we introduce consistency loss to our framework.

Cycle-consistency loss (see details in Section 3.3) is proved to be effective in GANs where the images are translated between two or more domains such as styles and classes. However,

in our work, beauty score of face images contains infinite continuous domains. To improve the quality of the generated faces and stability of the training, we introduce a simplified version of consistency loss called self-consistency loss which only considers one forward pass through the generator:

$$\mathcal{L}_{consist} = ||h(G(x|y)) - h(x)||^2 \tag{15}$$

where y is the ground truth beauty score of x and $h(.)$ is the intermediate feature of a pretrained face recognition network.

As shown in Eqn. (15), the generator is required to duplicate the content of the input image x if the input condition y is exactly the ground truth beauty score of the input image. As for the feature $h(.)$, we use the same face recognition network as in Section 4.5 and experiment with raw pixel and all intermediate layers of the FRN. At last, we find that raw pixel is the optimal option, therefore, the self-consistency loss is defined as:

$$\mathcal{L}_{consist} = ||G(x|y) - x||^2 \tag{16}$$

As shown in Figure 4.4, a sampled face image x_c is input to the generator with the target beauty score being equal to its ground truth beauty score, then the pixel-wise mean square error between the input face and generated face is minimized. In our experiments, this approach improves the performance of the generator by a large scale.

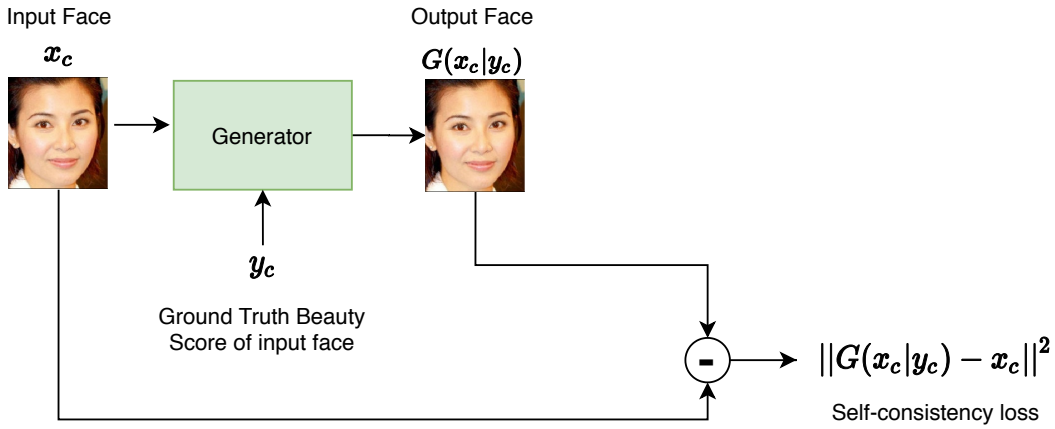

Figure 4.4: Self-consistency loss.

Another significant consideration to use self-consistency loss is the trade-off between identity preserve loss and beauty loss. If we use low-level features or raw pixels for identity preserve loss to improve the quality of the generated face, the generator may learn to ignore the input condition of beauty score and thus always produces identical face as the input face. In opposite, if high-level features are applied in identity loss to ensure enough freedom for the generator to synthesize the face according to input beauty score, the quality of the generated face will become worse. Because of this trade-off, we need to sacrifice the quality of the generated face for accurate beauty transformation or vice versa. However, in self-consistency loss, we require the generator to produce identical face only when the input condition is the ground truth beauty score of the input face. This method could improve the quality of the generated faces without affecting the ability of beauty transformation of the generator.

4.8 Training

4.8.1 Training Loss

The training loss of our conditional GANs consists of four parts: adversarial loss, beauty loss, identity-preserve loss, and self-consistency loss:

$$\begin{aligned} D_{loss} &= \lambda_1 \mathcal{L}_D \\ G_{loss} &= \lambda_2 \mathcal{L}_G + \lambda_3 \mathcal{L}_{beauty} + \lambda_4 \mathcal{L}_{identity} + \lambda_5 \mathcal{L}_{consist} \end{aligned} \tag{17}$$

where D_{loss} is the training loss of the discriminator and G_{loss} is the training loss of the generator. λ_1 and λ_2 control the adversarial game between D and G. λ_3 and λ_4 control to what extent the beauty of the face is edited and the identity is maintained, respectively. λ_5 controls the effect of self-consistency loss. For different losses, we may use different data

samples (x_d, (x_u, y_u) and (x_c, y_c)). Therefore, we rewrite all terms in Eqn. (17):

$$\begin{aligned}
\mathcal{L}_D &= (D(x_d) - 1)^2 + (D(G(x_u|y_u)))^2 \\
\mathcal{L}_G &= ||D(G(x_u|y_u)) - 1||^2 \\
\mathcal{L}_{beauty} &= ||FBP(G(x_u|y_u)) - y_u||^2 \\
\mathcal{L}_{identity} &= ||h(G(x_u|y_u)) - h(x_u)||^2 \\
\mathcal{L}_{consist} &= ||G(x_c|y_c) - x_c||^2
\end{aligned} \tag{18}$$

x_d, x_u, and x_c are sampled from the same dataset but with different samplers. y_c is the ground truth label of x_c. y_u is not the label of x_u but sampled from a uniform distribution in range $[1.0, 5.0]$.

4.8.2 Training Algorithm of cGANs

Algorithm 1 Training cGANs

foreach training iteration **do**

- Sample a batch of face images $x_d \sim p_{data}$, a batch of face images $x_u \sim p_{data}$, a batch of labeled face images with beauty scores $(x_c, y_c) \sim p_{data}$, and a batch of random beauty scores y_u from uniform distribution in range $[1.0, 5.0]$.
- Feed (x_u, y_u) to G and get the generated faces: $x_g = G(x_u, y_u)$.
- Update D by descending along its gradient:
$$\nabla_{\theta_D} \left[\lambda_1 \left[(D(x_d) - 1)^2 + (D(x_g))^2\right]\right]$$
- Pass the generated faces x_g to the face beauty predictor: $y_g = FBP(x_g)$.
- Extract identity features of x_u and x_g with the face recognition network: $f_u = h(x_u)$, $f_g = h(x_g)$.
- Update G by descending along its gradient:
$$\nabla_{\theta_G} \left[\lambda_2 (D(x_g) - 1)^2 + \lambda_3 (y_g - y_u)^2 + \lambda_4 (f_g - f_u)^2\right]$$
- Feed (x_c, y_c) to G and get the generated faces: $x'_c = G(x_c, y_c)$.
- Update G again by descending along its gradient:
$$\nabla_{\theta_G} \left[\lambda_5 (x'_c - x_c)^2\right]$$

end

4.9 Analysis on Beauty transformation

To understand how the training loss improves the performance of beauty transformation, we must identify the difference between beauty transformation and other research topics such as image-to-image translation [10], [12], and style transfer [7], [8]. In style transfer, the generator maps the input image from one domain to another domain specified by the input condition. As shown in Figure 4.5, these domains are separated and defined by discrete index and label. Usually, different styles will have totally different intensity, texture, and contrast. However, in beauty transformation, faces of a same identity with different beauty scores should look similar to certain extent and cannot be considered as different styles or domains. For example, the faces of the same identity with beauty scores of 3.0 and 3.1 should look almost the same. Therefore, we consider the beauty of a face as a continuous number axis ranging from 1.0 to 5.0 as shown in Figure 4.6. Each point on the axis represents a face image with beauty score specified by the value of the point.

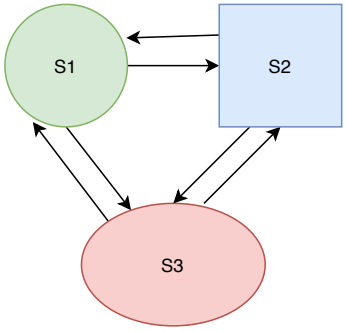

Figure 4.5: Style transfer. S1, S2, and S3 are different domains with different styles. Arrows stand for transfer between styles

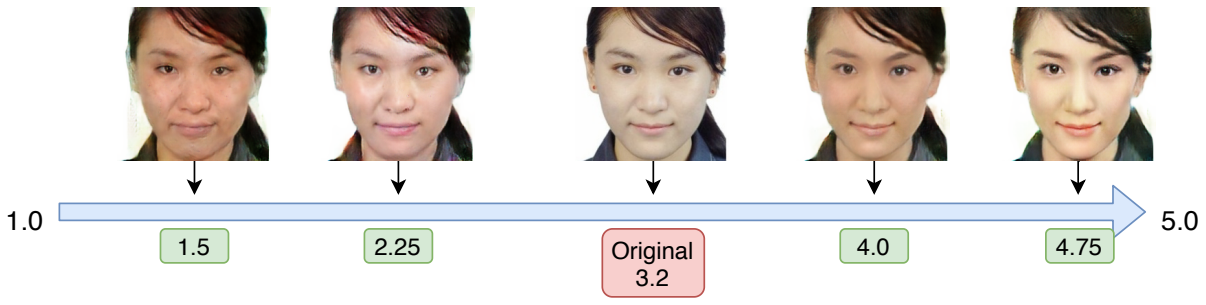

Figure 4.6: Beauty transformation. Beauty scores range from 1.0 to 5.0. Each point on the axis corresponds to a face image with specified beauty score.

With this assumption, our objective of beauty transformation becomes generating all the face images on the axis given the original face. To achieve this objective, identity-preserve loss is first applied to ensure all the face images belong to the same identity as the original face. Then beauty loss guarantees the beauty score of the generated face equals to the value of corresponding points on the axis. However, with only the identity and beauty information, the generator is not able to produce the details of the face. This problem is solved by self-consistency loss, which actually provides a reference point for beauty transformation. Feeding a face image along with its ground truth beauty score shows the generator that the generated face image on that point should be the same as the original one. Therefore, the generator will learn to produce identical face as the input face when the input condition is close to the ground truth beauty score, and for other conditions, the generator will edit some details based on the reference face.

Chapter 5

Experimental Setup

5.1 Dataset

We use SCUT-FBP5500 [19] as the face beauty dataset throughout the whole training and testing processes. The dataset consists of four racial and gender groups: Asian female, Asian male, Caucasian female, and Caucasian male with 2000, 2000, 750 and 750 images, respectively. Since we believe that the evaluation of face beauty is closely related to the race and gender of individuals, we only use the 2000 Asian female faces in our experiments (1800 for training and 200 for testing). The face beauty is labeled by 60 raters aged from 18-27 (average 21.6), who annotate each face image with a scale of beauty in [1.0, 2.0, 3.0, 4.0, 5.0] [19]. Then the beauty score of a face image is calculated by averaging the scores from the 60 raters. Therefore, the beauty score is theoretically in range [1.0 to 5.0], however, approximately 98.55% of the scores drop in the interval [1.5 to 4.5]. Due to this fact, the quality of the generated faces is not ideal when the input target beauty score is in [1.0 to 1.5] and [4.5 to 5.0].

5.2 Image Preprocessing & Data Augmentation

As shown in Figure 5.1, each face image in the dataset is annotated with 86 landmarks [19]. With these landmarks, we can locate the rectangular bounding box (blue rectangle)

of face by taking the minimum and maximum values of $x - y$ coordinate. Then the size of the bounding box is scaled up by a factor 1.1. Finally, the face image is cropped from the square bounding box (green square) with the side length being equal to the longer side of the rectangle.

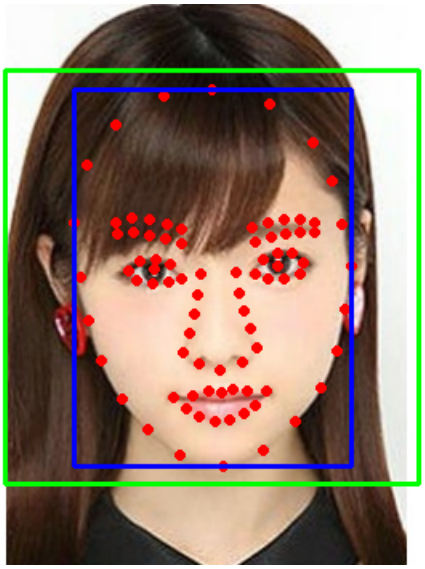

Figure 5.1: Face Crop. The red points are the 86 landmarks. The blue rectangle indicates the face bounding box where the green square represents the scaled square cropping region.

After cropping, the face image is resized to 236×236. During training, the resized image is randomly cropped to 224×224 and then random horizontal flip is applied for data augmentation. For testing, 224×224 center crop is applied with no horizontal flip. The pixel value is normalized to range $[-1.0, 1.0]$ to fit the range of *tanh* which we use as the output activation function of the generator.

In experiments, we find that using the cropped face improves the performance of the face beauty predictor and the quality of the generated face compared with the raw face image. Data augmentation could stabilize the training process by slowing down the convergence of the discriminator.

5.3 Details of Training

We train our conditional GANs using Adam optimizer ($\beta_1 = 0.5$ and $\beta_2 = 0.99$) for 300 epochs with batch size of 16. The initial learning rate is 0.0001 and decayed by 0.3 every 100 epochs. The weights of adversarial loss are set to $\lambda_1 = \lambda_2 = 1.0$ and the weight of beauty loss is $\lambda_3 = 1.0$. The optimal weights of identity preserve loss and self-consistency loss are $\lambda_4 = 0.8$ and $\lambda_5 = 0.08$, respectively. In experiments, we find that all the loss weights can be set to around 1.0 which does not affect the result much, except the weight of self-consistency loss, which must be set to around 0.1. There could be many possible combinations of loss weights, however, we find the settings above are the most stable and converge fastest during random search.

5.4 Network Details

In the following sections, the detailed information and structure of all the networks used in our experiments will be described. We use the notation "$k \times k$ conv - c - St" to represent a convolutional layer with kernel size k, stride t, and output channels of c. Max-pooling layer with kernel size k and stride t is denoted as "$k \times k$ maxpool - St". Fully connected layer with x units is denoted as "fc - x".

5.4.1 Face Beauty Predictor

We use Resnet-18 [23] as the face beauty predictor and replace the last fully connected layer as in Figure 5.2. The network is first pretrained with ImageNet [24] and then finetuned with the face beauty dataset. We fix the parameters of the whole network during the training of cGANs. Batchnorm [25] layer and ReLU activation are applied after each convolutional layer. We use no norm layer or activation function for the last fully connected layer which outputs the real-valued beauty score.

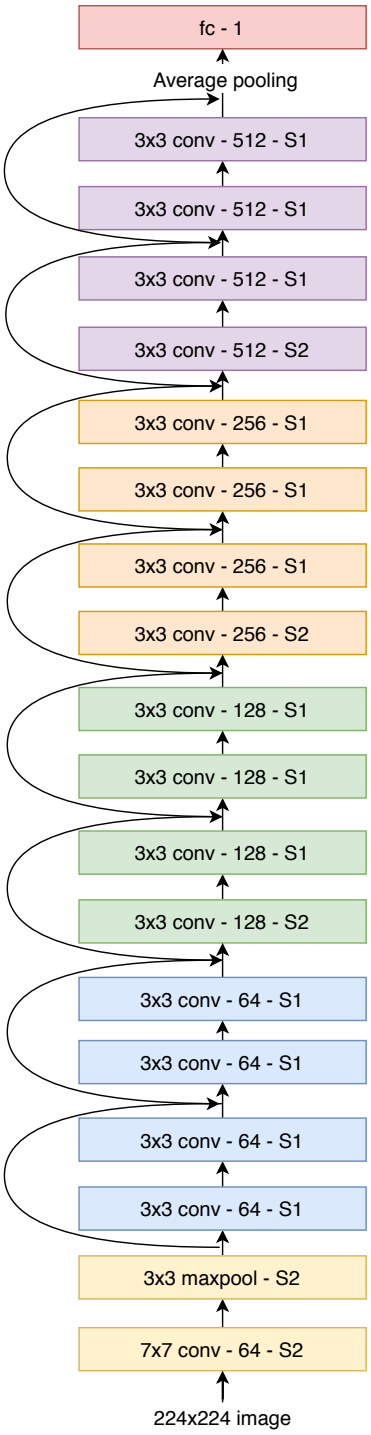

Figure 5.2: FBP Structure. The input is a 224×224 image and the output is a real-valued beauty score. The skip arrow stands for residual connection.

5.4.2 Face Recognition Network

We use a deep convolutional face recognition network [26] pretrained with VGGFace dataset as our FRN to extract the identity feature of the face image. As shown in Figure 5.3, each convolutional layer and fully connected layer is followed by a Batchnorm [25] layer and ReLU activation except the last fully connected layer.

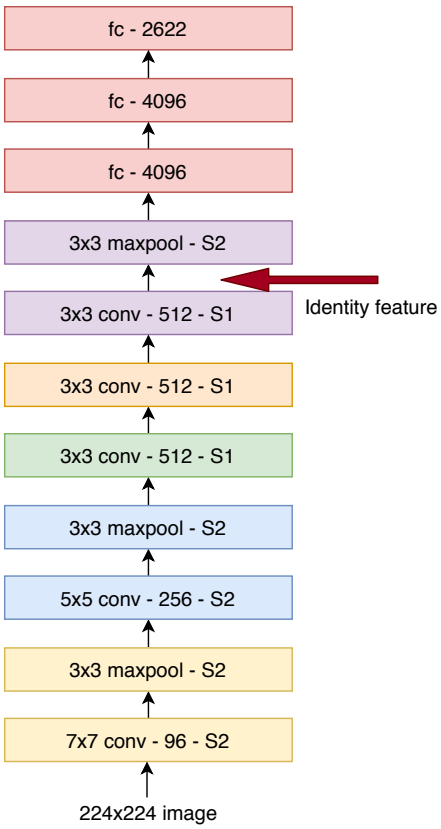

Figure 5.3: FRN Structure. The input is a 224×224 image and the output consists of 2622 identity indices. The red arrow indicates the feature layer used as identity feature.

As in Eqn. (12), a feature layer needs to be selected to represent the identity feature $h(.)$ which is significant as it directly affects the quality of the generated face and whether the generated face maintains the same identity as the input face. In our experiments, we line search from the first convolutional layer to the last second fully connected layer and finally choose the intermediate feature of the 5^{th} convolutional layer ($conv_5$) to be the identity feature of the face. During the whole training, the parameters of the whole network are fixed.

5.4.3 Discriminator

We use an architecture similar to the Patch Discriminator in [10]. As shown in Figure 5.4, the discriminator consists of 6 convolutional layers and the outputs are 4×4 scores, each of which indicates a 64×64 patch of the input image is real or fake. We use Leaky-ReLU activation with a negative slope of 0.2 for each convolutional layer except the last layer. To avoid border artifacts, no pooling layers are introduced and reflection padding is used in the first layer. We replace batchnorm with weightnorm [27] for stability [28]. Dropout with drop rate 0.5 is applied after the last second layer to provide noise and limit the discriminative power of the discriminator.

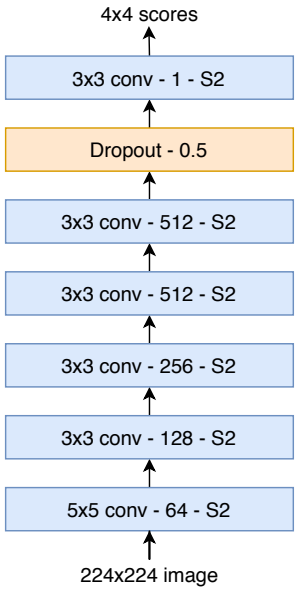

Figure 5.4: Patch Discriminator. The input is a 224×224 image and the outputs are 4×4 scores.

5.4.4 Generator

As shown in Figure 5.5, the input image is down-sampled by 4 convolutional layers (in blue) and then the extracted feature is normalized and combined with the target beauty score through conditioned instance normalization (in yellow). The combined feature is fed to n residual blocks (in purple) and then up-sampled by 3 transpose-convolutional layers (in green). Finally, a convolutional layer (in red) maps the feature to 3 RGB channels

followed by a *tanh* activation. The pixel value of the generated face is zero-centered in range $[-1, 1]$. No normalization is used for all the layers and ReLU activation is applied to the first three down-sample layers and up-sample layers.

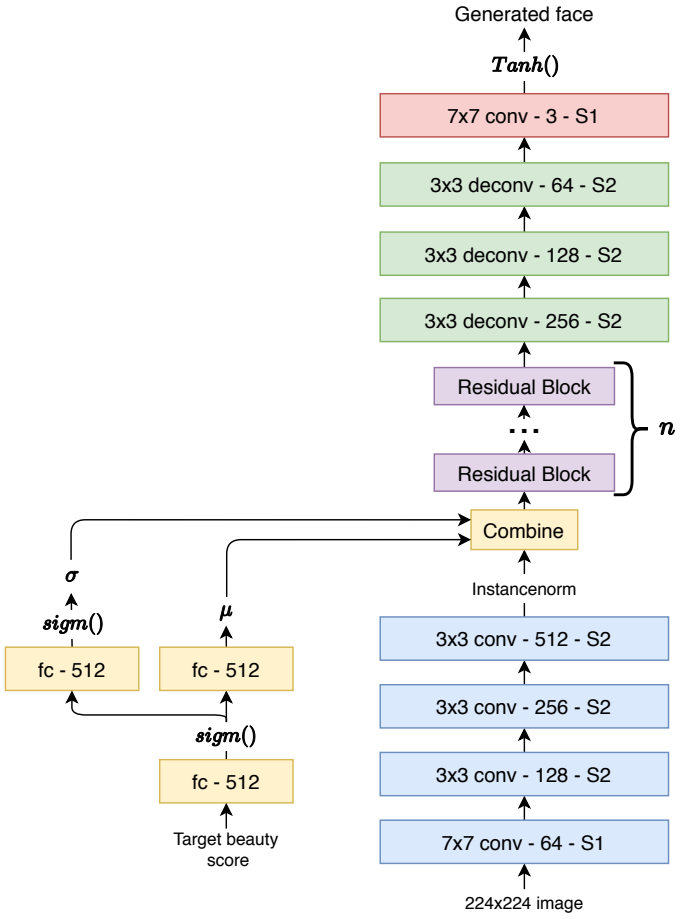

Figure 5.5: Generator. The input 224×224 face image is combined with the target beauty score to output the generated face.

Each residual block consists of two convolutional layers and a skip connection from input to output as shown in Figure 5.6. In our experiments, the optimal number of residual blocks is 4.

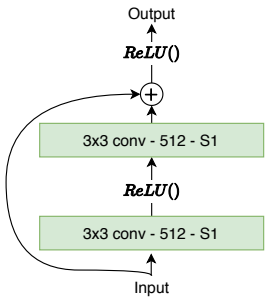

Figure 5.6: Residual Block. Consists of two 3×3 convolutional layers and a skip connection from input to output.

Chapter 6

Experimental Results

6.1 Loss

In this section, we train our GANs with the configurations and settings described in Chapter 5 and show the training and testing loss throughout the training. We use the square root of all the losses because RMSE could clearly indicate the real difference which is helpful for evaluation.

6.1.1 Adversarial Loss of Conditional GANs

From Figures 6.1 and 6.2, we could observe that the adversarial loss of discriminator for both real and fake images drops towards zero at the beginning of the training, which indicates the discriminator could easily distinguish between the real and fake images. As the training proceeds, the generator gradually learns to generate face-like images. After nearly 100 epochs, the adversarial losses for both the discriminator and generator converge to 0.5 which coincides with the proposition described in [6] (the global optimality of GANs appears when $D(x) = D(G) = 0.5$). After 100 epochs, we reduce the learning rate for fine-tuning and the adversarial losses begin to fluctuate around 0.5.

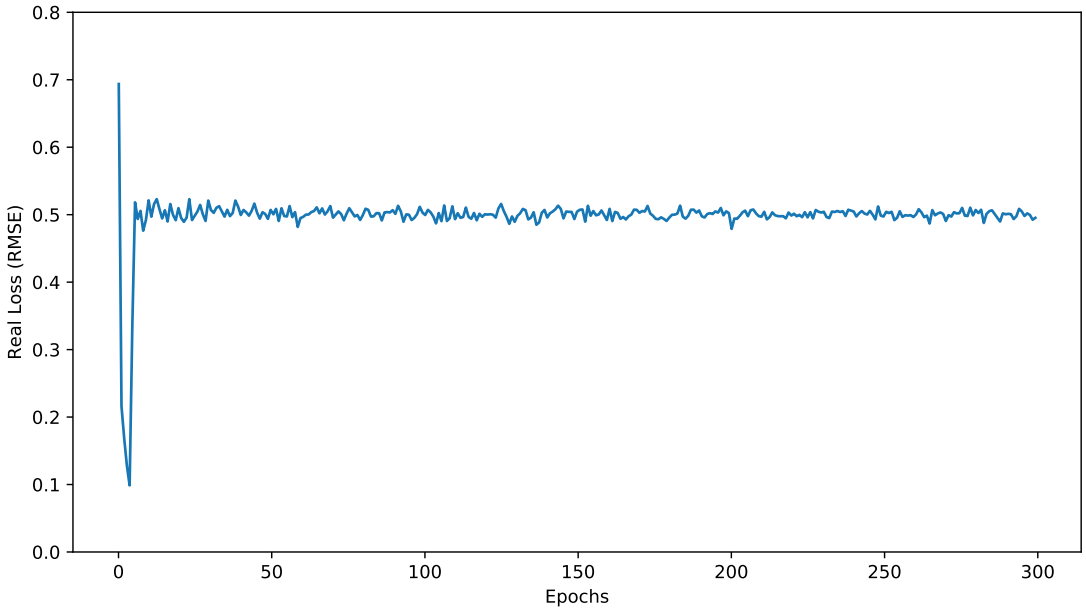

Figure 6.1: Adversarial loss for real face images (first term of $\mathcal{L}_D$ in Eqn. (18)).

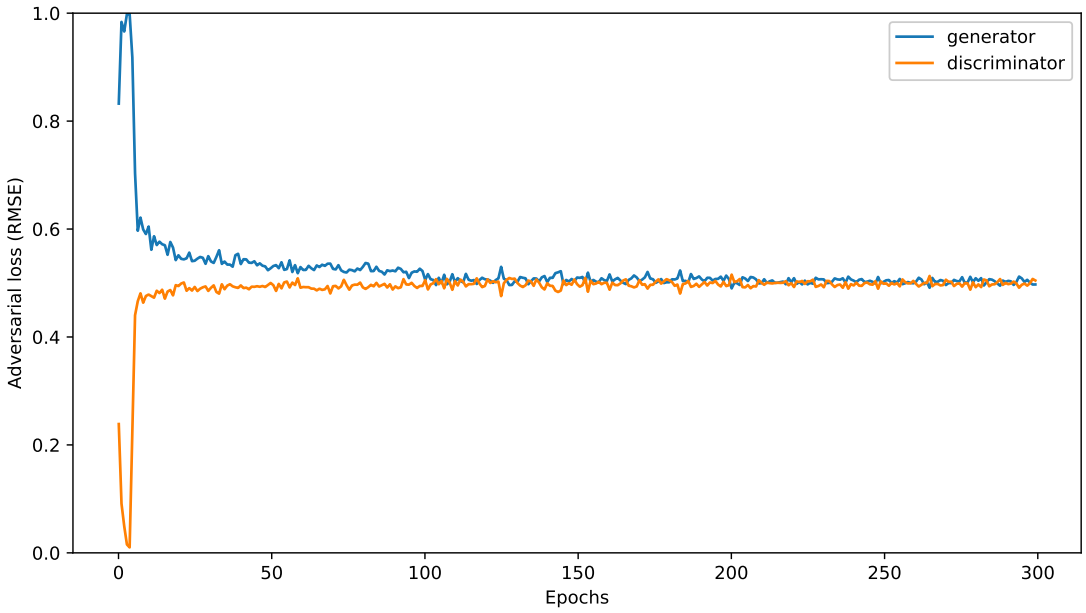

Figure 6.2: Adversarial loss for fake generated face images ($\mathcal{L}_G$ and second term of $\mathcal{L}_D$ in Eqn. (18)).

6.1.2 Identity-Preserved Loss

As shown in Figure 6.3, the identity-preserved losses of training and testing samples drop and keep similar during the whole training, which indicates that the trained generator could preserve the identity of the input face images from both trainset and testset without obvious overfitting.

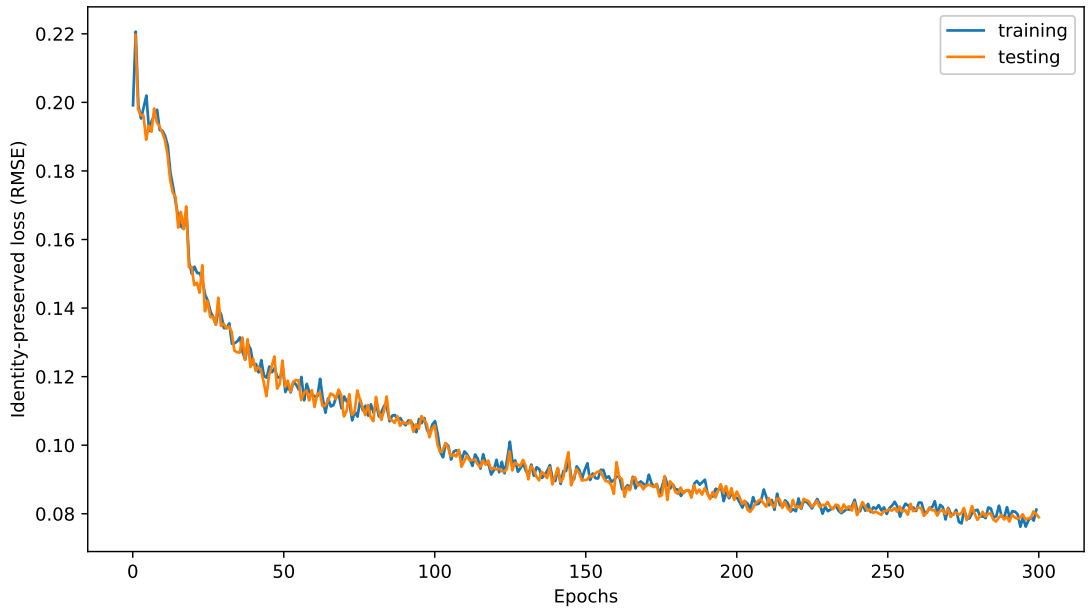

Figure 6.3: Identity-preserved loss for generated face images ($\mathcal{L}_{identity}$ in Eqn. (18)).

	Training	**Testing**
RMSE	0.0793	0.0790

Table 6.1: Identity-preserved loss.

6.1.3 Beauty Loss

Similar to identity-preserved loss, the beauty losses of training and testing samples are almost equal during the whole training process as shown in Figure 6.4. This means the generator could synthesize the input face images according to the input condition and produce faces correctly labeled by the FBP.

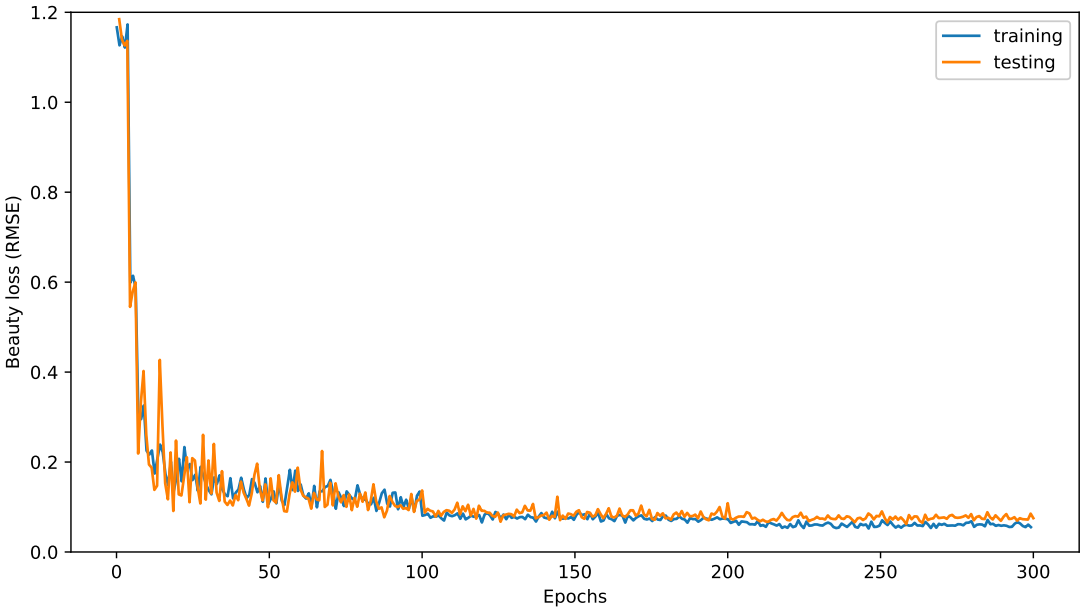

Figure 6.4: Beauty loss for generated face images ($\mathcal{L}_{beauty}$ in Eqn. (18)).

	Training	**Testing**
RMSE	0.0585	0.0752

Table 6.2: Beauty loss.

6.1.4 Self-consistency Loss

As shown in Figure 6.5, the self-consistency loss drops during training which means the generator learns to duplicate the input face images when the input condition is the ground truth beauty score of the input image. Compared with identity-preserved loss, the value of self-consistency loss is much higher, which is because the two losses are measured differently. The identity-preserved loss is calculated with the features of FRN, where the self-consistency loss is measured by raw pixel value.

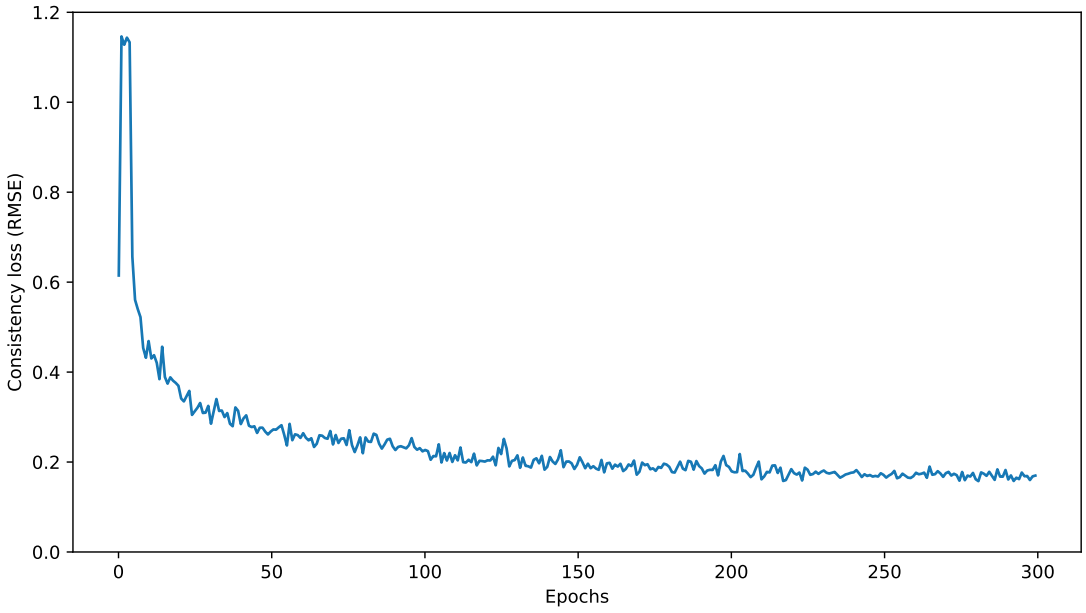

Figure 6.5: Self-consistency loss for the generator ($\mathcal{L}_{consist}$ in Eqn. (18)).

	Training	**Testing**
RMSE	0.1789	-

Table 6.3: Self-consistency loss.

6.2 Generated Faces

As shown in Figure 6.6, we select five face images with different beauty scores from the testset and input them to the generator. Each face is mapped to five target beauty scores. From the result, we could observe that low score faces tend to have darker skins and more noises compared with high score faces. High score faces usually have clearer facial contours and heavier makeup. The basic features of the face, such as the shape of face and the position of eyes and nose, etc., are kept unchanged to preserve the identity of the original face.

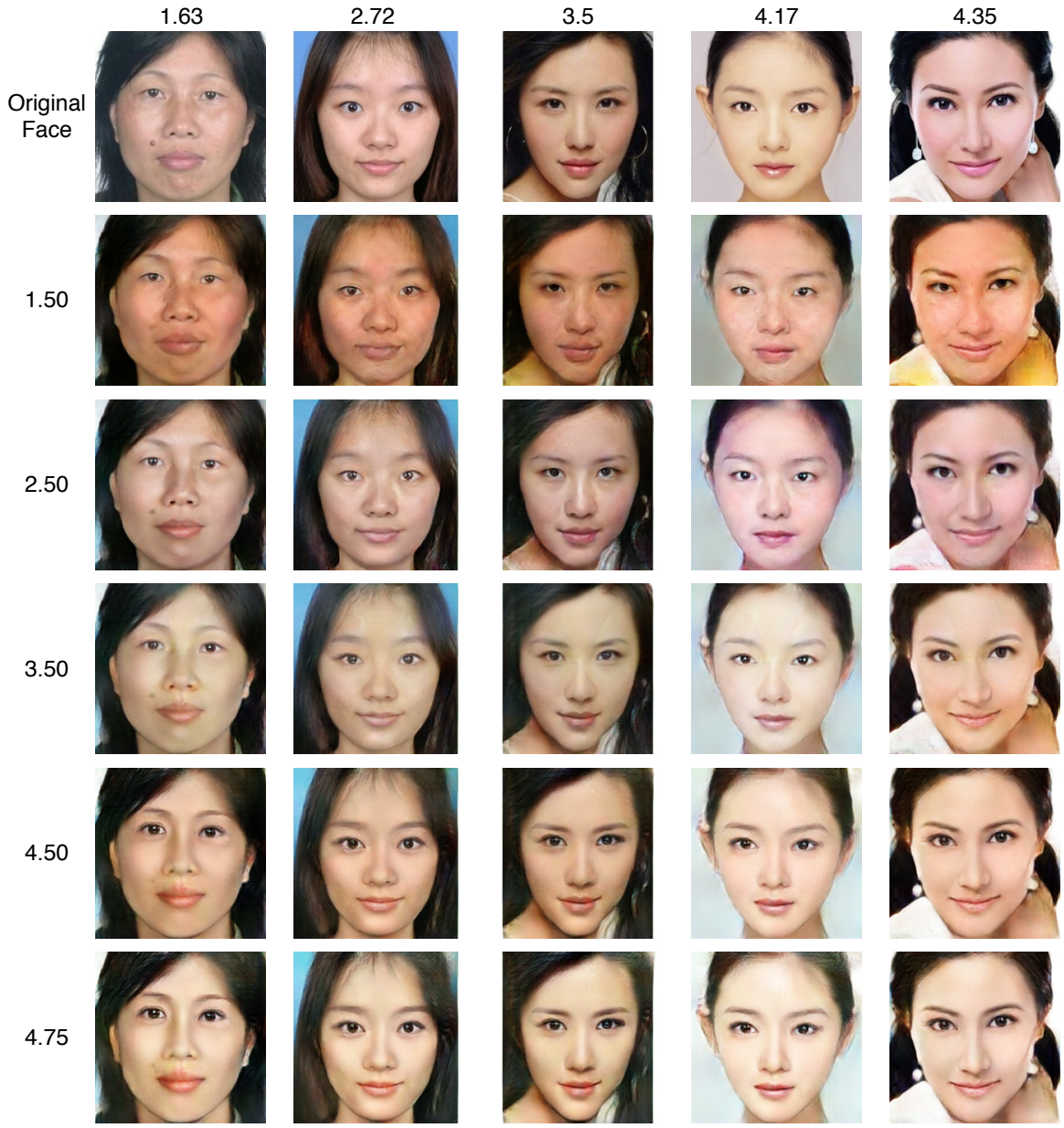

Figure 6.6: Generated faces. The first row shows the original face images and each row below shows the generated faces with the target beauty score indicated on the left. The numbers above the original faces show the original ground truth beauty score.

As the generator could receive continuous real-valued input condition, we display generated faces with smaller gaps to show how the faces gradually evolve with the beauty score in Figure 6.7.

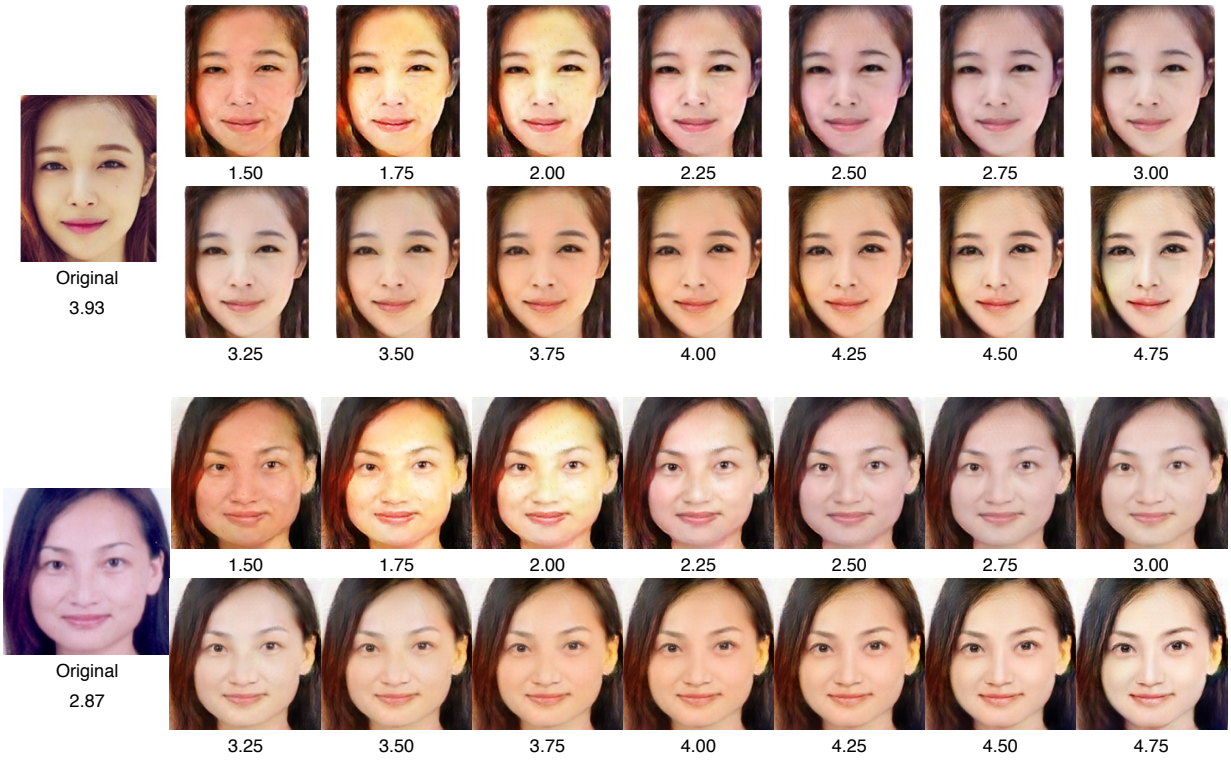

Figure 6.7: Generated faces with smaller gaps.

6.3 Comparison & Analysis

In this section, we will inspect the details of the generated faces and compare the distinctions between faces with different beauty scores. Because some differences in facial features could be very tiny when the beauty score is close, we choose the generated faces with big gaps in beauty score to show the differences more clearly.

6.3.1 Facial Features & Makeup

In our experiments, we observe that the relative positions of eyes, nose, and mouth are kept almost unchanged in the generated faces from the same input face, which is probably the reason why the FRN cannot distinguish them. The differences in beauty mainly reflect in makeup, texture, and illumination. As shown in Figure 6.8, the faces with higher beauty score look smoother than the faces with lower beauty score. Obvious makeup could be observed on the eyes (eyebrows and eyelashes) and mouths. Furthermore, the illumination

also plays an important role. For example, the light in the eyes makes them look clear and shining, where the highlight on the bridge of nose enhances the nose contour and looks more three-dimensional.

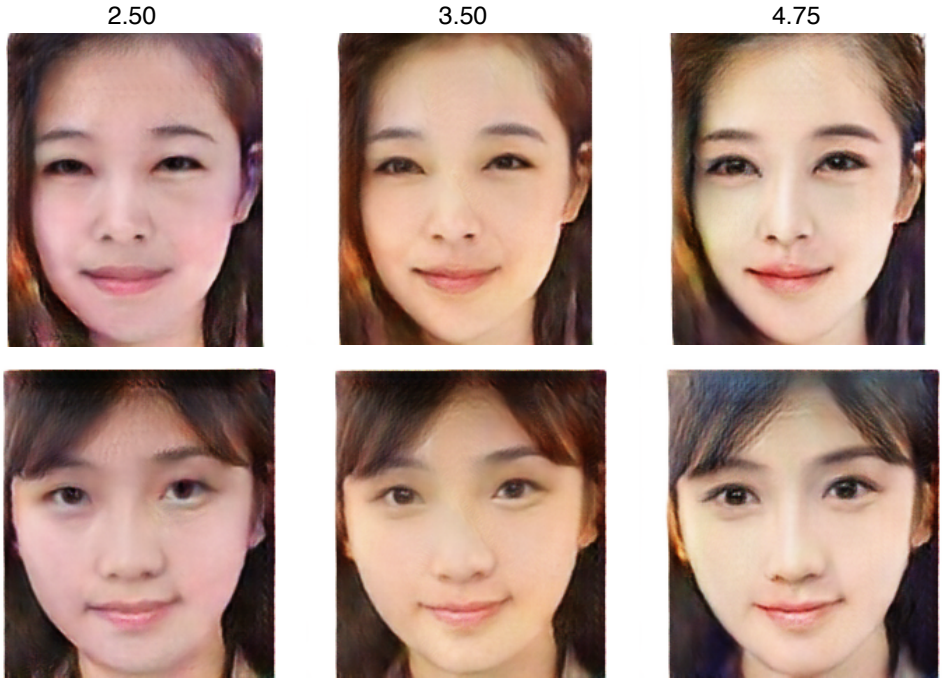

Figure 6.8: Generated faces with large gaps of beauty score. Each row is generated from the same input face.

6.3.2 Lying Silkworm

"Lying silkworm" is a special name for the eye fat below the lower eyelid. It is usually confused with eye bag which is caused by the skin under the lower eyelid. More specifically, lying silkworm is a narrow band of fat with a height of 4-7 mm, where eye bag is an inverted triangle shape with a height of 2-3 cm. As shown in Figure 6.9, the main difference between lying silkworm and eye bag is the shape and height. Compared with eye bag which looks old and listless, lying silkworm is considered to be a symbol of young, innocent, and intimate in many Asian countries such as China and South Korea.

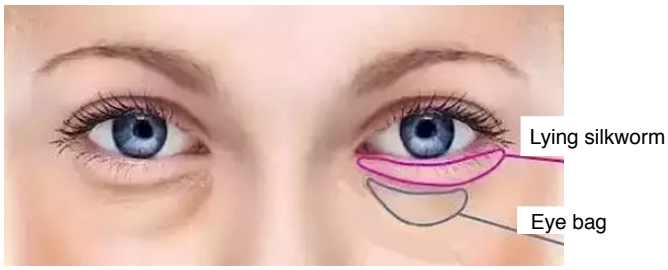

Figure 6.9: Lying silkworm. Lying silkworm is a narrow band of fat with a height of 4-7 mm, where eye bag is an inverted triangle shape with a height of 2-3 cm. Image source: http://finance.cz001.com.cn/2019-06/13/content_3620455.htm.

In our experiments, we find that the generator is able to catch the features of eye bag and lying silkworm. As shown in Figure 6.10, the faces with beauty score of 2.50 have obvious eye bags which are eliminated in the faces with beauty score of 3.50, while lying silkworms could be observed in the faces with beauty score of 4.75. Therefore, we could conclude that the feature of lying silkworm is considered as a symbol of beauty by the generator, while eye bag represents the feature of faces with lower beauty score.

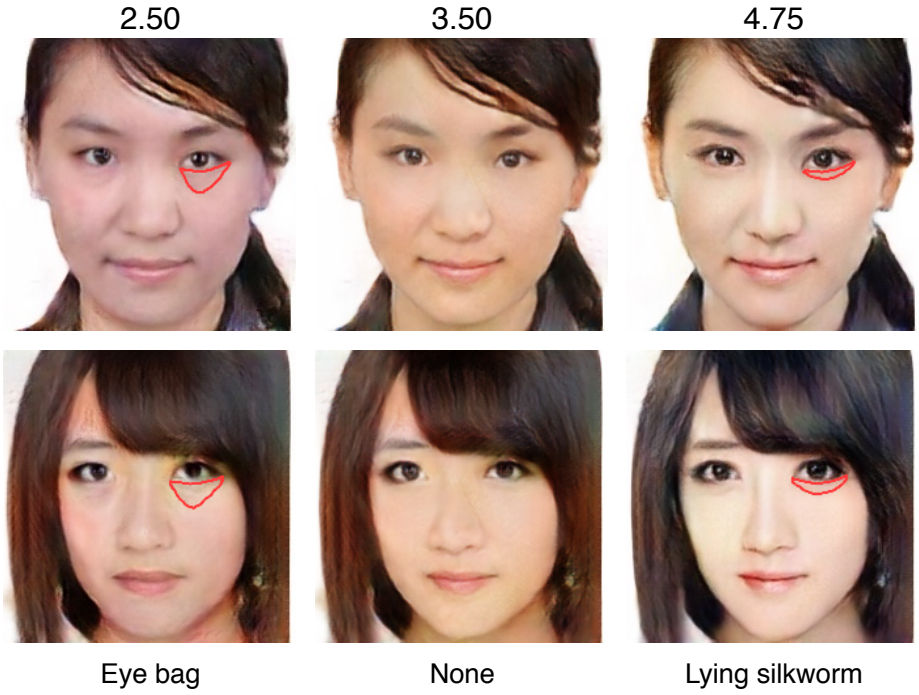

Figure 6.10: Generated faces with eye bags and lying silkworms. The red curves indicate the region of eye bags and lying silkworms.

6.3.3 Identity-Preserve Analysis

To further understand how the identity of the input face is preserved, we inspect the relation between the identity feature and the input face image. We design an experiment to quantify the influence of each pixel of the face image on the identity feature. Recall that in Section 4.5, the 5th convolutional layers of FRN is chosen to be the identity feature:

$$h(x) = FRN(x)_{conv5} \tag{19}$$

where $h(.)$ is the identity feature. Then the influence of each pixel is defined as the gradient of the identity feature w.r.t. this pixel:

$$Inf(x_{i,j}) = \frac{\partial \sum h(x)}{\partial x_{i,j}} \tag{20}$$

where $x_{i,j}$ is a single pixel in the input face image. From the result illustrated in Figure 6.11, we could observe that the pixels around the facial landmarks tend to have a stronger influence on the identity feature. Therefore, we could conclude that the identity of a face image is mainly determined by the pixels around the facial landmarks. To preserve the identity of the input face, the generator should maintain the positions of facial landmarks, i.e., the face contours and positions of eyes, nose, and mouth.

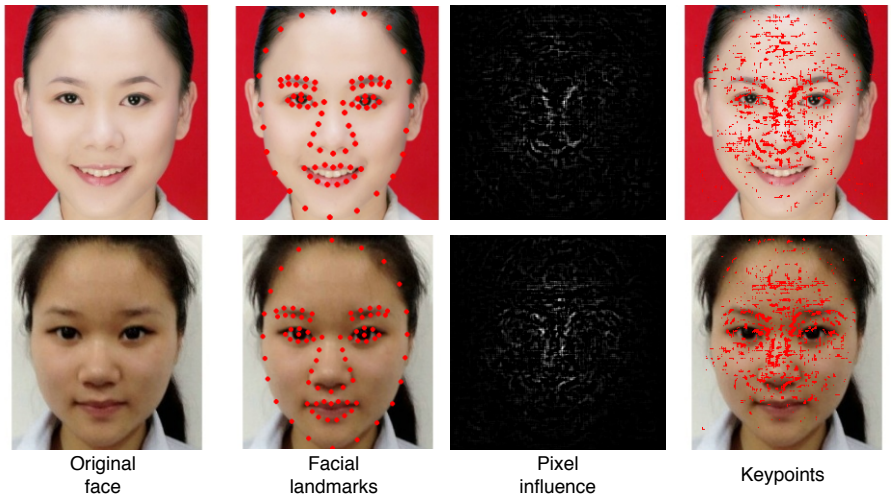

Figure 6.11: The 1st column presents the original input faces. The 2nd column presents the facial landmarks. The 3rd column presents the influence of each pixel (whiter means higher influence). The 4th column shows the top 5% keypoints with the strongest influence.

Chapter 7

Conclusions and Future Research

In this work, we propose a framework which consists of cGANs, FBP, and FRN for identity-preserved face beauty transformation. The trained generator could transform the beauty scale of the input image to the target scale indicated by the input condition of real-valued beauty score. To train such a framework, we introduce conditioned instance normalization and self-consistency loss. The result shows that certain features of the input face image are edited, including makeup, skin texture, illuminance, and other facial features. With regard to the ethical issues related to face beauty, we claim that this work is purely a research on face image generation and the bias which may occur in the experiments merely reflects the preferences of the raters of the dataset.

In future works, we consider three aspects:

(1) Artifacts could be observed in some generated images, especially in those images with uncommon occlusions or illuminance. A similar problem also occurs when the generator is required to produce face images with very low or high beauty scores. We believe the problems are mainly caused by the lack of data and the Gaussian-like distribution of the beauty score. In future works, we could enlarge our dataset by labeling other datasets with pretrained FBP as in [20]. Alternatively, we could improve our work when another large face beauty dataset is available or create our own dataset.

(2) In this work, only Asian female faces are considered. In future works, we should include

more faces with different races and both genders. However, annotators may have different standards of beauty towards different races and genders. There could also be significant bias, e.g. the average scores of female faces may be larger than those of male faces, which may cause the generator to produce face images of one specific gender or race. To address the problem, we could encode the race and gender information in the generator, so that face images of different races and genders could be treated differently.

(3) Through the experimental results, we observe that the color tones and contrast styles of transformed faces with close beauty scores are similar, even when the identity of original input face is different. Additionally, the facial features and makeup also appear to be almost the same. This means the generator synthesizes all the input images in a similar way regardless of their identities and original facial features. Therefore, in future works, we could research on instance-level face beauty transformation where the generator could edit the input face intelligently according to its original features. For example, some face images are labeled as low scores mainly because of their poor illuminance. To beautify such face images, we may focus on optimizing the illuminance to make the images look clearer instead of editing the facial features. Another possibility is that the generator could figure out the most appropriate makeup for different face images.

References

[1] C. LI *et al.*, "Triple generative adversarial nets," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4088–4098.

[2] J. Xu *et al.*, *A new humanlike facial attractiveness predictor with cascaded fine-tuning deep learning model*, 2015. arXiv: 1511.02465 [cs.CV].

[3] L. Lin *et al.*, "R2-resnext: A resnext-based regression model with relative ranking for facial beauty prediction," in *Proc. 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, pp. 85–90.

[4] S. Shi *et al.*, "Improving facial attractiveness prediction via co-attention learning," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 4045–4049.

[5] L. Xu *et al.*, "Hierarchical multi-task network for race, gender and facial attractiveness recognition," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 3861–3865.

[6] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[7] L. A. Gatys *et al.*, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2414–2423.

[8] D. Ulyanov *et al.*, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4105–4113.

[9] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1501–1510.

[10] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[11] Y. Choi *et al.*, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 8789–8797.

[12] J.-Y. Zhu *et al.*, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2223–2232.

[13] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, 2014. arXiv: 1411.1784 [cs.LG].

[14] Z. Wang *et al.*, "Face aging with identity-preserved conditional generative adversarial networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 7939–7947.

[15] G. Antipov *et al.*, "Face aging with conditional generative adversarial networks," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2089–2093.

[16] V. Dumoulin *et al.*, *A learned representation for artistic style*, 2016. arXiv: 1610.07629 [cs.CV].

[17] N. Diamant *et al.*, "Beholder-gan: Generation and beautification of facial images with conditioning on their beauty level," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 739–743.

[18] T. Karras *et al.*, *Progressive growing of gans for improved quality, stability, and variation*, 2017. arXiv: 1710.10196 [cs.NE].

[19] L. Liang *et al.*, "Scut-fbp5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction," in *Proc. 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1598–1603.

[20] T. Li *et al.*, "Beautygan: Instance-level facial makeup transfer with deep generative adversarial network," in *Proc. ACM international conference on Multimedia*, Oct. 2018, pp. 645–653.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] X. Mao *et al.*, "Least squares generative adversarial networks," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2813–2821.

[23] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning (ICML)*, vol. 37, Jul. 2015, pp. 448–456.

[26] O. M. Parkhi *et al.*, "Deep face recognition," in *Proceedings of the British Machine Vision Conference (BMVC)*, Sep. 2015, pp. 41.1–41.12.

[27] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 901–909.

[28] M. Arjovsky *et al.*, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, Aug. 2017, pp. 214–223.

Appendix A

Proof: Inefficiency of concatenation in cGANs

In this section, we will prove that concatenation, which is used in normal cGANs to combine the input image and condition, is inefficient in a convolutional structure because it only works as a bias term. The basic operation of concatenation can be found in Section 4.6.

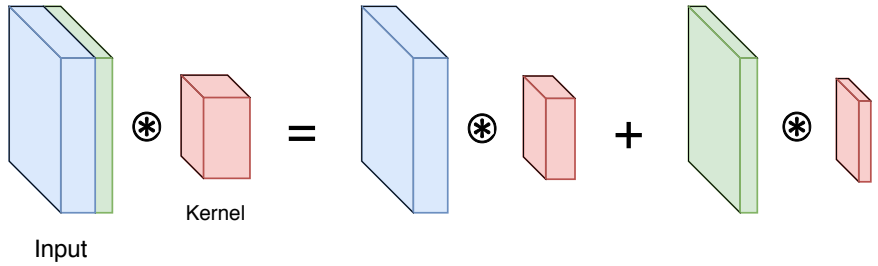

Figure A.1: Convolution of concatenated feature could be decomposed into convolution of input image and convolution of input condition.

As shown in Figure A.1, the concatenated input image and condition convolves with the kernel in a convolutional layer (only one kernel is considered for simplicity). Because of the linear property of convolution, the operation could be decomposed into two convolutions:

$$X_{concat} \circledast K = X_{image} \circledast K_1 + X_{cond} \circledast K_2 \tag{21}$$

where X_{image} is the input image of size $h \times w \times c$. X_{cond} is the reshaped input condition of size $h \times w \times 1$ which has the same height and width with the input image. X_{concat} is the concatenated input of size $h \times w \times c+1$. K is the kernel in convolutional layer of size $k \times k \times c+1$. K_1 and K_2 are the decomposed kernels of size $k \times k \times c$ and $k \times k \times 1$, respectively.

After the decomposition, the first term in Eqn. (21) ($X_{image} \circledast K_1$) would become the convolution of the input image in a normal convolutional layer. Then the second term ($X_{cond} \circledast K_2$) is the effect of the input condition added to the original result. Recall that X_{cond} is reshaped from a 1-d real-valued number, therefore, X_{cond} is an $h \times w \times 1$ tensor filled with a single value. The convolution can be written as:

$$
\begin{aligned}
Y_{cond} &= X_{cond} \circledast K_2 \\
y_{cond}^{ij} &= \sum_{m=-\frac{k}{2}}^{\frac{k}{2}} \sum_{n=\frac{k}{2}}^{\frac{k}{2}} x_{cond}^{i+m,j+n} \cdot k_2^{mn} \\
&= x \cdot (\sum_m \sum_m k_2^{mm})
\end{aligned}
\tag{22}
$$

where Y_{cond} is the convolution result and y_{cond}^{ij} is the i^{th} row, j^{th} col element of Y_{cond}. x_{cond}^{ij} and k_2^{ij} are the elements of X_{cond} and K_2, respectively.

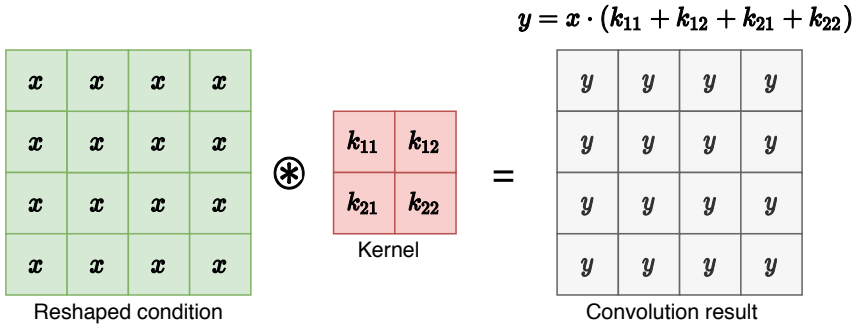

Figure A.2: Convolution of reshaped condition. x is the input real-valued condition. k_{ij} is the element of the kernel. The output of the convolution is an $h \times w \times 1$ tensor full of the value y.

As shown in Figure A.2, the convolution output of the input condition is a tensor of size $h \times w \times 1$ filled with the same value. Adding this tensor to the convolution output of

the input image is actually adding a bias term. Specifically, the input condition merely serves as a bias term in cGANs when the input image and condition are combined through concatenation.