FPT UNIVERSITY - HOA LAC CAMPUS
DEPARTMENT OF COMPUTER SCIENCE
GRADUATION THESIS FINAL REPORT

HIGH FIDELITY FACE SWAPPING USING GENERATIVE ADVERSARIAL NETWORK

Nguyen Tien Manh

A thesis submitted in part fulfillment of
the degree of BSc. in Computer Science with
the supervision and moderation of Dr. **Do Thai Giang**

Hoa Lac High-tech Park, Ha Noi
April 27, 2021

Abstract

Many current face swapping works are achieving state-of-the-art results with realistic face-looking performance. However, the developer of these works does not completely open-source the code or release only the inference section of the code, making it challenging for the community to replicate the results as shown in these articles.

In this work, we going through the process of developing recent advances in using Generative Adversarial Network to solve face manipulation problems. To this end, we studied a novel framework of high fidelity face swapping method, namely FaceShifter.Different from earlier works that only a limited amount of information captured from the target image was used to to synthesize swapped face, our framework is capable of producing high fidelity swapped faces by attentionally utilizing and integrating the target face information into the output image.

Based on previous works, we develop a novel multi-level face attributes encoder to efficiently exploit attribute latent vector representation. Besides, we propose a new generator approach, by dynamically incorporate information between identity and attribute features using novel carefully built Adaptive Attentional Denormalization (AAD) layers for robust image generation.

Various experiments on a wide range of wild faces datasets indicate that our framework synthesis is not only considering more realistic and aesthetically pleasing but also involving network structure generality, architecture scalability as well as more stable in maintaining identity's features in comparison to other state-of-the-art techniques. By utilizing image-to-image translation across domains, our methods could be trained without subject-specific annotations, facilitate the network for further research and applications.

We also publish the code used for training and testing so that it can be used by anyone for research purpose and open-source communities.

Acknowledgement

I would like to express the deep and sinecre gratitude to the Computer Science Department at FPT University and VinAI Research Institute. Without the their supporting and facilities this thesis would not have been possible.

I am also grateful to all of whom I have had the pleasure to collaborating on this and other related projects. I would extremely thankful to my supervisor, Dr. **Do Thai Giang**, for his kindness help and expertise in developing the research problem and methodology. Your insightful feedback pushed me to improved my thought and leverage my work to a higher level.

Most importantly, I want to express my deepest appreciation to my parents, whose love and support are with me in whatever I pursue.

Contents

List of Figures

CHAPTER 1

Introduction

"I have always been convinced that the only way to get artificial intelligence to work is to do the computation in a way similar to the human brain. That is the goal I have been pursuing. We are making progress, though we still have lots to learn about how the brain actually works."

Geoffrey Hinton (AI Scientist, 2018 Turing Award Winner)

1.1 Introduction

Recently, in the social media and digital world, we often encounter warnings about the reliability of information that we exposed to. In the earlier years, it is thought to be impossible to trick somebody with visual content since they appear to be apparently unordered, unrealistic and contain irregular shapes. Given the brilliant human visual system, a person could easily distinguish between the actual image and the generated image while viewing a photo on the Internet.
However, as we move to the age of deep learning and big-data driven world, the visual information are now getting easier to trick the viewer. Along with the advent of deep learning in the field of computer vision in recent years, the processing of digital images, especially human portraits, has progressed rapidly and, in most cases, could produced photorealistic results. One of the prominent generated content exist online

is called DeepFake[40], which could change the source identity of a person in a video given a target face image. DeepFake application include not only usage in entertainment field but also a malicious usage to create pornographic or unwanted content.

Face swapping is a common method of creating false content that involves replace a target face with a source face while preserving the target's facial attribute and identity information. The development of face swapping (or identity swapping) works have been devoted to studied the challenging problem of how to change one source face to target face which keep facial attribute thoroughly and consistent. With application span from entertainment purpose[1, 43] to security approach[4], face swapping has gained much attention in the research community.

Early method utilize the replacement of using segmantic information of inner region or ROIs between two faces[3, 42], create a discrepancy between source and target poses, perspective, etc.... Many works apply 3D Face model to estimate face representation either by calculated[39] or determined in advance[33], thus require specific segmentation map dataset. However, the results of 3D face modelling is still not convincing and lack of flexibility due to large variation in face appearances and postures.

Recently, GANs (Generative Adversarial Networks)[13] is the driving force behind the progression of face synthesis and manipulation task. GANs not only able to achieve more realistic result, but also have a clearer pipeline compare to other classical method. Face synthesis results created by recent state-of-the-art works [9, 20, 19] are now becoming increasingly realistic and completely indistinguishable to the human vision system.
Latest advance in image-to-image translation task often involving Conditional GANs (cGANs) [29], which was found to be generalized to different domain effectively [18].

The aim of this work is to focus on the fidelity of face swapping method. Specifically, to get more perceptually appealing results, the synthesized swapped face should be seamlessly blended into the target image with stable quality and follow the target scene's lighting conditions. We believe that the simple basic alpha or Poisson mixing is not capable of focus attentionally and could lead to information loss during generation.

1.2 Related Works

In computer vision and graphics science community, face switching is still an active research field with many works and gain community attention. In earlier works[4, 5], simple approach using region-mapping or inner modify was proposed when two face has an identical poses. Latest works consisted of two main focus direction which use different approach to achieve more and more realistic and high-fidelity generated faces.

1.2.1 3D-Based Methods

One of the earliest face switching methods require manual interaction to switch the face[4]. This approach however does not handle the changes in face expression. The process is then automated and develop further by adding a offline face library with common coordinate system[3]. Olszewski et al.[35] proposed a hybrid model using multi-linear PCA to acquire the 3D geometry and a single texture of the target face, while realistic per-frame texture deformations was generated using a deep generative network. More recently, to best capture the face attributes with high precision, many works exploit the use of 3D morphable face model (3DMM)[5] by fitting it to the faces and make the transfer task in 3D space to smoothing the variation[21, 39]. In 2017, Nirkin et al.[33] show 3D face shape estimation is unnecessary for realistic face swapping and can be replace by a segmentation mask to mapping the face to high latent space. These 3D Face Model estimation technique require a large amount of specific 3D data, making further investigation challenging for researcher. Moreover, training 3D model involve the distribution of the data need to be fixed, which could lead to information loss and implausible results.

1.2.2 GAN-Based Methods

GANs[13] were shown to be able to learn robust deep features without the need of extensively annotated training data. CycleGAN [46] proposed a method of employ a multi-scale cGAN architecture as well as include a perceptual loss to maintain the pixel-mapping between the desired output and the generated output. GANimation[37] proposed a dual conditional GAN[29] conditioned on Action Units annotation, allow the network control the magnitude of facial movement base on an attention map. By separate hair and facial region, RSGAN[31] independently handles face and hair appearances in the latent spaces, swapping face in the latent-space and reconstruct the entire face with the representation. FSGAN[32] similarly separate hair and face representation and use Poisson blending loss to fit the source face into target images. However, these methods often failed in the case of various lighting conditions, challenging expressions or extreme pose in the target face. Moreover, splitting the hair and face feature require an additional segmentation neural network, thus increase the overall network complexity and lack of generality when face is occluded.

1.3 Main contribution

In this works, we demonstrate a generative adversarial network based approach to generate high fidelity face swapping using images. Unlike previous approaches, our work is specifically design to be subject agnostic: it could run swapping on two different faces without retraining the model. Furthermore, our works also include a novel end-to-end pipeline which can be use to train the network that allows for easy for customizing and modifying architecture.
Specifically, we make the following contributions:

- Dynamically integration identity and facial attributes: Our work can achieve high-fidelity looking faces by attentionally combine identity and attribute embedding, thus making the model robust to various poses and expression.

- Subject agnostic swapping: Our works can overcome the process of pair-specific training when run network inference. This result in scalable network with easy usage and can be trained on multiple dataset without pair annotations.

With extensive validation report, our methods are both qualitative and quantitative significantly better than current state of the art network.

1.4 Outline

In thesis, we address the problem of high fidelity face swapping, specifically:

In **Chapter 1**, we give an gentle introduction and related works about high fidelity face swapping.

In **Chapter 2**, we give some background knowledge about neural network, coming from feed forward neural network to gradient-based optimisation and generative adversarial neural network.

In **Chapter 3**, we demonstrated the effectiveness of generative adversarial neural network in the task of face image manipulation, we then show that with GAN we are able to generate realistic face image.

In **Chapter 4**, we studied a novel framework named FaceShifter, a state of the art method in face swapping field.

In **Chapter 5**, we showed some experiment results between different methods on different face datasets.

In **Chapter 6**, we concluded the thesis and then make some future work in the subject.

CHAPTER 2

Artificial Neural Network

2.1 Introduction

Scientist have long questioned how can machine acquire the human's brain ability to get and process complex information to come up with precise decision and human-level cognition. By copy the mother nature architecture of the brain itself, we can stimulate the brain's work including processing and transferring information between neurons. Artificial Neural Network (ANNs) take brain structure as inspiration to building networks composed of many artificial neurons, small elements that perform very simple operations of their inputs.

2.2 Neural network

2.2.1 Feed forward neural network

Feed forward neural network

The feedforward neural network was the first and most basic artificial neural network to be developed. Follow [12], we define feedforward neural network as a function approximation that learn a mapping $y = (x; \theta)$ to best approximate the desired output. Through the training process, the network gradually learn parameter θ that result in the best function approximation. Neural network commonly composed of many layers subsequently connected one to another, information flow from the first layer to second

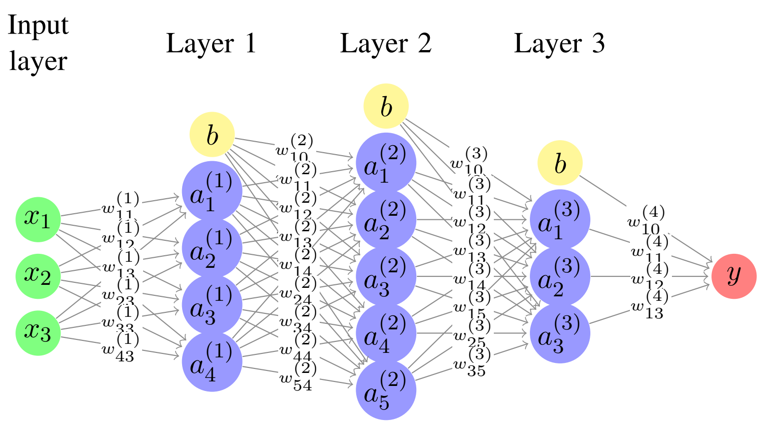


Figure 2.1: *A MultiLayer Perceptron. The sum and the non-linearity nodes have been omitted for the sake of clarity.*

layer,... to output(final) layer. When the network does not extract output from the intermediate layers, they are called hidden layers. The term feedforward neural network also used interchanged with multilayer perceptions (MLPs).

For each layer, given the n-dimensional input $x = (x_1, x_2, ..., x_n)$, the output computed as:

$$y = f(x) = \sigma(\sum_{i=1}^{n}(w_i \cdot x_i) + b) \tag{2.1}$$

Where σ is an activation function(for example ReLU [30], Sigmoid, Tanh...) and the layer stored a learnable weight $w_1, w_2...w_n$, along with a bias b. All the equation are calculate using matrix multiplication, giving the neural network flexibility to compute and optimise. When define multi-layer perception, we can apply same principle but the notion is slightly different:

$$a^{(l)} = \sigma(z^{(l)}), z^{(l)} = W^{(l)} \cdot a^{(l-1)} \tag{2.2}$$

Gradient-based learning

Neural networks are often trained by using gradient-based optimizers, iteratively minimize the loss function so that the network can learn respectively. Normally, modern neural networks are trained using either negative log-likelihood, which equivalent to the cross-entropy error , or the mean square error between the training data and the model distribution:

$$E_{mse} = \frac{1}{M}\sum_{D}\frac{1}{2}||y - \hat{y}||_2 \tag{2.3}$$

The back-propagation algorithm [see, e.g., 38], which allows the information flow backward from the cost function through the network, was the most native way to compute the gradient. To gradually approach the global optima, the target is optimizing each layer's weight by a factor proportional to the cost $(C = \hat{y} - y)$ and to the input (x):

$$w' = w^L - \frac{\partial C}{\partial w^L} \tag{2.4}$$

For example, when using MSE cost function, we can compute the fraction of cost that contribute to each layer's weight by taking the derivative of the cost with respect to it:

$$\begin{aligned}
\frac{\partial E_{mse}}{\partial w_{ij}^{(l)}} &= \frac{\partial}{\partial w_{ij}^{(l)}} \left(\frac{1}{m} \sum_{\mathcal{D}} \left[\frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|_2 \right] \right), \\
&= \frac{1}{m} \sum_{\mathcal{D}} \left[\frac{1}{2} \frac{\partial}{\partial w_{ij}^{(l)}} \left[(\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \right] \right], \\
&= \frac{1}{m} \sum_{\mathcal{D}} \left[(\mathbf{y} - \hat{\mathbf{y}})^T \frac{\partial(-\hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} \right].
\end{aligned} \tag{2.5}$$

By utilize the chain rule of derivation, we can compute the partial derivative $\frac{\partial \hat{y}}{\partial w_{ij}^{(l)}}$, for example one weight of the third layer $w_{ij}^{(3)}$:

$$\begin{aligned}
-\frac{\partial \hat{\mathbf{y}}}{\partial w_{ij}^{(3)}} &= -\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z}^{(4)}} \cdot \frac{\partial \mathbf{z}^{(4)}}{\partial w_{ij}^{(3)}}, \\
&= -\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z}^{(4)}} \cdot \frac{\partial \mathbf{z}^{(4)}}{\partial \mathbf{a}^{(3)}} \cdot \frac{\partial \mathbf{a}^{(3)}}{\partial w_{ij}^{(3)}}, \\
&= -\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z}^{(4)}} \cdot \frac{\partial \mathbf{z}^{(4)}}{\partial \mathbf{a}^{(3)}} \cdot \frac{\partial \mathbf{a}^{(3)}}{\partial \mathbf{z}^{(3)}} \cdot \frac{\partial \mathbf{z}^{(3)}}{\partial w_{ij}^{(3)}}
\end{aligned} \tag{2.6}$$

Consider $\hat{\mathbf{y}}$ is equal to $\mathbf{a}^{(4)}$ in this example, we can have the final equation:

$$-\frac{\partial \hat{\mathbf{y}}}{\partial w_{ij}^{(3)}} = -\frac{\partial \mathbf{a}^{(4)}}{\partial \mathbf{z}^{(4)}} \cdot \mathbf{W}^{(4)} \cdot \frac{\partial \mathbf{a}^{(3)}}{\partial \mathbf{z}^{(3)}} \cdot [a_j^{(1)}]_i \tag{2.7}$$

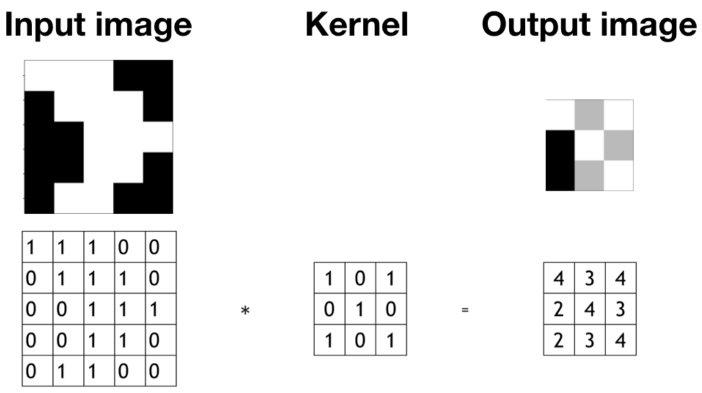
2.2.2 Convolutional neural network

Without a doubt, human brain has exposed to vision system in a way as too strong and workable. Human vision system has long play a crucial part in strengthen the ability to capture and process image data smoothly and precisely. So, how do we cope with image information with the neural networks ? The most prominent solution is convolutional neural network, which share the same neuron structure as neural network. The CNN still basically consisted of many neural layer that take some inputs, perform a matrix multiplication and optionally follows it with a non-linearity. The network also exploit the gradient-base learning and cost function to optimize the output. What make

CNN different and special is the fact that input data to each neuron is an image, make possible by take advance of the spatial structure in the data. Specifically, each layer in convolutional neural network consisted of many matrix act as a kernel (filter) sliding across the image data to gain information.

Convolutional Layer

Convolutional layer(conv layer) are special layer in CNNs. Each layer perform a convolution (or sometime refer as cross-relation in signal processing) function to the input.



$$\begin{bmatrix}1&1&1&0&0\\0&1&1&1&0\\0&0&1&1&1\\0&0&1&1&0\\0&1&1&0&0\end{bmatrix} * \begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix} = \begin{bmatrix}4&3&4\\2&4&3\\2&3&4\end{bmatrix}$$

Figure 2.2: *Operation of a convolution calculation in convolutional layer.*

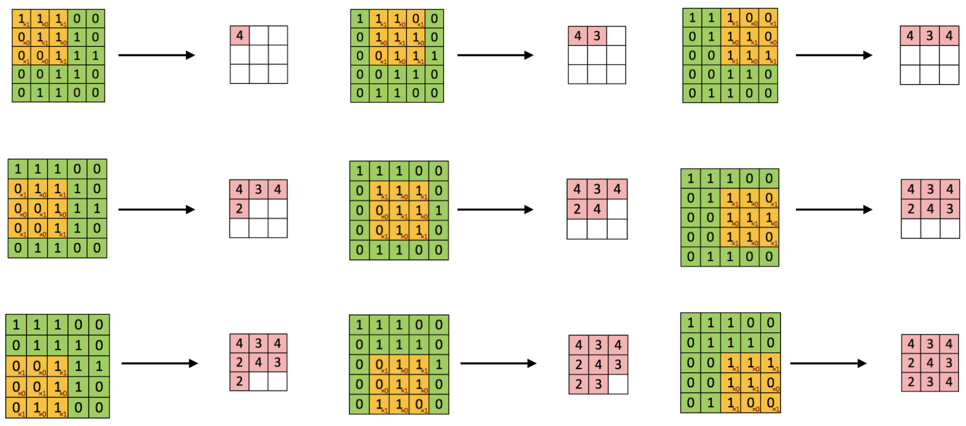


Figure 2.3: *Convolution step by step (illustration in 2D plane).*

The filter moves to the right with a certain stride value till it parses the complete width. Moving on, it continue start over at the beginning (left) of the plane and repeats

the process until the entire image is traversed.
Normally, except conv layer, a CNNs usually contain some pooling layers and fully connected layers(plain neural network) at the end of the network to achieve the highest accuracy. A convolutional neural network is able to efficiently captivate the spatial and temporal dependencies in an image through the implication of multiple kernels. Moreover, the convolutional network architecture achieve a better fitting to the image dataset because the reduction in the number of parameters involved and reusability of weights. With the advent of Alexnet[23] and Resnet[14], which both has achieve a significant increase in accuracy of the Imagenet data competition at the time, convolutional neuralwork has recently has been boosted to a new level which come to near-human level of object detection and recognition.

2.3 Generative Adversarial Network

2.3.1 Generative Model

Recent state of the art result in deep learning mostly came out from discriminative model, that is, the model learn by differentiate extracted features to classify or regression the output. On the other hand, the development of generative model has stagnant for many year and has not yet reach the human-level until the advent of GAN[13]. Most recent generative model like StarGAN[8] or ProGAN[19] has achieve realistic-level of image generation and become indistinguishable to human vision system. For clarity, we describe generative model as the learning of the distribution of the training data and generalise the output driven by this model. In other words, generative models are type of model that learn to generate output data similar to the training data.

2.3.2 Encoder-Decoder Network(ED Network)

An ED network is a compact of two separated networks, a Encoder E and a Decoder D, specifically design so that the output of the network is used to approximate the input data itself: $D(E(x)) = x_g$. Encoder composed a list of narrower layers, target of extract the compressed knowledge representation that contain most important information of input. Decoder, conversely contain upsampling layer or unpooling layer that enlarge the extracted embedding to map the content of the original image. Normally, if the encoder and decoder are symmetrical and the network is trained with the objective $D(E(x)) = x$, then the network is called autoencoder. An autoencoder's first part or encoder could sometimes be used separately to exploit the compact feature information of the input data.

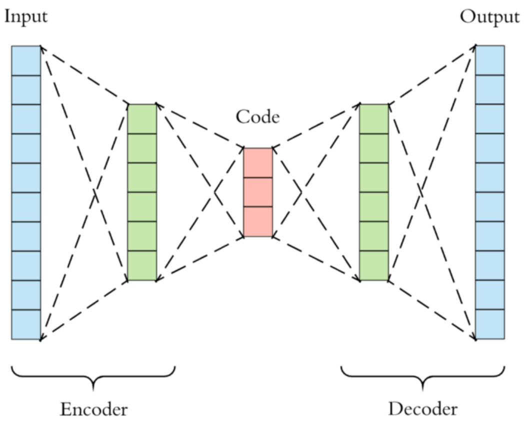


Figure 2.4: *Architecture of a conventional Autoencoder.*

However, vanilla autoencoder require corresponding input need to mapping with the decoder's output and therefore not capable of generating similar output with little variability. Variational autoencoder is a special version of autoencoder, which allow the encoder learn the posterior distribution of the decoder by maximizing a lower bound on the log likelihood of the input data.

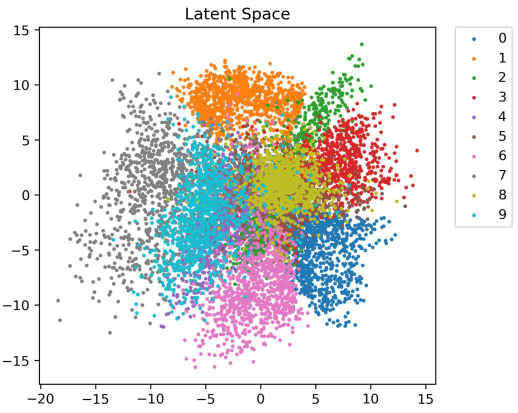


Figure 2.5: *Latent space visualization of an Adversarial Autoencoder[27], which behave similarly to Variational Autoconder.*

2.3.3 Generative Adversarial Network

In a paper published in 2014[13], Ian GoodFellow has introduced the architecture of generative adversarial neural network, which basically composed of two separated parts: a generator G and a discriminator D. The goal of generator G is to learn how to generate reasonable sample that belong to the training dataset's distribution. Conversely, the discriminator will differentiate between the training dataset samples and produced data, which acted as a negative samples for the classification. Based on the generated data, the discriminator will punish the generator if the result is unconvincing or unlikely to come from the training data. The generator's job is to fool the discriminator such that it can not distinguish generated image and real image. The generator and discriminator is then iteratively training in an adversarial approach. This means D and G involve in a two-player minimax game, the generator try to minimize when the discriminator try to maximize the expected error between two distribution with target to the value function $V(G, D)$:

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(x)))] \tag{2.8}$$

By this way of training, we can mapping the generator's distribution and discriminator's distribution close together, first by force the discriminator learn to distinguished the synthetic features between realistic ones by gradient-based optimization. Then, the generator is forced to learn to generate an image which capture the distribution or the region the D has learn. This process is iteratively repeat many times until G capture the distribution of the data and D are unable to differentiate between the two distribution.

Algorithm 1 Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments. The algorithms is taken and modified from [13]

```
1: for the number of iterations do
2:     for k steps do
           • Sample minibatch of $m$ noise samples $\{z^{(1)}, z^{(2)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
           • Sample minibatch of $m$ examples $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
           • Update the discriminator by ascending its stochastic gradient:
             $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \Big[ \log D\big(x^{(i)}\big) + \log\big(1 - D\big(G\big(z^{(i)}\big)\big)\big) \Big] \tag{2.9}$$
3:     end for
           • Sample minibatch of $m$ noise samples $\{z^{(1)}, z^{(2)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
           • Update the generator by descending its stochastic gradient:
             $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \Big[ \log\big(1 - D\big(G\big(z^{(i)}\big)\big)\big) \Big] \tag{2.10}$$
```

The evolution of GAN has lead to an exponential research on understand how GAN work and achieve better results. The adversarial training in specific has drive an enormous amount of work and application. Some relevant task included: Image-Super Resolution, Anime Character Recognition, Deblurring, Image Synthesis, ...

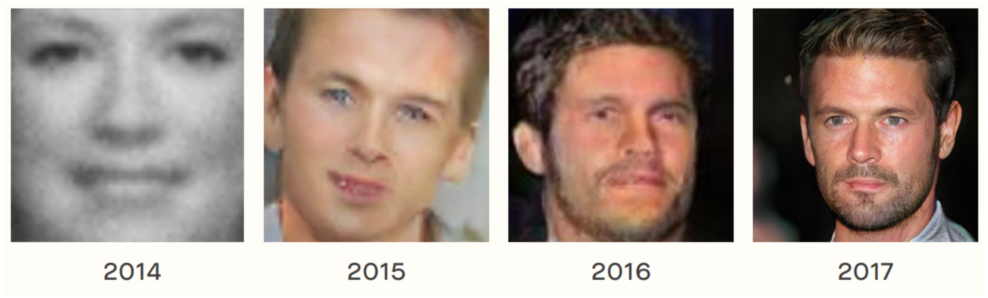


Figure 2.6: *Evolution of GAN in generating real-looking face images. Figure taken from [6].*

However, adversarial training is very difficult approach in reality due to 2 network's convergence. Overtime, when the generator improve the result of generating images, the discriminator become less contribute to the optimisation and gradually give false advice to the generator (perfect discriminator has 50% accuracy). The generator is not capable of learning based on junk feedbacks, and fail to converge. This sometime has been called model collapse and may indicate in the vanish gradient of the discriminator. In 2017, a paper has introduced Wasserstein Loss[2] to solve the problem of intractable gradient of the discriminator. Nonetheless, recent advance in improvement of GAN mostly come from distribution of training sample and compact of many neural network, rather than improve the adversarial training methods. Therefore, for a adversarial network approach, training often lead not to complete convergence but rather than unstable, fleeting with a lot of fluctuation in the cost function.

CHAPTER 3

Generative Adversarial Network (GAN) for Facial Manipulation

A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.

– The Turing Test,
Alan Turing

3.1 Introduction

Follow [40, 28], we define facial manipulation as a task composed of four different categories based on the level of manipulation. We provide a description of each of them as below:

- Face Synthesis: The creation of entire non-existing face image, which recently has achieved tremendous success through the practice of GAN.
- Face Swapping(Identity Swap): Consist of the task replace a face of one person by another person. We will give a clearer definition and interpret the usage of GAN to do swapping in the next chapter.
- Face Editing: Refer as the task of modify facial attributes like the hair, gender, age, etc.

- Face Reenactment: A Face Reenactment technique is where a source face image is used to drive the expression, mouth, gaze, pose, or body of a target face image.

Some of the earliest research paper in generative models has included face systhesis field as a demonstration for the effective of model[22, 13], this also raise an immense interest for researcher in this field. With the development of GAN-based network, many face image datasets [20, 7, 26, 19] has been published to leverage the performance of face image generation. Recently, numerous spoof images generated by GAN-based methods has gained huge attention in the community with exceptional reality and indistinguished from the real face images[40]. Even so, generative neural network still acted like a black box and researcher has not yet fully understand how face attributes are featurized and realistic face are generated until the introduction of StarGAN [8]. By specifically redesign the network with the additional classification branch for the discriminator and a classification penalty in target function, StarGAN has capability of generating realistic face in multiple domain according to face attributes(hair, expression, pose, gaze,...). This has open up to many application which use StarGAN as their baseline and then training on their own dataset to achieve the desired output.

However, when it come to the field of other facial manipulation: face replacement, face editing and face reenactment, face attributes play a significant role to make the face look realistic. This is essentially difficult since the face attribute is not clearly seen and 2D face modelling is lack of generality. These pose some inevitable challenge and require either specific design of the network or content-specific cost function. In this section, we try investigate some of the recent advance in image generation when using GAN that have been found effective in face image generation topic.

3.2 Batch Normalization

In a paper wrote in 2015[17], the authors has showed that we could facilitate the training process of neural network considerably by attach a normalization layer before the activation layer(i.e ReLU). The batch normalization layer(BN) will exploit the feature statistic to compute the output of normalization. The BN layer is originally designed to alleviate the issue of internal covariate shifting - a common problem while training a deep neural network. It first standardizes each feature in a mini-batch, and then learns a common slope and bias for each mini-batch. Not only discriminative models was benefit from BN layers to become converge-quicker, recent generative model also utilize BN layers to exploit image feature more effectively[44].
Given the input batch $x \in^{N \times C \times H \times W}$, the output of BN layer will be computed based on the feature statistics(input's batch mean and standard deviation). In specific, we could defined $\gamma, \beta \in \mathbb{R}^C$ are the slope and bias parameter which the network study

from the training process:

$$BN(x) = \gamma(\frac{x - \mu(x)}{\sigma(x)}) + \beta \tag{3.1}$$

where $\gamma, \beta \in \mathbb{R}^C$ is the mean and standard deviation and bias parameters that was calculated from the training data. The feature statistics ($\mu(x) \in \mathbb{R}^C$ denoted for the mean, $\sigma(x) \in \mathbb{R}^C$denoted for standard deviation) will be computed along the mini-batch set of the training input, across the spatial space. We could then formulate the statistic calculation by applying sum computing with regard to each channel:

$$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{nchw} \tag{3.2}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (x_{nchw} - \mu_c(x))^2 + \epsilon} \tag{3.3}$$

The BN layers offer a guarantee that the input distribution of each layer remains unchanged across different mini-batches. When training using Stochastic Gradient Descent (SGD) optimization, the network could learn a stable input distribution and thus help model converge quicker. The network inference by store the global statistics of all training samples, which is used to normalize every mini-batch of test data.

However, in practice, the use of BN layers encounter problem of discrepancy between training and inference due to different setting of mean and standard deviation statistics during each stage. The author of [16] address this issue by create a gradient optimizer to optimize statistics parameter approach the popular statistic during training. As researcher extend the field utilize BN layers, Lou et al. [25] found that by utilizing destination domain's mean and standard deviation, BN layer could adapt to the domain changes smoothly when running.

3.3 Adaptive Instance Normalization

3.3.1 Instance Normalization

In 2017, when improving the feed-forward stylization methods, Ulyanov et al.[41] has achieve a significant boost in output image just by modify the network to use Instance Normalization instead of Batch Normalization. Surprisingly, IN layers share the same equation with BN:

$$IN(x) = \gamma(\frac{x - \mu(x)}{\sigma(x)}) + \beta \tag{3.4}$$

What make IN layer different is how the $\mu(x)$ and $\sigma(x)$ parameter are calculated. For each sample and each channel, the spatial dimension is computed separately to generalize the contrast content given in the input. This will be kept consistent throughout the process of training and testing.

$$\mu_n c(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \tag{3.5}$$

$$\sigma_n c(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw} - \mu_n c(x))^2 + \epsilon} \tag{3.6}$$

3.3.2 Adaptive Instance Normalization

With an external style image, Dumoulin et al[11] introduce conditional instance normalization technique, which learn a different slope γ^s and bias β^s for each style s. The network is able of generating images with a variation of style just by modify the affine parameters in the CIN layers. The author of [15] has an assumption of originally Instance Normalization layer could act like a style-normalization by normalizing feature statistics, therefore we could adaptively learn the affine transformation parameters in the layer. In that paper, the author also introduce Adaptive Instance Normalization (AdaIn), which eliminate the use of fixed affine parameters and dynamically study affine transformation from the style images. Specifically, given the content image x and a style image y, the AdaIn layer compute the output as:

$$AdaIn(x, y) = \sigma(y)(\frac{x - \mu(x)}{\sigma(x)}) + \mu(y) \tag{3.7}$$

In other words, the AdaIN layer facilitate the style image into the content image by a slope $\sigma(y)$ and a bias $\mu(y)$, which will be computed along the spatial space and channel-wise matrix. This obviously add no extra cost of computation to the original IN layer, making AdaIn stand out as an efficient of training style-base model architecture.

3.4 StyleGAN

Earlier advance work in GAN are contributed firstly by the development of the discriminator. From the experience of enhance many discriminative model, we could increase the complexity then therefore the classification result of the discriminator, researcher believe generator will adapt significantly and generate more realistic images.

As such, the generator has been stagnant and still be a black box that is very difficult to modify or capture changes in visual result. For example, some random input used in synthetic images generation are perform better than others, while latent space give us no clue but remain a immense challenge to understand.

In 2018, StyleGAN[20] has been proposed a new approach of how we define generator. Traditional GANs encounter a problem of controlling styles and features within the same images due to feature entanglement. It does not allow for control over finer

styling of the image because the model is driven by its own distribution, as governed by its training with high-level attributes, and also because it gets influenced by the general "trend" of its dataset. For example, training dataset has the problem of entanglement between the color of the hair and age, that is the data only include image of senior with white hair and young people with black hair. The GANs model will eventually "fixed" the distribution of hair and age, leading to changing one result in both experience changes. The Style-GAN could alleviate these by utilize a separate mapping network f, such that input will go through several fully-connected layers before go to the generator. Input go through this network will be regarded as intermediate latent vector. And since the mapping network could perform variation of affine transformation on the input, the generator will not be suffered from any unchangeable distribution. Moreover, this network mapping f also facilitate the latent space, make the output implicitly learn a normalization feature.
By using the intermediate latent space, the user could modify input embedding with slightly changes so that the final output vector will not changes significantly. In other words, the StyleGAN architecture will enforce the linear transformation, making the produced images more realistic.

3.5 Spatial Adaptive Denormalization (SPADE)

3.5.1 Conditional Normalization layers

Both Conditional Normalization layer [11] and Adaptive Normalization layer [15] have share the same idea of first normalize the activation layer output to have the zero mean and unit standard deviation. This process is operate across each spatial location separately to generalize and best capture the input distribution. Then the activation layer encounter a second stage, where normalize features are denormalize back to learn an affine transformation by store affine parameter from the external data. Because of affine parameter and normalization was computed uniformly across all spatial position, the content of input and style images can not be leverage and result in low performance in content-specific area.

3.5.2 Spatial Adaptive Denormalization

To address the problem of uniform spatial computation, Taesung el at.[36] has proposed a normalization layer with spatial-varying affine transformation, which could leverage information from the segmentic masks by modulating normalized activations.

Using the same notation as in(3.2), the activation value at site $(n \in N, c \in C^i, y \in H^i, x \in W^i)$ can be computed as:

$$SPADE(x,y) = \mu^i_{c,y,x}(m)\frac{h^i_{n,c,y,x} - \mu^i_c}{\sigma^i_c} + \beta^i_{c,y,x}(m) \tag{3.8}$$

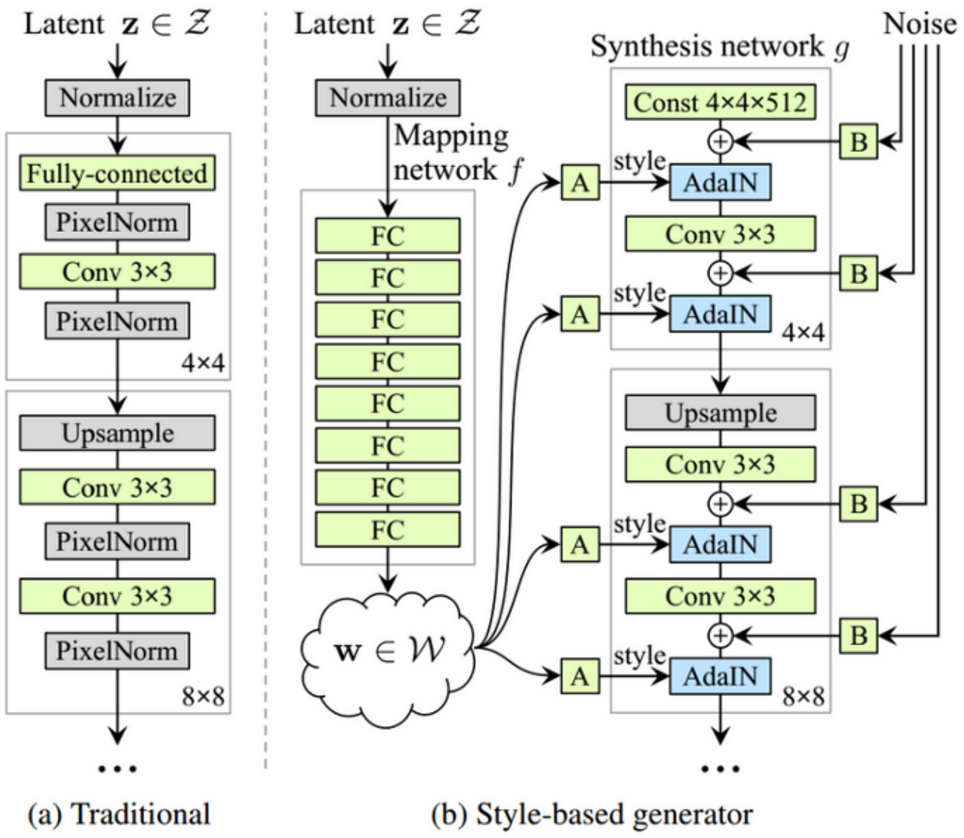


Figure 3.1: *Compare between original GAN and StyleGAN architecture. Figure taken from [20].*

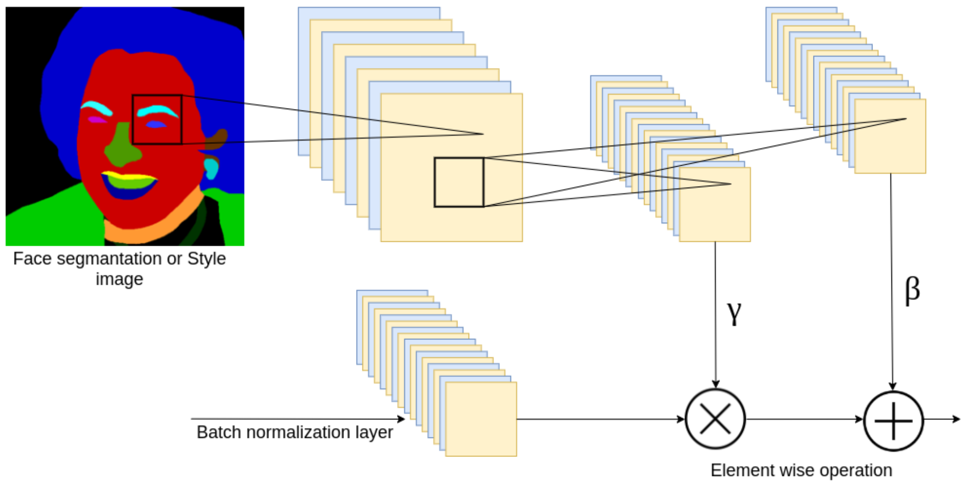


Figure 3.2: *Spatial Adaptive Denormalization Layer.*

Here the $h_{n,c,y,x}$ is stand for activation before normalization, the mean μ_c^i and standard deviation σ_c^i is calculated base on spatial information with regard to the channel c:

$$\mu_c^i = \frac{1}{NH^iW^i} \sum_{n,y,x} h^i_{n,c,y,x} \tag{3.9}$$

$$\sigma_c^i = \sqrt{\frac{1}{NH^iW^i} \sum_{n,y,x} \left((h^i_{n,c,y,x})^2 - (\mu_c^i)^2 \right)} \tag{3.10}$$

The key variable used in equation (3.8), namely $\mu^i_{c,y,x}(m)$ and $\beta^i_{c,y,x}(m)$ was learnt by two separate convolutional neural network, increasing the flexibility of the overall generalization layer. Each parameter network utilize two convolutional layers with input is the segmantic masks, the output then was element-wise add and multiply to the original batchnorm features. We argued that these variable act like a mask, attentionally focus on style-specific location to adapt with variety of different spatial information area. Indeed, as suggested in the paper, SPADE is a generalization version of BN and AdaIN layer. By change the segmentation map m to class annotation map and set the modulation parameters to constant, we could achieve a type of Conditional BatchNorm layer[11]. Changing setting m by a style image, spatially-invariant parameter and modify $N = 1$ bring us the AdaIN layer[15].
Therefore, SPADE has capability of achieving great result with more lightweight network and more realistic images. In the next chapter, we study face swapping architecture that utilize normalization as a core functional activation layer in order to generate high fidelity face images.

3.6 CycleGAN

3.6.1 Network Architecture

Follow [46], we define image-to-image translation as a class consist of computer vision and graphical problems where the target is to learn a transformation between a pair of image set using sample training data consisted of aligned image pairs. However, the number of pair dataset is still quite limited and require specific finetune to adapt with new changes. In 2017, CycleGAN has been introduce to solve the problem of unpaired image-to-image translation.
The CycleGAN is an extension version of the GAN architecture that consist the contemporary training of two generator models and two discriminator models. The two generators have adverse direction, one convert input image from domain A to B and other from B back to A. Two discriminators try to distinguish the A-generated image with A domain images and B-generated image with B domain images.

3.6.2 Objective function

The target of CycleGAN is to learn a best mapping or transformation between two domains X and Y. Considering a sample $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_i\}_{j=1}^M$ where

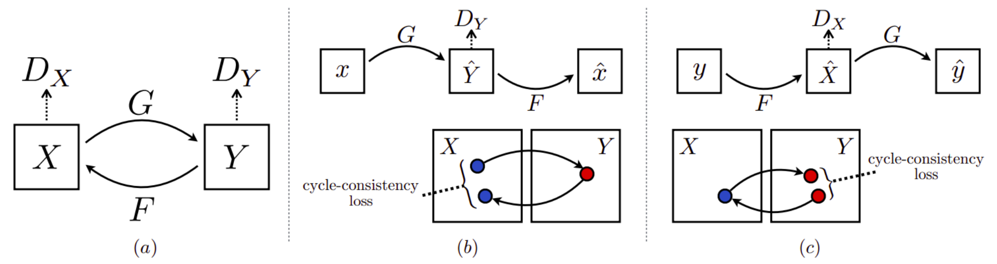


Figure 3.3: *CycleGAN architecture. Figure taken from [20]*

$y_i \in Y$, the network include two mappings: $G : X \rightarrow Y$ and $F : Y \rightarrow X$, along with two discriminator D_X and D_Y, where D_X try to classify an image ($\{x\}$ and $\{F(y)\}$) come from which distribution, D_Y similarly differentiate between images $\{y\}$ and translated image $\{G(x)\}$.

Adversarial loss

The adversarial loss for mapping function from domain X to Y is defined as:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y\sim p_{data}(y)}\left[\log D_Y(y)\right] + \mathbb{E}_{x\sim p_{data}(x)}\left[\log 1 - D_Y(G(X))\right] \tag{3.11}$$

Cycle consistency loss

The learn-mapping function should be consistent, such as: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. The network enforce the consistent the by adding a cycle consistency loss:

$$\mathcal{L}_{CYC}(G, F) = \mathbb{E}_{x\sim p_{data}(x)}\left[\|F(G(x)) - x\|_1\right] + \mathbb{E}_{y\sim p_{data}(y)}\left[\|G(F(y)) - y\|_1\right] \tag{3.12}$$

Full objective

The full objective to train an unpaired image-to-image translation is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \mathcal{L}_{CYC}(G, F) \tag{3.13}$$

In previous works, the absence of paired examples is always the limitation for researcher to apply neural network technique into image transformation tasks. The advent of CycleGAN has bring us a lot of opportunity to study the application of GANs in unpaired image-to-image translation or cross-domain generation. Although not mentioned in the paper, the framework of CycleGAN can be used for identity swap easily by training on a training datasets consisted of unpaired source face and target face images with some modification. We will detail the approach using a variation of CycleGAN in the next chapter.

CHAPTER 4

High Fidelity Face Swapping

4.1 Introduction

According to [32], we commonly define face swapping as the task of transferring a face from source image to target image. Specifically, the output face image must be matched with pose, attribute,... with target face image and have realistic look features, that is indistinguishable from the real face. Earlier methods relied on 3D Face Modelling (3DMM)[5] to control and direct the attribute of the output face to match this objective. However, 3D face models either require specific data preparation or unable to leverage target attribute like occlusions, lighting conditions or image styles. With the advancement of neural network and especially GAN and its variations, recent methods focus on the use ED network to extract latent space of face image and combine face identity features with face attribute feature to drive the target face to match the output image. The approach come from one id to another(retrained for the target image), many to one , and many to many or subject agnostic architecture.

4.2 FaceShifter

4.2.1 Introduction

In this section, we introduce FaceShifter[24], a novel methods for face swapping. The methods input require two face images in the same domain: X_s and X_t(source and target image respectively). The input source image then go to an Adaptive Embedding

Integration Network, composed of to an encoder-decoder style. First, the source image pass through an identity encoder, a state of the art face recognition model named ArcFace[10] to get the identity feature. Meanwhile the target face image is going through an multi-level attributes encoder, contain U-Net like structure. Then, the attributes feature are compact with identity feature through an adaptive attentional denormalization generator to generate the output image. The output face image should contain the identity from source face and attributes from target images.

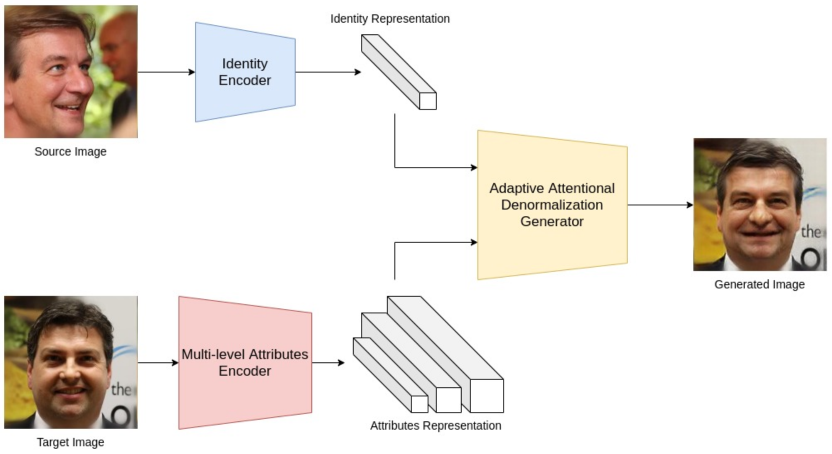

Source Image
Identity Encoder
Identity Representation
Adaptive Attentional Denormalization Generator
Generated Image
Target Image
Multi-level Attributes Encoder
Attributes Representation

Figure 4.1: *Overall Face Swapping framework.*

4.2.2 Identity Encoder

State of the art face recognition system require massive training data and accuracy to leverage the recognition up to millions people. Therefore, the traditional softmax loss function is incapable of optimise the feature embedding to enforce higher similarity for intra-class sample and diversity for inter-class sample. Arcface[10] has proposed an Additive Angular Margin Loss in replace to softmax loss to improve the discriminative capability of the feature embedding. The loss add a margin penalty m between feature embedding and weight to increase the intra-class compactness and inter-class discrepancy.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(s\cos(\theta_{y_i}+m))}{\exp(s\cos(\theta_{y_i}+m))+\sum_{j=1,j\neq y_i}^{n}\exp(s\cos\theta_j)} \tag{4.1}$$

Notably, ArcFace has surpass recent face recognition methods and achieved highest results in relative benchmarks. In this face swapping method, we use pretrained model Arcface from [10], which set embedding feature size to 512. In [24], it is believed that training face recognition model on large quantity of subject images could result in better representation that 3D-based model like 3DMM[5].

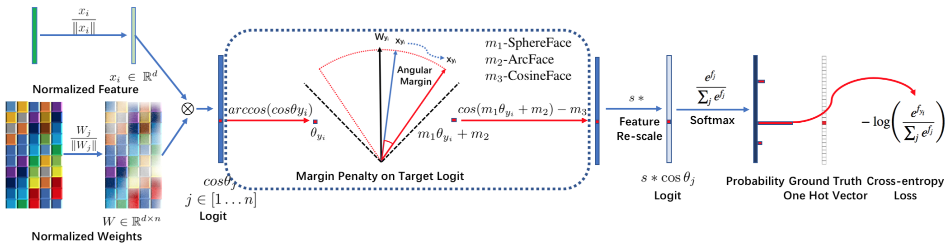


Figure 4.2: *Additive Angular Margin architecture. Figure taken from [10]*

4.2.3 Multi-level attributes encoder

Unlike identity embedding, face attributes contain a lot of variation features and more general representation for the facial information. Face attributes also contributed by non-consistent properties and could have very large entangle latent space and therefore identify by a single embedding is quite a challenge. To preserve detail of facial attribute, the feature need to capture more spatial information at various scale. FaceShifter consider face attributes as a multi-level feature maps, each represent a level of complexity. Specifically, we design a U-Net like structure that take input a image and output a list of n feature embeddings, where $z^k_{att}(X_t)$ corresponding to the k-level attribute feature map from the UNet decoder:

$$z_{att}(X_t) = \{z^1_{att}(X_t), z^2_{att}(X_t), ..., z^n_{att}(X_t)\} \tag{4.2}$$

The attributes embedding network does not require any external notations, since it can learn the embedding that well reflect the facial attribute through effective self-supervise training approach.

4.2.4 Adaptive Embedding Denormalization Network

When we have extracted identity embedding and facial attribute embeddings, how we corporate these two embeddings to generate a swapped face $\hat{Y}_{s,t}$ play a critical role in the desired output. Previous works simply concatenate these embeddings and often encounter a problem where the output image is blurry or contain a lot of GAN artifacts. FaceShifter integrate such embeddings by using a so-called adaptive attentional denormalization network, which composed of different adaptive attentional denormalization layer (ADA layer). The layer utilize a mechanism of SPADE, facilitate feature denormalization technique to dynamically combine features in different levels.

Specifically, let h^k_{in} represent the input activation map of an AAD layer, which has shape of $C^k \times H^k \times W^k$, with C^k denote the number of channels and H^k and W^k is the height and width of the tensor respectively. Follow SPADE, the input first was normalized:

$$\bar{h}^k = \frac{h^k_{in} - \mu^k}{\sigma^k} \tag{4.3}$$

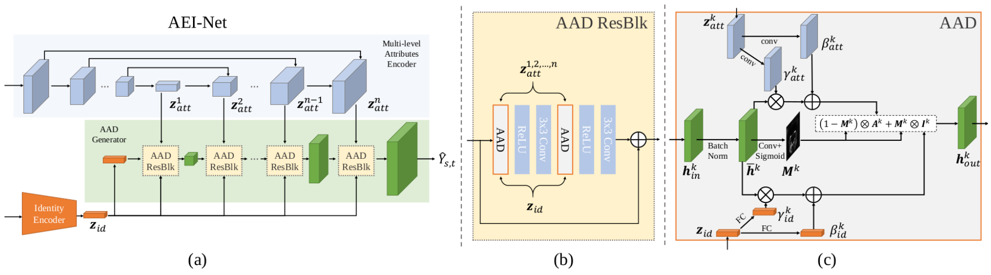


Figure 4.3: *AEI Net architecture. Figure taken from [24].*

The mean μ^k and standard deviation σ^k was computed channel-wise using a popular statistic in the N samples mini-batch input:

$$\mu_c^k(h_{in}) = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} h_{nchw} \tag{4.4}$$

$$\sigma_c^k(h_{in}) = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(h_{nchw} - \mu_c^k(h_{in}))^2} \tag{4.5}$$

The $\bar{h^k}$ then follow three different branch to integrate facial attribute, identity embedding into attentional mask to produce targeted output. The structure of these branches has been best demonstrated in Figure 4.1c.

The identity embedding (z_{id}^k) will be integrated by computing an identity activation I^k. The identity activation was computed similar to the mechanism of SPADE, through the process of denormalizing $\bar{h^k}$ with regard to the identity embedding. Two modulation parameter use in this process is generated through fully connected layers on the input identity vector. The attribute embeddings level k (z_{att}^k) will be integrated in a similar way by an attribute activation A^k. However, the denormalization parameters should be computed using two convolutional layers, since the input attribute feature map is a 3D tensor with height H^k, width W^k and C^k channels.

$$\begin{aligned} I^k &= \gamma_{id}^k \otimes \bar{h^k} + \beta_{id}^k, \\ A^k &= \gamma_{att}^k \otimes \bar{h^k} + \beta_{att}^k \end{aligned} \tag{4.6}$$

To leverage more spatial information from the source images, AAD layer create a mask filter with weight form a mechanism of attention, adaptively select different parts of identity vector and attribute vectors. These two embeddings could generally contribute to multiple location of the synthesized face, with identity embedding contribute more on the most discriminative identity recognition features like mouths and face landmarks, while facial attribute embeddings should play a critical role in generating poses, mouth state, expression,..etc. Therefore, the layer generate an attentional mask M^k by convolving h^k through a series of convolutional layer and a sigmoid activation.

The output of the AAD layer could be formulated as the element-wise multiplication of anttentional mask M and two activation I^k and A^k, each complement another to form the final output:

$$h^k_{out} = (1 - M^k) \otimes A^k + M^k \otimes I^k \tag{4.7}$$

Note that the value of attentional matrix M^k is between 0 and 1. As show in Figure 4.1b, the network utilize the AAD Res Block, which include a shortcut connection to enhance learning similarly to ResNetBlock[14]. The AAD ResBlk main backbone was obtain by stack two AAD Layer with two convolutional layer and two ReLU activation. The AAD Generator is the result of cascading many AAD ResBlk with different scale to enlarge and integrate two embeddings.

Overall, the architecture of the network will have the look as visualize in the Figure 4.1a. The network follow an Encoder-Decoder design with multi-scale layer output different feature-maps at each level.

4.2.5 Training Objective

The FaceShifter follow the strategy of adversarial training to achieve realistic result. Multi-scale discriminator was utilized since the generator could have the multiple output feature maps. The adversarial loss was computed using hinge loss:

$$\mathcal{L}_{adv}(\hat{Y}_{s,t}) = \max(0, (1 - X_t) \cdot \hat{Y}_{s,t}) \tag{4.8}$$

An Identity loss is added to the network to enforce the identity preservation:

$$\mathcal{L}_{id} = 1 - \cos\big(z_{id}(\hat{Y}_{s,t}), z_{id}(X_s)\big) \tag{4.9}$$

Also, the training objective include an attribute loss, which defined as the L2 distance between attribute embeddings of target and generated face image:

$$\mathcal{L}_{att} = \frac{1}{2} \sum_{k=1}^{n} \left\| z^k_{att}(\hat{Y}_{s,t}) - z^k_{att}(X_t) \right\|_2^2 \tag{4.10}$$

Specifically, when the source face image and the target face image are the same person, the network compute a reconstruction loss between the generated face image with regard to the target image:

$$\mathcal{L}_{rec} = \begin{cases} \frac{1}{2}\left\| \hat{Y}_{s,t} - X_t \right\|_2^2 & \text{if } X_s = X_t \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

The final training object for the network is described as:

$$\mathcal{L} = \mathcal{L}_{adv} + \lambda_{id}\mathcal{L}_{id} + \lambda_{att}\mathcal{L}_{att} + \lambda_{rec}\mathcal{L}_{rec} \tag{4.12}$$

When training the network, we optionally set the $\lambda_{att} = \lambda_{rec} = 10$ and $\lambda_{id} = 5$.

CHAPTER 5

Experimental Result

5.1 Dataset

We use the dataset from CelebA-HQ[26], FFHQ[20] and VGGFace[34] for training the network. First, we get all the face images from CelebA-HQ and FFHQ and merge together. We then use state-of-the-art face detection in [45] to crop the only part of the face covering hair, chins and some bland background. In specific, we recognize the VGGFace dataset contain many low resolution, blurry and disordered faces that could restrain the performance while training. Therefore, we decided to keep only face with high resolution(i.e bigger than 180x180) for training. We then resize the face images all to 256×256.
Totally, we use nearly one million face samples belong to over 100,000 subject for training the network. We manually set the same ratio to 1:5, meaning that 20% of the training samples will have the source image and target image share the same identity.

5.2 Experimental Result

5.2.1 Performance

We run our framework on a single V100 GPU and recorded the performance. With two input face images size 256x256, it take around 200 ms to swapping the face. We demonstrate the result of face swapping on multiple dataset with variation ages, gender, expression, poses,...etc. To get the best comparison, we take pair of images

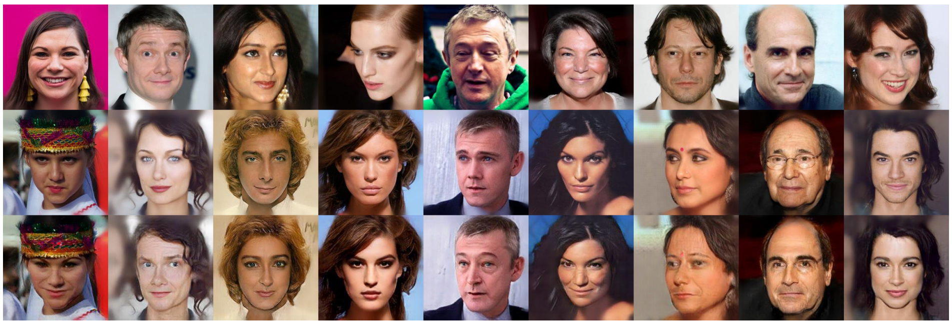


Figure 5.1: *Result of our framework on multiple dataset.*

and run through our framework and FSGAN model taken from [32] and compare the results. This is showed on Figure 5.2.

We can observe that FSGAN swapping strategy focus on synthesize the face with the inner face contour segmentation map, thus making the output face have the same face shape with target face but not source face, creating the discrepancy between face contour and identity features. This result in animated looking images and poor lighting adaptive face. Our method is able to address all these issue considerably. The framework can achieve the high-fidelity generated images by preserve the source face shape and dynamically adjust the lighting condition and face expression to match the target face image. In general, our works can produce more high fidelity results, significantly outperform other old methods.

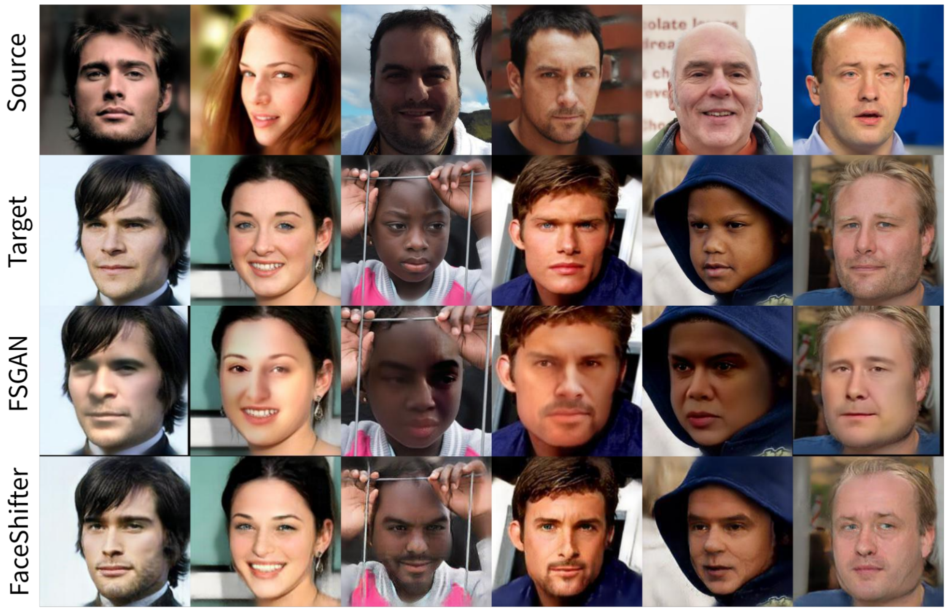


Figure 5.2: *Compare result of our framework with state of the art methods.*

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

In this works, we studied a novel framework for high fidelity face swapping task by using generative adversarial network. We begin by covered a basic knowledge about neural network and the process of developing generative adversarial neural network. We then provided comprehensive overview and in-depth analysis of recent research paper on the topic of using generative adversarial in face swapping and face manipulation field. We also demonstrate how we can accomplish a network with a compact architecture yet efficiently extract face identity and attribute features with high high level of complexity by combining and improving previous works.
Based on our methods, realistic face images were generated through dynamically incorporate identity and attributes feature map of the source and target face respectively. Without subject-specific annotations, our works is able to surpass other approaches in producing accurate high fidelity facial images by providing just two face images.

Unlike previous works which utilize face contour segmentation and mapping between inner regions, our network is able to generalize and bring together identity and attribute mapping between faces. The proposed methods could synthesize the swapped face with face contour from source face and expression, lighting from target face. Extensive experiments show that our framework significantly outperforms current state-of-the-art face swapping methods.
We also publicize the code used for training and testing the model, which we hope will considerably contribute to further research works and related open-source projects.

6.2 Future Work

The Figure 6.1 depicts the effects of our face swapping methods with different failed cases. Obviously, more angular or expression variations occurs need more attention and smoothness in the outer region. Furthermore, in some generated images, the face encounter many blurry part and GAN artifacts position. We believe more stable results could be achieved through either more quality dataset or more robust objective function.
Our work also limited by the resolution of the training data, since the training of high resolution images demand a large amount of computation power that currently is not available. Another drawbacks include our framework include a pretrained face recognition work from, which require carefully training and testing to obtain the best results.

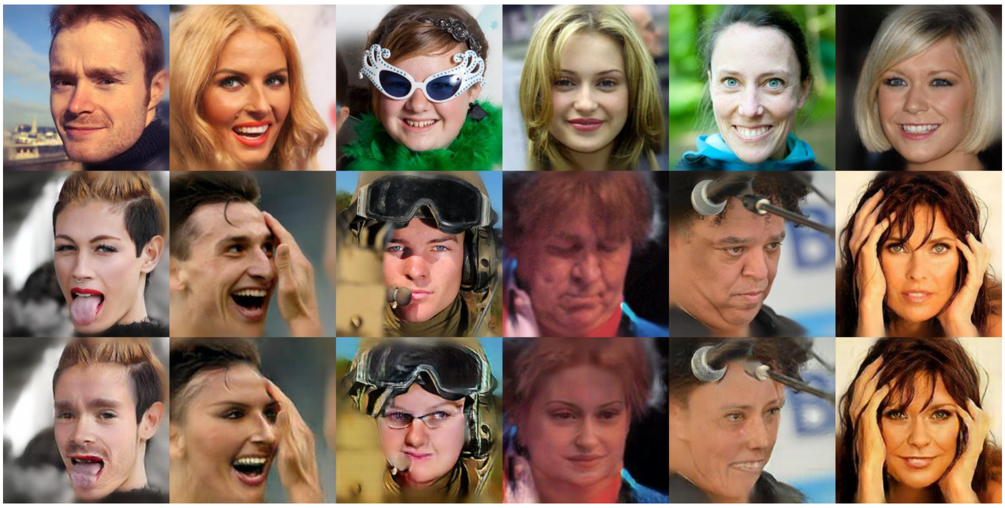


Figure 6.1: *Limitations when using our framework: occlusion, extreme poses, GAN artifacts.*

Bibliography

[1] Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. Creating a photoreal digital actor: The digital emily project. In *2009 Conference for Visual Media Production*, pages 176–187, 2009.

[2] Martin Arjovsky, Soumith Chintala, and LÃ©on Bottou. Wasserstein gan, 2017.

[3] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, and Shree Nayar. Face swapping: Automatically replacing faces in photographs. *ACM Trans. Graph.*, 27, 08 2008.

[4] Volker Blanz, Kristina Scherbaum, Thomas Vetter, and Hans-Peter Seidel. Exchanging faces in images. *Computer Graphics Forum*, 23(3):669–676, 2004.

[5] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 187â194, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[6] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, Hyrum S. Anderson, Heather Roff, Gregory C. Allen, Jacob Steinhardt, Carrick Flynn, Seán Ó hÉigeartaigh, Simon Beard, Haydn Belfield, Sebastian Farquhar, Clare Lyle, Rebecca Crootof, Owain Evans, Michael Page, Joanna Bryson, Roman Yampolskiy, and Dario Amodei. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *CoRR*, abs/1802.07228, 2018.

[7] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. *CoRR*, abs/1710.08092, 2017.

[8] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *CoRR*, abs/1711.09020, 2017.

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. *CoRR*, abs/1912.01865, 2019.

[10] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *CoRR*, abs/1801.07698, 2018.

[11] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017.

[16] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *CoRR*, abs/1702.03275, 2017.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[21] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Trans. Graph.*, 37(4), July 2018.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[24] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *CoRR*, abs/1912.13457, 2019.

[25] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *CoRR*, abs/1603.04779, 2016.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[27] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[28] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes. *ACM Computing Surveys*, 54(1):1â41, Mar 2021.

[29] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[30] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines, 2010.

[31] Ryota Natsume, Tatsuya Yatagawa, and Shigeo Morishima. RSGAN: face swapping and editing using face and hair representation in latent spaces. *CoRR*, abs/1804.03447, 2018.

[32] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: subject agnostic face swapping and reenactment. *CoRR*, abs/1908.05932, 2019.

[33] Yuval Nirkin, Iacopo Masi, Anh Tuan Tran, Tal Hassner, and Gérard G. Medioni. On face segmentation, face swapping, and face perception. *CoRR*, abs/1704.06729, 2017.

[34] A. Zisserman O. M. Parkhi, A. Vedaldi. Vgg face dataset. In *Deep Face Recognition British Machine Vision Conference, 2015.u*, 2015.

[35] Kyle Olszewski, Zimo Li, Chao Yang, Yi Zhou, Ronald Yu, Zeng Huang, Sitao Xiang, Shunsuke Saito, Pushmeet Kohli, and Hao Li. Realistic dynamic facial textures from a single image using gans. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5439–5448, 2017.

[36] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *CoRR*, abs/1903.07291, 2019.

[37] Albert Pumarola, Antonio Agudo, Aleix M. Martínez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. *CoRR*, abs/1807.09251, 2018.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[39] Justus Thies, Michael ZollhÃ¶fer, Marc Stamminger, Christian Theobalt, and Matthias NieÃner. Face2face: Real-time face capture and reenactment of rgb videos, 2020.

[40] Rubén Tolosana, Rubén Vera-Rodríguez, Julian Fiérrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. *CoRR*, abs/2001.00179, 2020.

[41] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *CoRR*, abs/1701.02096, 2017.

[42] Hong-Xia Wang, Chunhong Pan, Haifeng Gong, and Huai-Yu Wu. Facial image composition based on active appearance model. pages 893 – 896, 05 2008.

[43] Lior Wolf, Ziv Freund, and Shai Avidan. An eye for an eye: A single camera gaze-replacement method. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 817–824, 2010.

[44] Sitao Xiang and Hao Li. On the effects of batch and weight normalization in generative adversarial networks, 2017.

[45] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multi-task cascaded convolutional networks. *CoRR*, abs/1604.02878, 2016.

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

APPENDIX A

Network Structure and Training Details

A.1 Network Structure

We demonstrate the network structure in detail in the Figure A.1:

A.2 Training Detail

The framework is trained with 500,000 steps using a single V100 GPU. Due to GPU memory, when training we optionally set the $batch\_size = 8$. Throughout the training, we collected all the data and plotted the network loss:

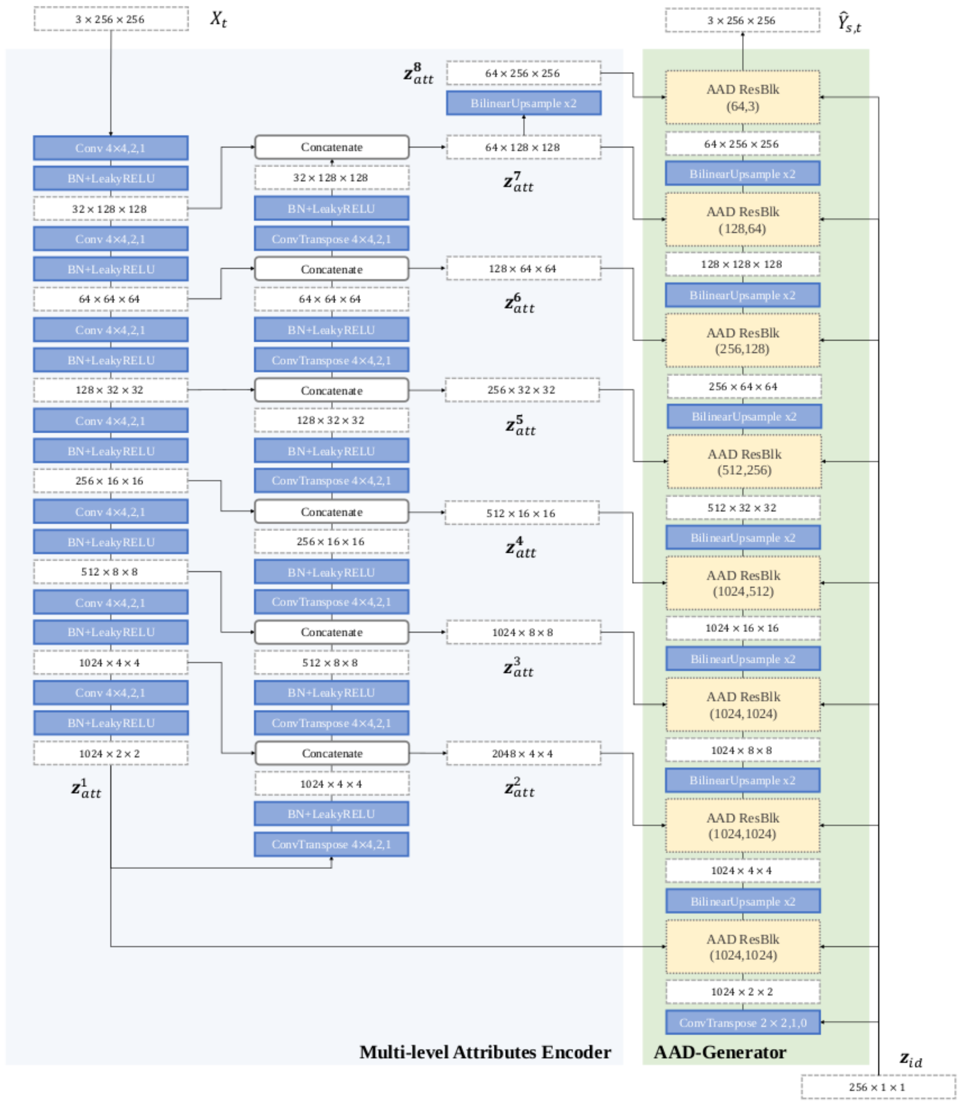


Figure A.1: *Network architecture. Figure taken from [24]*

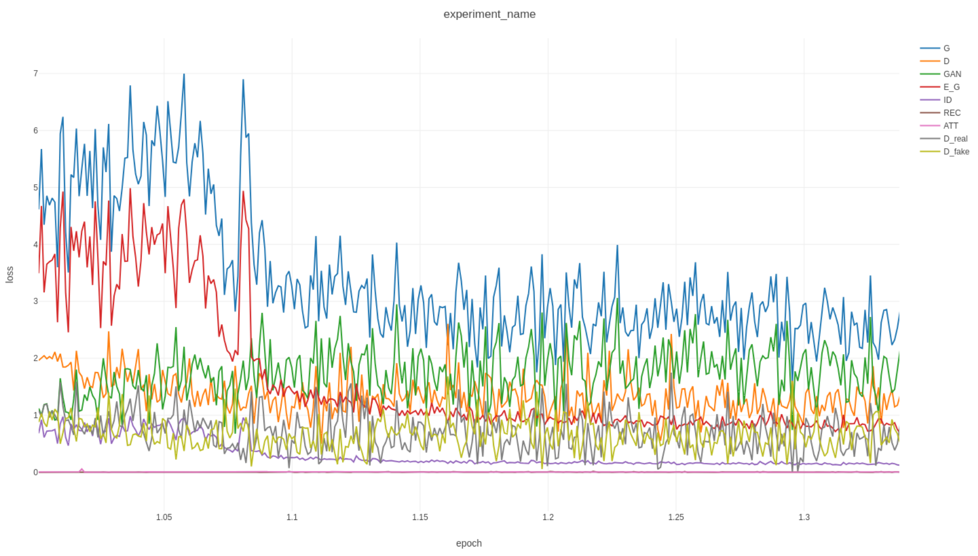


Figure A.2: *Training loss in 200,000 steps.*

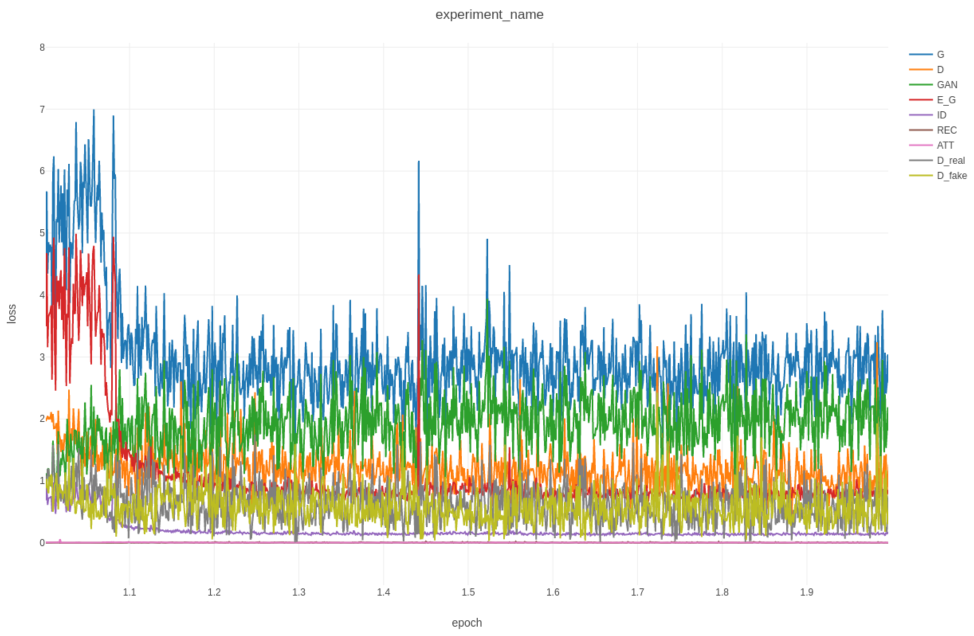


Figure A.3: *Training loss in 500,000 steps.*

APPENDIX B

Sample Code

In this appendix, we demonstrate the adversarial training coding use for training the framework:

```
def optimize_parameters(self):
    # forward
    self.forward(self.target_img, self.source_img)
    # update G
    self.set_requires_grad(self.netD, False)
    self.optimizer_GE.zero_grad()
    self.loss_G = self.compute_G_loss()
    self.loss_G.backward()
    self.optimizer_GE.step()
    # update D
    self.set_requires_grad(self.netD, True)
    self.optimizer_D.zero_grad()
    self.loss_D = self.compute_D_loss()
    self.loss_D.backward()
    self.optimizer_D.step()

def forward(self, target_img, source_img):
    with torch.no_grad():
        Z_id_real = self.netZ(F.interpolate(source_img[:, :, 19:237,
            19:237], size=112, mode='bilinear', align_corners=True))
    self.Z_id_real = F.normalize(Z_id_real).detach()
    self.feature_map_real = self.netE(target_img)
    self.fake = self.netG(self.Z_id_real, self.feature_map_real)
```

```
        Z_id_fake = self.netZ(F.interpolate(self.fake[:, :, 19:237,
            19:237], size=112, mode='bilinear', align_corners=True))
        Z_id_fake = F.normalize(Z_id_fake)
        self.Z_id_fake = Z_id_fake
        self.feature_map_fake = self.netE(self.fake)

    def compute_G_loss(self):
        D_score_fake = self.netD(self.fake)
        self.loss_GAN = self.criterionGAN(D_score_fake, True,
            for_discriminator=False)

        self.loss_ATT = self.criterionATT(self.feature_map_real, self.
            feature_map_fake)
        self.loss_ID = self.criterionID(self.Z_id_real, self.Z_id_target)
        self.loss_REC = self.criterionREC(self.target_img, self.fake,
            self.same)

        self.loss_E_G = self.lambda_ID * self.loss_ID + self.lambda_REC *
             self.loss_REC + self.lambda_ATT * self.loss_ATT

        self.loss_G = self.loss_E_G + self.loss_GAN
        return self.loss_G

    def compute_D_loss(self):
        """Calculate GAN loss for the discriminator"""
        fake = self.fake.detach()
        D_score_real = self.netD(self.target_img)
        D_score_fake = self.netD(fake)

        self.loss_D_real = self.criterionGAN(D_score_real, True)
        self.loss_D_fake = self.criterionGAN(D_score_fake, False)

        self.loss_D = self.loss_D_fake + self.loss_D_real
        return self.loss_D
```