3D Facial Reconstruction – inferring missing face sections from face poses at an angle

Xuyang Shen

A report submitted for the course
COMP3740 Project Work in Computing
Supervised by: Jo Plested, Tom Gedeon
The Australian National University

June 2020
© Xuyang Shen 2020
Except where otherwise indicated, this report is my own original work.

Xuyang Shen
12 June 2020
Acknowledgments

I would like to express my sincere gratitude to my supervisor Jo Plested and Professor Tom Gedeon, for their patience, enthusiasm, and immense support throughout this project. I would also wish to thank Dr Sabrina Caldwell and Dr Liaxng Zheng, for their excellent advice in the presentation of this project.

Lastly, I would like to express my great appreciation to my parents as well as my friends for their support and encouragement.

Without all of you, I can’t complete this work.
Abstract

Three-dimensional facial reconstruction is one of the popular applications in computer vision. However, even its state-of-art model still requires a frontal face as inputs, which restricts its usage scenarios in the wild. A similar dilemma also happens in face recognition; hence, the idea to recover the frontal face from a single side-pose facial image emerges. The state-of-art in this area is the Face-Transformation generative adversarial network, built based on the Cycle-GAN from pixel-to-pixel transformation. It inspires us to explore further the performance of two models from pixel transformation in frontal facial synthesis, which are Pix2Pix and Cycle-GAN.

In this paper, we firstly conduct the experiments on five different loss functions on Pix2Pix model to improve its performance, then followed by proposing a new network Pair-GAN in frontal facial synthesis. Pair-GAN uses two parallel U-Nets as the generator and PatchGAN as the discriminator. The detailed hyper-parameters of it are also discussed. Based on the quantitative measurement by face similarity comparison, we can obtain from the results that Pix2Pix model with L1 loss, GDL loss (gradient difference loss), and identity loss can results in below 8 % of improvement compared to default Pix2Pix model. Additionally, the performance of Pair-GAN is 13 % better than the CycleGAN and 25 % than the Pix2Pix model. Besides, limitation and future work of this project is also discussed in the final section.
# Contents

Acknowledgments iii  
Abstract v  

1 Introduction 1  
1.1 Contribution ......................................................... 2  
1.2 Report Outline ...................................................... 3  

2 Background and Related Work 5  
2.1 Background ................................................................... 5  
2.1.1 Convolution Neural Network Architectures ................. 5  
2.1.2 Generative Adversarial Networks .............................. 9  
2.1.3 Conditional Generative Adversarial Networks .......... 11  
2.2 Related work .............................................................. 14  
2.2.1 Frontal Facial Synthesis ....................................... 14  
2.3 Summary ................................................................. 16  

3 Experimental Methodology and Setup 17  
3.1 Dataset and Pre-processing ........................................ 17  
3.2 Synthesis Loss Function ............................................ 19  
3.2.1 Experimental setup ............................................... 19  
3.2.2 Overview of four loss functions ............................. 19  
3.2.3 Adversarial Loss ...................................................... 20  
3.2.4 L1 Loss (Mean Absolute Error) ............................... 20  
3.2.5 Gradient Difference loss ..................................... 20  
3.2.6 Symmetry Loss ....................................................... 21  
3.2.7 Identity Loss .......................................................... 22  
3.3 Network Architecture: Pair-GAN ................................ 22  
3.3.1 Generator .............................................................. 23  
3.3.2 Discriminator .......................................................... 24  
3.3.3 Loss Functions ....................................................... 25  
3.4 Software platform ...................................................... 27  
3.5 Hardware platform ..................................................... 27  

4 Results and Evaluation 29  
4.1 State-of-Art Models in Frontal Facial Synthesis Analysis .... 29  
4.2 Loss Function Analysis .............................................. 31
## Contents

4.3 Pair-GAN Analysis .................................................. 33

5 Conclusion ............................................................... 37
  5.1 Limitation and Future Work ....................................... 37

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

1.1 Display the 3D reconstruction model from side-pose face (left) and frontal face (right). Results are provided by Volumatic regression Network (VRN) [Jackson et al., 2017] .................................................. 1

2.1 A Simple Convolution Neural Network Architecture: LeNet-5 [LeCun et al., 1998]. The architecture from left to right is Input Layer, Convolution layer, Sub-sampling Layer, Convolution layer, Sub-sampling Layer, Fully Connection Layer, Fully Connection Layer, and Output Layer. ................................. 6

2.2 Manipulate a 3x3 kernel on 5x5 input data. The fist 3x3 local receptive field is located at top-left of the input data, which is marked as yellow square. The calculation formula refers to the formula 1. For example, the result 4 in activation map is derived from 1x1 + 1x0 + 1x1 + 0x0 + 1x1 + 1x0 + 0x1 + 0x0 + 1x1. ......................................................... 7

2.3 Perform average-pooling (left) and max-pooling (right) to the original data with 2x2 filters and stride 2 .............................................. 7

2.4 How dropout works. Image from Srivastava et al. [2014] .................. 8

2.5 Difference of batch normalization and instance normalization. Image from Wu and He [2018] ................................................................. 9

2.6 Abstract architecture of DCGAN with the training results on MNIST handwritten database after 50 epochs. Results of generated data are from TensorFlow DCAGN tutorial [TensorFlow, 2020] ........................................ 9

2.7 Convolution operation (left) with 3x3 kernel size, 0 padding, and strides = 1. Transpose Convolution operation (right) with 3x3 kernel size, 0 padding, and strides=1. Images refer to paper [Dumoulin and Visin, 2016] ......................................................... 10

2.8 Generator architecture of DCGAN (left) [Radford et al., 2015] and fractionally strided convolution (right). [Dumoulin and Visin, 2016] . . 11

2.9 Forward pass of CGAN on MNIST handwritten database, based on the [Reed et al., 2016] ................................................................. 12

2.10 Abstract architecture of auto-encoder with images from [Zhu et al., 2017] 13

2.11 Overview the forward pass of Pix2Pix based on the [Isola et al., 2017] . 13

2.12 Overview the generator of CycleGan based on the [Zhu et al., 2017] . 14

2.13 Network architecture of TP-GAN(Two Pathways Generator Adversarial Network) from Huang et al. [2017b] .............................. 15

2.14 Network architecture of FT-GAN(Pose-Invariant Generator Adversarial Network) from Zhuang et al. [2019] ........................................ 15
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Sample of color FERET Database. From left to right is: frontal face, left pose face at 67.5 angle, right pose face at 67.5 angle, left pose face at 90 angle, right pose face at 90 angle.</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison between raw data from FERET and pre-processed images. Top: Raw data of same person but taken in two different times (940928; 940620); Below: cropped images by MTCNN.</td>
</tr>
<tr>
<td>3.3</td>
<td>U-Net architecture used in Pix2Pix. Image is from [Isola et al., 2017].</td>
</tr>
<tr>
<td>3.4</td>
<td>Overview of loss functions used in this experiment.</td>
</tr>
<tr>
<td>3.5</td>
<td>Intuitive view of L1 loss calculation between the ground truth image and the generated image.</td>
</tr>
<tr>
<td>3.6</td>
<td>Intuitive view of GDL loss calculation between the ground truth image and the generated image.</td>
</tr>
<tr>
<td>3.7</td>
<td>Intuitive view of symmetry loss calculation inside the generated image.</td>
</tr>
<tr>
<td>3.8</td>
<td>Intuitive view of identity loss calculation between the ground truth image and the generated image.</td>
</tr>
<tr>
<td>3.9</td>
<td>Generator’s architecture of Pair-GAN.</td>
</tr>
<tr>
<td>3.10</td>
<td>Intuitive view of the discriminator in Pair-GAN (Patch-based discriminator of GANs) [Isola et al., 2017].</td>
</tr>
<tr>
<td>3.11</td>
<td>Experiments on different size of receptive field. Reference to Pix2Pix model [Isola et al., 2017].</td>
</tr>
<tr>
<td>3.12</td>
<td>Intuitive view of pair loss calculation between the left generated image and right generated image.</td>
</tr>
<tr>
<td>3.13</td>
<td>Overview of loss functions explored in PairGAN.</td>
</tr>
<tr>
<td>4.1</td>
<td>Examples of facial comparison by using Face++.</td>
</tr>
<tr>
<td>4.2</td>
<td>Average similarity, best similarity, and worst similarity between ground truth and generated image by Pix2Pix model and Cycle-GAN with default loss ratio and train epoch is 125. Higher similarity is better.</td>
</tr>
<tr>
<td>4.3</td>
<td>Best result derived from of Pix2Pix model and CycleGAN with default loss ratio and train epoch is 125.</td>
</tr>
<tr>
<td>4.4</td>
<td>Compare similarity between ground truth and generated image by Cycle-GAN on different epochs but with default loss ratio. Higher similarity is better.</td>
</tr>
<tr>
<td>4.5</td>
<td>Compare average similarity between ground truth and generated image on different loss configurations among 40 test images. Two brown color bars are the state-of-art model (Pix2Pix, CycleGAN) mentioned in the above section. Higher similarity is better.</td>
</tr>
<tr>
<td>4.6</td>
<td>Monitoring changes of symmetric loss during whole training progress. The plot is generated by TensorBoard.</td>
</tr>
<tr>
<td>4.7</td>
<td>Experiment of comparing different loss configuration.</td>
</tr>
<tr>
<td>4.8</td>
<td>Compare PairGAN’s performance with Pix2Pix and CycleGAN.</td>
</tr>
<tr>
<td>4.9</td>
<td>Compare generated images by CycleGAN and PairGAN.</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Python packages used in Data Pre-processing ........................................ 27
3.2 Python packages used in loss function and network architecture experi-
    ment ................................................................. 27
3.3 Hardware platform details ................................................................. 28
3.4 GPU Computation ability ................................................................. 28

4.1 Results of different configurations on PairGAN. Where Adv means
    the proportion of adversarial loss; L1 means the proportion of L1 loss;
    GDL means the proportion of gradient difference loss; Id means the
    proportion of identity loss; Pair means the proportion of pair loss; ..... 34
Chapter 1

Introduction

Three-dimensional (3D) facial reconstruction from a single image is one of the popular topics in computer vision from the past twenty years. The landmark in three-dimensional facial reconstruction without applying learning strategy is 3D Morphable Model (3DMM), which is proposed in 1999 [Blanz and Vetter, 1999], and continually being followed from period 1999 to 2015. Due to the outstanding performance of convolution neural network (CNN) in computer vision, several researchers proposed end-to-end CNN which can directly calculate 3DMM shape based on a single input image, such as Volumatic Regression Network (VRN) [Jackson et al., 2017]. Although end-to-end learning methods achieve the state-of-the-art in three-dimensional facial reconstruction, they still cannot infer the missing facial information if the input facial image is a side-pose at more than 30 angles in our experiments (figure 1.1).

Figure 1.1: Display the 3D reconstruction model from side-pose face (left) and frontal face (right). Results are provided by Volumatic regression Network (VRN) [Jackson et al., 2017]

This issue is not only happened on three-dimensional facial reconstruction tasks, but also commonly exists on other facial applications in computer vision. Face recognition as a widely used authentication, and detection technique requires the frontal faces at the current stage. However, this condition is challenging to be satisfied in the real-life, especially for video surveillance which captures the object under any situations.

Consequently, inferring the missing facial information is a mildly right solution for these applications to solve the issue raised by the side-pose facial image. Inferring missing face sections from face poses at an angle is also known as the frontal facial synthesis, which is performed by deep learning since the kernel filter is applicable to learn the non-linear transformation from different angles. Specifically, the auto-encoder network equipped with feature extraction (decoder) and image generation (encoder) is an option to achieve frontal facial generation. Stacked auto-encoders pro-
posed by Kan et al. is able to recover the frontal faces from unconstrained poses [Kan et al., 2014]. Furthermore, generative adversarial networks (GANs) based methods replace the auto-encoder networks to perform the frontal facial synthesis in recent years since the model itself is more robust when dealing with the pose image in a large angle [Huang et al., 2017b; Zhuang et al., 2019].

Based on this background, GANs based frontal facial synthesis draws our attention, which we continued to explore the style transformation achieved by conditional generative adversarial networks (CGANs). Style transformation refers to the tasks that perform the pixel-to-pixel transformation from one style to another one. Hence, frontal facial synthesis also belongs to the style transformation as all facial features of the original image are reserved in frontal face image except for the pose. The connection between style transformation and frontal facial synthesis is also indicated in the Pose-Invariant generative adversarial network (FT-GAN) which takes the CycleGAN (the most successful model in style transformation) as a base to achieve state of the art in the frontal facial synthesis [Zhuang et al., 2019].

With these motivations, we begin our project by exploring two models from style transformation into frontal facial synthesis, which are Pix2Pix and CycleGAN. This work aims to evaluate the performance of these two models under default hyper-parameters in color FERET Database (Facial Recognition Technology) [Patricia.flanagan@nist.gov, 2019]. Based on this baseline, we conduct our experiments in two stages:

1. We firstly apply different loss functions to Pix2Pix model, and evaluate their performance.

2. Based on the network architecture of Pix2Pix and CycleGAN, we proposed a new CGAN network named as Pair GAN. Additional, we also compare its performance with state-of-the-art models.

The detail of our experiments are described in the Chapter 3. Besides, we also employ multiple evaluation techniques to examine the results from experiments, including facial similarity comparison, loss comparison, and object-visualising (Chapter 4).

1.1 Contribution

The main contribution of this project are listed below:

- Five different loss functions are analyzed to improve the performance of Pix2Pix model in frontal facial synthesis, including adversarial loss, L1 loss, gradient difference loss, symmetry loss, and identity loss. The best achievement results in an 8% improvement compared to Pix2Pix model, which is close to Cycle-GAN’s performance.

- Pair-GAN as a new network architecture of CGAN is proposed, which particular targets at frontal facial synthesis. It gains 13% better than the CycleGAN and 25% than the Pix2Pix model.
A new quantitative measurement on model performance is introduced to evaluate the similarity between the generated frontal face and ground-truth image.

1.2 Report Outline

This report has totally 6 chapters:

- Chapter 2 includes: background about CNN, GAN, CGAN, CGAN in style transformation; related works of this project which also includes current state-of-art in frontal facial generation.

- Chapter 3 thoroughly describes the color FERET Database and its pre-processing; Pair GAN network architecture; different loss function attempted in the experiments; experimental setup.

- Chapter 4 evaluates and concludes the results from experiments; compares our work to some state-of-art results.

- Chapter 5 summarises what we did in this project as well as the limitation and future works.
Introduction
Chapter 2

Background and Related Work

Chapter 2 mainly covers the background knowledge and related work. Our study focuses on the frontal facial generation by applying conditional Generative Adversarial Network (CGAN); hence this chapter explored the Convolution Neural Network (CNN), Generative Adversarial Network (GAN) and their intersection of the style transformations task.

Section 2.1 gives background material in the architecture of CNN, as well as an overview of GAN and CGAN. Section 2.2 introduces a couple of state-of-art methods in frontal facial synthesis, and CycleGAN as one of the milestones in style transformation task.

2.1 Background

2.1.1 Convolution Neural Network Architectures

Convolution Neural Network (CNN) is a type of neural network which targets spatial problem domain, such as digits classification. CNN is also considered as a foundation of deep learning since it won the challenge of ImageNet Large-Scale Visual Recognition (ILSVRC) in 2012 [Krizhevsky et al., 2012]. Since then, more and more research studied on the CNN, and developed a couple of deep Convolution Neural Networks (DCNN) which continually refreshed the accuracy records in ILSVRC [Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Huang et al., 2017a].

Despite the fact that various modification is applied on the modern neural networks, such as LSTM, GAN, they are still sharing some basic techniques with CNN, including local field, weight sharing, and pooling. To be specific, the convolution layer, pooling layer, and fully connected layer are the three essential components in the architecture of CNN (figure 2.1). Notably, the convolution layer, together with the pooling layer is also the key element to build the complex and deep CNN models. The following subsections covered the detailed description of these three layers.
Background and Related Work

Figure 2.1: A Simple Convolution Neural Network Architecture: LeNet-5 [LeCun et al., 1998].
The architecture from left to right is Input Layer, Convolution layer, Sub-sampling Layer,
Convolution layer, Sub-sampling Layer, Fully Connection Layer, Fully Connection Layer, and
Output Layer.

Convolution Layer

Convolution layer is a core block of CNN, which aims to produce a feature map from the input data. Each convolution layer is built by learnable filters and activation functions, in which each filter has a set of neurons that can apply to a specific area of input data named as local receptive field [LeCun et al., 1995]. Once data is fed in, the filter kernel begins to systematically scan the data block by block and learns to activate when it sees a specific feature, which works similar to biological neuron in visual cortex [Goodfellow et al., 2017]. From the mathematical perspective, a feature map is calculated from the linear operation of input data, weights carried by a filter kernel (2.2). Moreover, a different filter can be applied to the same data and getting a new feature map, which builds up the entire convolution layer. For example, the feature map generation is repeated six times in the first convolution layer (figure 2.1).

\[
feature \, map = f(input \otimes weights \oplus bias) \\
= f\left( \sum\limits_{y=0}^{\text{columns}} \sum\limits_{x=0}^{\text{rows}} (input(x - a, y - b) \times kernel(x, y)) + bias \right)
\]  

(1)

where “feature map” is the convolved data; \( f(\ast) \) is the activation function; “input” is the original data; weights is the filter kernel; bias is an independent value coordinating with filter kernel, but it is not commonly applied in the large model; weights and bias are optimized by gradient descent during training progress.

Activation function generally is a non-linear transformation applied on the arithmetic result of convolving data, but before the pooling operations. If convolutional layer without activation function, only simple linear features would be extracted through successive linear operations. The typical activation function in CNN is Rectified Linear Unit (ReLU), which the sigmoid and hyperbolic tangent function is widely utilized before it is introduced to the deep learning [Glorot et al., 2011]. Technically, ReLU function only activates the positive input value and ignores the negative one.
\section*{2.1 Background}

**Figure 2.2:** Manipulate a 3x3 kernel on 5x5 input data. The fist 3x3 local receptive field is located at top-left of the input data, which is marked as yellow square. The calculation formula refers to the formula 1. For example, the result 4 in activation map is derived from $1 \times 1 + 1 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 0 + 0 \times 0 + 0 \times 1 + 0 \times 0 + 1 \times 1$.

This property helps to speed up convergence and induce sparsity, whereas it also causes dead ReLU issue [Glorot et al., 2011]. Instead of dropping out the negative part, Leak ReLU as an improved ReLU function was introduced, which not only keeps ReLU positive features but also solves the neuron dying issue.

$$ReLU : \quad f(x) = \max(0, x) \quad (2)$$

$$Leaky ReLU : \quad f(x) = \max(ax, x) \quad a = 0.01 \quad (3)$$

**Pooling layer (Sub-sampling Layer)**

The word pooling and sub-sampling in deep learning refer to the same concept whose functionality is to summarise the feature map from the previous convolution layer through a particular function, such as average-pooling, max-pooling. This mechanism reduces the number of parameters as well as connecting the adjacent local receptive field. In the meantime, it also loses partial spatial information, which affects the stability of the reconstruction task in pixel level. Hence, strided convolution was more recommendable than the sub-sampling operation in the previous GAN paper [Radford et al., 2015]. However, opposite experiments results were published in the recent GAN research, as pooling layer is more effective than convolution-only architecture [Barua et al., 2019]. Accordingly, pooling operation is still a disputing topic in the GAN model.

**Figure 2.3:** Perform average-pooling (left) and max-pooling (right) to the original data with 2x2 filters and stride 2
Fully connected layer

The concept of the fully connected layer was derived from multilayer perceptron (MLP), and is added after the convolutional layers in CNN. Each neuron in a fully connected layer is connected to all neurons in the previous layer, while neurons in the convolutional layer are partially connected to preceding layers. This special structure determines the extracted feature maps from convolutional layer will lose their spatial topology in the fully connected layer [Goodfellow et al., 2017], and are expanded into a vector which can perform classification. Similar to the pooling layer, the influence of fully connected layers to GAN becomes a controversial and developing topic since the research of Fully Connected and Convolutional Net Architecture for GANs (FCC-GAN). It points out the spatial loss caused by fully connected layers has a minor influence on GAN. [Barua et al., 2019].

Dropout

When a complex feed-forward neural network is trained on a small dataset, it is easy to cause the over-fitting. In order to prevent it, the performance of the neural network can be improved by avoiding the combination of feature detectors, which is known as dropout [Hinton et al., 2012; Srivastava et al., 2014]. In detail, after enabling dropout, activation value of several neurons will stop performing with a pre-defined probability p in each training pass, which enforces the network skip some local features (as it shown in Figure 2.4). As a result, a deep convolution neural network with dropout is more generalized.

![Figure 2.4: How dropout works. Image from Srivastava et al. [2014]](image)

Batch Normalization and Instance Normalization

Batch normalization is applied to pull the data in each batch back to a positive distribution with a mean of 0 and a variance of 1, which ensures the consistency of the data distribution and avoid the disappearance of gradients [Ioffe and Szegedy, 2015]. It is suitable for classification tasks, such as face recognition. On the other hand, batch normalization is also sensitive to the batch size since mean and variance are calculated on one batch each time. If the batch size is set up to a small value, the calculated mean and variance are not accurate to represent the entire data distribution.

Compared to batch normalization, instance normalization is to normalize a single channel in a single image (figure 2.5), which is mainly designed for the style trans-
formation by GAN. The output of the style transformation task primarily depends on each image instance; hence, normalizing the entire batch is not suitable for it. Instance normalization can not only accelerate model convergence but also maintain the independent [Ulyanov et al., 2016].

2.1.2 Generative Adversarial Networks

Generative adversarial network (GAN) as one of deep learning models attracts an increasing number of researchers since it was proposed by Goodfellow et al. [Goodfellow et al., 2014]. Compared to the vanilla convolution neural network as discussed above, a typical GAN consists of a generative model and a discriminative model, which targets at generation and evaluation, respectively. Specifically, the generator learns the distribution of data and reproduce candidates from a latent space, while discriminator distinguishes them from the actual data distribution. Besides, it is to be observed that generators and discriminators are not limited to be neural network models in GAN theory, which can be any functions that achieve responsibility [Goodfellow et al., 2014].

Forward Pass

Unlike the vanilla convolution neural networks, forward pass of GAN training in one epoch includes three steps. Firstly, the generator takes a vector of random numbers in
Gaussian distribution as inputs, and outputs one generated data \( G(z) \) (figure 2.6 left). Then, one real data from the database is randomly selected as \( x \) where the example in figure 2.6 is a handwritten image 9. In the last stage, both generated data and real data are passed into the discriminator (figure 2.6 right). It will return a scalar value from \([0, 1]\), which represents the ranks of synthesis. Usually, the label sampling from real database is \((D(x), 1)\), while the \((D(G(z)), 0)\) is for generated data.

**Transposed convolution (Fractionally-Strided Convolution, Deconvolution)**

Transposed convolution layers are also known as fractionally-strided convolution layers and deconvolution layers, which are utilized to achieve the functionality of GAN. Transposed convolution performs to transfer the matrix with the same shape as the convolution output into the same matrix as the convolution input (figure 2.7, that is, considered as a special convolution operation. It is to be observed that the data as convolution input cannot be restored by applying transposed convolution since there exist free variables in the weight matrix of kernel [Dumoulin and Visin, 2016].

![Figure 2.7: Convolution operation (left) with 3x3 kernel size, 0 padding, and strides = 1. Transpose Convolution operation (right) with 3x3 kernel size, 0 padding, and strides=1. Images refer to paper [Dumoulin and Visin, 2016]](image)

**Back-propagation**

The generator and the discriminator are optimized separately in the back-propagation. The responsibility of discriminator is to differentiate between real samples and generated samples, the target of \( D(x) \) in formula 4 is close to 1 whereas the value of \( D(G(z)) \) is expected to 0. Therefore, the optimization regarding discriminator is to maximize the \( V(D, G) \).

\[
\text{max} V(D, G) = E_{x \sim P_{\text{data}}(x)}[\log(D(x))] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]
\]

(4)

Unlike supervised machine learning, there is no ground truth for each generated data. Hence, \((D(G(z)), 0)\) is utilized to optimize the generator, which derives similar formula as 4. The target of the generator is to produce the synthesis close to real data. Therefore, the ideal value \( D(G(z)) \) is close to 1, which leads to minimizing the \( V(D, G) \).
\[ \min V(D, G) = E_{z \sim P(z)} \left[ \log(1 - D(G(z))) \right] \] (5)

In majority GAN applications, the discriminator is chosen to updated firstly, then followed by the generator, since this training flow is beneficial for the discriminator to provide an accurate loss and gradient to the generator [Goodfellow et al., 2014]. In concluding the forward pass and back-propagation, the training process of traditional GAN networks is similar to a min-max competition between two networks, which the generator attempts to fool the discriminator. In the meanwhile, the discriminator aims to clarify the fake samples against the real data.

**Architecture of GAN with Convolution Neural Network**

![Generator architecture of DCGAN (left) and fractionally strided convolution (right).](image)

Figure 2.8: Generator architecture of DCGAN (left) [Radford et al., 2015] and fractionally strided convolution (right). [Dumoulin and Visin, 2016]

Combining the GAN model with convolution neural networks is the main branch in GAN’s researches, where CNN is the primary function of generators and discriminators. Among these hybrid models, DCGAN can be treated as the most typical and successful one. The generator and discriminator of DCGAN are two modified CNNs, which the pooling layers in vanilla convolution neural networks are replaced into fractionally strided convolution and strided convolution respectively (figure 2.8). In addition, after the input vector being passed four convolution layers, generators directly output the 64 by 64 data without fully connected layers. Theoretically, these two modifications aim to increase the stability of GANs, but also increases the time for networks to converge [Radford et al., 2015]. The architecture of discriminator in DCGAN can be treated as the flipped generator, which takes 64 by 64 data as input and convolves into a scalar value.

### 2.1.3 Conditional Generative Adversarial Networks

Compared to generative adversarial networks, conditional generative adversarial networks (CGANs) add constraints to the generator and discriminator, which aims to control the content of synthesis [Mirza and Osindero, 2014]. As shown in figure 2.9, the generator takes a random array in normal distribution and a condition as inputs, and produces a data which closes to real samples and satisfies the condition. It is to be
observed that conditions can be either words or images. Apart from the modifications on generators, the classification tasks of the discriminator are adjusted as well. It is not only responsible for distinguishing real data from the generated data, but also required to examines whether they are matching the condition or not. In practical, the output of discriminator in CGAN is simplified into a scalar value, which is same as normal GAN. Specifically, if input data of discriminator is realistic as well as satisfied with condition, the discriminator will return 1. On the other hand, if either of them fails, it will return 0 as two examples in figure 2.9.

![Figure 2.9: Forward pass of CGAN on MNIST handwritten database, based on the [Reed et al., 2016](#)](image)

**Back-propagation of CGAN**

Based on the functionality of GAN, the CGAN also considers the input conditions, which results in analogous modification on the loss functions. As shown in the formula (6), three blocks are corresponding to different situations in discriminator, including realistic images with matching the condition, realistic images with unsatisfying the condition, and synthetic images with matching conditions. The ground truth of the first situation is 1, which leads to maximizing the first item \((c^i, x^i)\) in the formula. On the contrary, the ground truth of other two situations are 0, and as a result, the negative signs are added before the \((c^i, \tilde{x}^i)\) and \((c^i, \hat{x}^i)\). Apart from the discriminator, the loss function of generators is to get a high scalar under condition \(c^i\) through deceiving discriminators, which is similar to traditional GANs.

\[
\max_V(D) = \frac{1}{m} \sum_{i=1}^{m} \log D(c^i, x^i) + \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(c^i, \tilde{x}^i)) + \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(c^i, \hat{x}^i))
\]

(6)

\[
\max_V(G) = \frac{1}{m} \sum_{i=1}^{m} \log D(G(c^i, z^i))
\]

(7)

where \(c^i\) represents the condition; \((c^i, x^i)\) represents the positive condition; \(z^i\) represents the random values; \(\tilde{x}^i\) represents the generated data; \((c^i, \tilde{x}^i)\) represents the object from database but in false condition.
CGAN in Style Transformation - pix2pix and CycleGAN

Style transformation refers to the task that transfers the input image from one domain to another domain. There is a special neural network structure to complete this task named auto-encoder, which contains one encoder network and one decoder network. Analyzing and extracting the features of input images are the responsibility of encoder, while the decoder performs to modify partial encapsulated information and reproduce into a new image. As the Zebra example shown in figure 2.10, the result of the encoder is the summarized information of input images, such as two big items with black-white skin color and four legs. After decoder receiving these data, it alters the skin color from black-white into brown, which becomes the horse. Besides, it is to be observed that the kernel filter controls which information needs to be preserved or modified.

![Figure 2.10: Abstract architecture of auto-encoder with images from [Zhu et al., 2017]](image)

Theoretically, this architecture seems to be reasonable, since the kernel filter can be trained based on the specific domain. However, the generated image by using the auto-encoder structure is usually blur and distorted in practices. The reason behind is that the auto-encoder usually trends to take the average value between the generated image and ground-truth to reduce the loss. In order to solve this issue, Isola et al. proposed the Pix2Pix model by applying the auto-encoder into CGAN [Isola et al., 2017]. The generator of Pix2Pix model is an auto-encoder based U-Net, which transforms the image from one domain A into another domain B. The discriminator is responsible to distinguishes whether the input image belongs to the domain B or not through learning the real image data in that domain 2.11. Based on this mechanism, the blur issue from auto-encoder is eliminated, since unclear generated images will receive a low mark from the discriminator.

![Figure 2.11: Overview the forward pass of Pix2Pix based on the [Isola et al., 2017]](image)

There is an issue exposed within the competition between generator and discriminator in Pix2Pix model, which the generator can deceive the discriminator by
generating a realistic image but unrelated to the input domain. To eliminate this drawback, Zhu et al. proposed the CycleGan by doubling the generator of Pix2Pix model [Zhu et al., 2017]. From the structure of CycleGan in figure 2.12, we can notice that images from two domain are constructed into a loop through two symmetric generators. This consistency restricts two generators to focus on the input feature since there must one generator is cheating if the image in one domain cannot be traversed back. In conclusion, the network architecture of CycleGan is treated as an outstanding sample in style transformation.

![Figure 2.12: Overview the generator of CycleGan based on the [Zhu et al., 2017]](image)

2.2 Related work

2.2.1 Frontal Facial Synthesis

As described in the introduction, frontal facial generation is an essential topic in computer vision, which draws an increasing number of researches attention on it, particularly in combining the deep learning technique. In 2013, a model using many-to-one encoder was proposed to face frontalization for face recognition Zhang et al. [2013], which represents the beginning stage of applying the neural network in frontal facial generation. Then, a stacked progressive auto-encoders proposed by Kan et al. is utilized to synthesize the frontal face based on one side-pose face image [Kan et al., 2014]. Specifically, each progressive auto-encoder is aimed to map the facial section in a large pose into a virtual view at smaller ones, while the facial part in a small pose remains unchanged. Based on progressively stacking each transformed facial section, the frontal facial image is synthesized eventually. After a short period, methods based on three-dimensional space to infer missing face sections became popular [Hassner et al., 2015; Yin et al., 2017]. Specifically, Hassner et al. suggested to use 3D model understand the facial information, and approximate the frontal face based a single face at a small angle [Hassner et al., 2015]. In 2017, the frontalization generative adversarial network (FF-GAN) was proposed by [Yin et al., 2017]. FF-GAN firstly extracts the coefficient of the 3D morphable model (3DMM) from input 2D side-pose image. It is followed by passing both 3DMM coefficients and original 2D images into GAN, which generates the frontal data eventually.

Since generating data based on the inputs is the specialization of generative adversarial networks (GAN), an increasing number of researchers proposed their models in frontal facial synthesis based on it, but back into two-dimension again [Huang
§2.2 Related work

Firstly, Huang et al. proposed the two pathways generator adversarial network (TP-GAN) [Huang et al., 2017b]. From its structure (figure 2.13, two generators are working simultaneously, which is responsible for recovering global facial information and local facial features, respectively. Compared to stacked progressive auto-encoder, TP-GAN is more robust when dealing with different side-pose images, even at a large angle. Besides, benefiting on the advantages of GAN, TP-GAN can generate the synthesis more precise. In 2018, CR-GAN was designed to generating multi-view images from a single-view input. This model not only has a reconstruction path, but also equips a generation side to maintain the completeness of learned embedding space [Tian et al., 2018].

The state-of-the-art technique at frontal facial synthesis is FT-GAN (Pose-Invariant Generator Adversarial Network) proposed by [Zhuang et al., 2019], which is trained and test on CMU Multi-PIE dataset. In detail, FT-GAN combines the CycleGAN (Cycle-Consistent Adversarial Networks) and key point alignment to generate frontal facial synthesis more realistic. The reason behind it is that CycleGAN is considered as the state-of-the-art in pixel-to-pixel transformation. Based on the high-quality synthesis, the key point alignment adds the spatial relationship of original facial features, which makes the generated face closer to the original person. From the result indicated from their experiments, FT-GAN is 10 % and 4 % better than CR-GAN and CycleGAN respectively when the inputting side-pose image is between 60
degrees to 75 degrees.

2.3 Summary

In this chapter, we had a brief overview of convolution neural networks, including their network architecture, convolution layer, pooling layer, and fully connected layer. Based on the basic knowledge of CNN, we explored the generative adversarial networks (GANs) and conditional adversarial networks (CGANs) by introducing their forward pass and back-propagation. The application of CGAN in the pixel-to-pixel transformation was also shown at the end of the background. Apart from that, we also briefly reviewed some recent research on frontal facial synthesis.

With the background knowledge on CGAN and frontal facial synthesis, we can continue exploring the rest of this project, including methodology and experiment results.
In Chapter 3, we mainly cover the dataset utilized for this project and detailed implementation of each experiment. Since our purpose is to improve the current state-of-art model in the frontal facial synthesis, we firstly explored different loss functions on Pix2Pix model, and followed by studying different network architecture.

Section 3.1 introduces the raw data as well as its pre-processing. Section 3.2 presents five different loss function from both mathematical and intuitive perspectives. Section 3.3 describes a new network architecture for frontal facial synthesis task, which is Pair-GAN. Experimental setup is shown in Section 3.4 and Section 3.5, including software and hard platforms.

3.1 Dataset and Pre-processing

Color FERET Database from NIST (Facial Recognition Technology, National Institute of Standards and Technology USA) and Multi-PIE Face Database from CMU (Pose and Illumination Images, Carnegie Mellon University) are two widely used datasets in various research on frontal facial generation. Since the former one is free to access within research purpose, we select the color FERET Database in this project.

Raw Data and Pre-processing

![Sample of color FERET Database](image)

**Figure 3.1:** Sample of color FERET Database. From left to right is: frontal face, left pose face at 67.5 angle, right pose face at 67.5 angle, left pose face at 90 angle, right pose face at 90 angle
Color FERET Database is the updated version of FERET Database, which contains 11,338 facial images from 994 different people, collected from 14 different times between December 1993 and August 1996 [Patricia.flanagan@nist.gov, 2019]. There are at least five different poses of each person provided in this database, which is frontal (0 angles), 67.5 angles at left and right posture, and 90 angles at left and right pose (figure 3.1).

Although color FERET dataset satisfies our purpose from the size and angle perspectives, it still exits a significant issue that the raw data cannot be directly input into model training. Since images were taken from 14 different times, the focus length and position of the camera is different in each time, which causes a various facial size (as shown in figure 3.1).

In order to minimize the influence of this noise to GAN training, MTCNN is used to crop out the face for each image (Multi-task Cascaded Convolutional Neural Networks for Face Detection). In the meanwhile, the images’ dimensions are reduced from 512 x 768 pixels into 512 x 512 pixels as well. The comparison between the image before and after cropping is displayed in figure 3.1. Considered that some side-pose images may fail in being detected in the facial position by MTCNN, thus these images are cropped based on the area derived from their frontal image.

Another bias caused by the dataset is the angle of side-pose images; it is commonly agreed that synthesizing the frontal image based on the side-image at a small angle is more accessible than the large-angle since the former one is able to provide more facial features. Therefore, we added another constraint to frontal facial generation in our experiments, which requires the angle of side-pose images is larger than 60 degree. On the other hand, we also restrict the maximum angle of fewer than 90 degrees as models cannot gain any facial information at this degree except for the nose.

![Raw Data](image1.png) ![Cropped Images](image2.png)

**Figure 3.2:** Comparison between raw data from FERET and pre-processed images. Top: Raw data of same person but taken in two different time (940928; 940620); Below: cropped images by MTCNN
§3.2 Synthesis Loss Function

### 3.2.1 Experimental setup

Based on the train-set and test-set, we perform our first stage of experiments, which exploring five different loss functions on Pix2Pix model (Pixel to Pixel Generative Adversarial Networks). The reason behind choosing this model is that Pix2Pix and CycleGAN are two state-of-art models in style transformation where frontal facial synthesis also belongs. Additionally, the Pix2Pix model is more applicable to experiment with different loss functions compared to the CycleGAN.

The generator in Pix2Pix is a modified U-Net which contains eight blocks in the encoder and eight blocks in the decoder. Each block in the encoder is constructed by one convolution layer, one batch normalization layer, and Leak ReLU as the activation function. Each block in the decoder consists of one transposed convolution layer, one batch normalization layer, and ReLU as the activation function. Additionally, the dropout is also applied to the first three blocks of the decoder.

![U-Net architecture used in Pix2Pix](image)

**Figure 3.3:** U-Net architecture used in Pix2Pix. Image is from [Isola et al., 2017]

### 3.2.2 Overview of four loss functions

Overview of 4 different loss functions is shown in figure 3.4. X and Y in the figure are the input image (side-pose facial image) and the generated frontal image. Besides, it is to be observed that the ground identity is generated by applying the generator to the ground-truth image, which aims to add identity loss into gradient descent.
3.2.3 Adversarial Loss

Adversarial loss aims to optimize the generator, which achieves a minimum Kullback-Leibler divergence (KL-divergence) between the generated data and ground-truth data [Goodfellow et al., 2014]. From the formula perspective, the generator tries to get a higher score from the discriminator. In other words, the generated image should be a sharp frontal facial image whose facial feature is closer to the input person.

$$\text{Adversarial Loss} = E_{z \sim P(z)}[\log(D(G(z)))]$$  \hspace{1cm} (8)

where $z$ refers to generated data by the generator; $D$ refers to the discriminator in Pix2Pix model; $G$ refers to the generator in Pix2Pix mode.

However, in practice, it commonly appears that the discriminator in the conditional adversarial network only focuses on whether the generated image is sharp and contains basic facial features or not, ignoring the input of the domain. Based on this, we continually explore other loss functions.

3.2.4 L1 Loss (Mean Absolute Error)

As described in the background of Pix2Pix, the generator may deceive the discriminator by generating a realistic image but unrelated to the input domain. Mean absolute loss (L1 loss) is employed to facilitate content consistency between the generated frontal image and the ground-truth frontal image. Additionally, L1 loss is also able to accelerate optimization in CGAN [Huang et al., 2017b].

$$\text{L1 Loss} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} | GT_{i, j} - GI_{i, j} |$$  \hspace{1cm} (9)

where $W$, $H$ is the width and height of the images; $i$, $j$ is the x coordinate and y coordinate of that pixel; $GT$ refers the ground truth image; $GI$ refers to the generated image.

3.2.5 Gradient Difference loss

Gradient Difference Loss (GDL loss) is proposed by Mathieu et al. [2015], which aims to penalize the differences of image gradient predictions directly. Since the facial
images usually have continuous value within the neighbourhood, this formula can strengthen this relationship in the generated image, making it more realistic.

$$GDL\ Loss = \sum_{i, j} ||Y_{i, j} - Y_{i-1, j}|| - |X_{i, j} - X_{i-1, j}|^\alpha + ||Y_{i, j} - Y_{i, j-1}|| - |X_{i, j} - X_{i, j-1}|^\alpha$$  \hspace{1cm} (10)

where $X$, $Y$ represents the generated image and ground truth image respectively; $i, j$ is the x coordinate and y coordinate of that pixel; $\alpha$ can be any integer greater or equal to 1, we test $\alpha = 1$ and $\alpha = 2$ in our experiments.

Figure 3.6: intuitive view of GDL loss calculation between the ground truth image and the generated image

### 3.2.6 Symmetry Loss

Symmetry is one of the features in human faces. Based on this phenomenon, [Huang et al., 2017b] proposed the symmetry loss, which encourages symmetrical structure generated by the generator. Additionally, this formula behaves more robust in the Laplacian image space as it can overcome the illumination difference between two sides of faces. However, we only apply it to the original pixel image, because the illumination difference is minor in color FERET dataset.

$$Symmetry\ Loss = \frac{1}{W/2 \times H} \sum_{i=1}^{W/2} \sum_{j=1}^{H} |GI_{i, j} - GI_{W-(i-1), j}|$$  \hspace{1cm} (11)
Experimental Methodology and Setup

where W, H is the width and height of the images; i, j is the x coordinate and y coordinate of that pixel; GI refers to the generated image.

![Generated Image](image)

Figure 3.7: intuitive view of symmetry loss calculation inside the generated image

### 3.2.7 Identity Loss

Identity loss firstly appears in the implementation of CycleGAN but without explanation [Zhu et al., 2017]. One intention of this function is to regularize the generator, which should not map the input image into a different domain image if it is already in the target domain. Apart from that, in our experiment, we also found out that this loss function helps to preserve facial identity information from different people.

In terms of the formula, identity loss is the same as mean absolute error but calculates different images.

\[
\text{Identity Loss} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} | GT_{i,j} - GI_{i,j} | \tag{12}
\]

where W, H is the width and height of the images; i, j is the x coordinate and y coordinate of that pixel; GT refers the ground truth image; GII refers to the ground identity image.

![Ground Truth and Ground Identity](image)

Figure 3.8: intuitive view of identity loss calculation between the ground truth image and the generated image

### 3.3 Network Architecture: Pair-GAN

During the experiments, we found out Pix2Pix model and Cycle-GAN have outstanding achievement in general pixel-to-pixel transformation but gain ordinary performance on frontal facial synthesis. Notably, it is hard to balance the ratio of adversarial
loss and other loss. If the proportion of adversarial loss is adjusted higher than other, the generator can easily deceive the discriminator by generating a realistic image but unrelated to the input domain. On the contrary, partial of the generated image is close to the original person, but the entire image is not a real face. For example, there contain three eyes in the face.

With this motivation, we study on the network architecture, which eventually proposes Pair-GAN based on the network architecture of Pix2Pix and Cycle-GAN.

3.3.1 Generator

The generator of Pair-GAN is based on U-Net which is also used in the generator of Pix2Pix and Cycle-GAN [Isola et al., 2017; Zhu et al., 2017]. Unlike usual conditional adversarial networks, there are two U-Nets in the generator, and which are connected by weight sharing. The layered architecture of U-Nets are the same in figure 3.3, which contains eight blocks in the encoder, eight blocks in the decoder, and skip connection between the encoder and decoder. Specifically, the detail layers used in each block of encoder and decoder is the same as Pix2Pix model, except for the normalization layer. In Pair-GAN, we use instance normalization instead of batch normalization, which is able to boost the performance of generation.

When training, Pair-GAN requires two side-pose images at the same angles from the same person but in a different direction as inputs \((X_{\text{left}}, X_{\text{right}})\). Left generator \(G_1\) takes the side-pose image at the left direction, while another side-pose image will be passed into generator \(G_2\). After facial features being extracted by different encoders, two frontal facial images are generated through two decoders. In terms of prediction, since these two generators are independent, only one side-pose image can be input
24

Experimental Methodology and Setup

to the network.

The intention to include two separate networks in the generator is that one single auto-encoder is not affordable to the big style transformation, especially from two different domains, such as left pose and right pose. Besides, frontal facial generation is also considered as one of the difficult tasks in pixel-to-pixel transformation since the spatial relationship of human faces is complex. This idea is not first to be employed in designing the generator of CGAN [Liu and Tuzel, 2016; Huang et al., 2017b; Anoosheh et al., 2018]. Unlike their purposed network which mixes up the encoder and decoder from two networks, Pair-GAN has two independent auto-encoder and sharing the weights of the first two-layer of the decoder.

3.3.2 Discriminator

As described in the background (section 2.1.2), a usual discriminator in generative adversarial network maps the input image into a scalar value which classifies whether it is real or fake. It is to be observed that the output scalar value is a weighted value of the whole data field, which cannot reflect the characteristics of the local feature. Therefore, a normal discriminator is not affordable for pixel-to-pixel transmission task that requires high accuracy.

Figure 3.10: intuitive view of the discriminator in Pair-GAN (Patch-based discriminator of GANs Isola et al. [2017])

Based on this motivation, we employ the patch-based discriminator of GANs (PatchGAN) as the discriminator of Pair-GAN, which was originally proposed in Pix2Pix Isola et al. [2017]. The main idea of the PatchGAN is to map the input image (256 by 256 pixel) to an N-by-N matrix of outputs $X$, where each $X_{i,j}$ represents whether the patch $(i, j)$ is real or fake. Additionally, the patch $(i, j)$ is the convolution result of one receptive field which the discriminator is sensitive to. For example, the classifier (discriminator) might be interested in the ear, tooth, and eyes if the input image is a human face (figure 3.10).
The operation described above is mathematically equivalent to crop the input data into multiple overlapping patches, respectively discriminating the difference by the classifier (discriminator), and averaging the obtained results. From the experimental result indicated in CycleGAN, PatchGAN is beneficial for extracting the local characterization of the images, which is conducive to generate the images in a high resolution.

Due to the overlapping and local characteristics of patches, the size of the receptive field is varied in different tasks. Specifically, a larger scope of receptive field focuses on the relationship between objects that are in a large area, in meanwhile, it also consumes more expensive computation power. From the experimental results offered by Pix2Pix model (figure 3.11, we can notice that the receptive field with 16-by-16 size already achieves a sharp output, whereas the receptive field provides the result with more colourfulness at 70 by 70. As a result, we settle the size of the receptive field is 70 by 70, which results in the output matrix is at 30 by 30.

![Figure 3.11: Experiments on different size of receptive field. Reference to Pix2Pix model [Isola et al., 2017]](image)

### 3.3.3 Loss Functions

#### Pair Loss

In order to reduce the difference of generated image between the left generator and right generator, we not only consider the weight-sharing between two encoders, but also pair loss is added into the loss function. This loss is expected to penalize the network parameters if there exists a major difference between generated images by the left generator and right generator. The formula of pair loss is the same as the mean absolute error but calculates different images.

\[
    \text{Identity Loss} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} | GIL_{i,j} - GIR_{i,j} | 
\]

where \( W, H \) is the width and height of the images; \( i, j \) is the x coordinate and y coordinate of that pixel; \( GIL \) refers the generated image by the left generator; \( GIR \) refers to the generated image by the right generator.
**Figure 3.12:** intuitive view of pair loss calculation between the left generated image and right generated image

**Figure 3.13:** Overview of loss functions explored in PairGAN
Overall Loss Function

A weighted sum of all losses discussed above constructs the final synthesis loss function of PairGAN:

\[
L_{\text{left generator}} = \lambda_1 \times L_{\text{adversarial}} + \lambda_2 \times L_1 + \lambda_3 \times L_{\text{GDL}} + \lambda_4 \times L_{\text{sym}} + \lambda_5 \times L_{\text{id}} + \lambda_6 \times L_{\text{pair}}
\]

\[
L_{\text{right generator}} = \lambda_7 \times L_{\text{adversarial}} + \lambda_8 \times L_1 + \lambda_9 \times L_{\text{GDL}} + \lambda_{10} \times L_{\text{sym}} + \lambda_{11} \times L_{\text{id}} + \lambda_6 \times L_{\text{pair}}
\]

where \( \lambda_\ast \) means the weight of that loss.

3.4 Software platform

1. Data pre-processing: the detail package requirement is listed in table 3.1.

2. Loss function and network architecture experiment: we use TensorFlow as our coding template, and the package requirement is listed in table 3.2

Additionally, all experiments were built and raw in Ubuntu 19.0.

3.5 Hardware platform

In order to gain a better computation power, we employ GPU to perform our experiments. The platform’s description can be found in table 3.3. The computation ability of GPU can be considered at table 3.4.

<table>
<thead>
<tr>
<th>Packages</th>
<th>Python</th>
<th>Matplotlib</th>
<th>Numpy</th>
<th>opencv (cv2)</th>
<th>mtcnn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>&gt;=3.6</td>
<td>&gt;=3.1.3</td>
<td>&gt;=1.18.1</td>
<td>4.2.0</td>
<td>0.1.0</td>
</tr>
</tbody>
</table>

Table 3.2: Python packages used in loss function and network architecture experiment

<table>
<thead>
<tr>
<th>Packages</th>
<th>python</th>
<th>tensorflow</th>
<th>tensorboard</th>
<th>keras</th>
<th>opencv (cv2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>3.7.6</td>
<td>2.1.0</td>
<td>2.1.0</td>
<td>2.3.1</td>
<td>4.2.0</td>
</tr>
</tbody>
</table>
### Table 3.3: Hardware platform details

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>1 x NVIDIA RTX 2060 Super with 8GB graphic memory</td>
</tr>
<tr>
<td>CPU</td>
<td>1 x Intel Core i7-8086K with Frequency at 5.00 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>4 x 16GB</td>
</tr>
</tbody>
</table>

### Table 3.4: GPU Computation ability

<table>
<thead>
<tr>
<th>Model</th>
<th>Time to Train One Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pix2Pix</td>
<td>128 seconds</td>
</tr>
<tr>
<td>Cycle-GAN</td>
<td>315 Seconds</td>
</tr>
<tr>
<td>Pair-GAN</td>
<td>224 Seconds</td>
</tr>
</tbody>
</table>
In Chapter 4, we evaluated the results obtained from experiments on different loss function and Pair-GAN, and drew some conclusions from our observation. Besides, the detailed hyper-parameters of each experiment was also included.

In section 4.1, we discussed the result of frontal facial synthesis by using Pix2Pix and CycleGAN, which provides a comparison to other experiments. Section 3.2 covered the performance and evaluation of different loss configuration on Pix2Pix model. The analysis of our proposed new network architecture to frontal facial generation task Pair-GAN was discussed in Section 4.3.

In order to provide convincing results or evaluation, we also employ the facial comparison to derive the similarity (confidence) between frontal facial synthesis and ground-truth image. Additionally, the result is provided by a commercial face comparison API (Face ++). Besides, the average similarity listed below is calculated from 40 pair images which are picked from test-set, due to the access limitation to that API.

![Figure 4.1: Examples of facial comparison by using Face++](image)

### 4.1 State-of-Art Models in Frontal Facial Synthesis Analysis

The first stage of overall experiments is to study current state-of-art models in frontal facial synthesis, which is also treated as a baseline to compare our methods. As discussed in the last section, we employed the Pix2Pix model and CycleGAN to perform frontal facial generation on pre-processed color FERET database. The default
Results and Evaluation

Figure 4.2: Average similarity, best similarity, and worst similarity between ground truth and generated image by Pix2Pix model and Cycle-GAN with default loss ratio and train epoch is 125. Higher similarity is better.

loss function used in Pix2Pix is adversarial loss and L1 loss with the ratio at 1 of 120. In contrast, the CycleGAN uses adversarial loss, cycle consistent loss and identity loss with the ratio at 1 of 5 of 10. Other hype-parameters of these two models were left as default, which optimized with Adam and a learning rate of 0.0002. Additionally, both Pix2Pix and CycleGAN was trained with 125 epoch which one epoch takes 2,090 pairs of images. The result of the similarity between ground truth and generated image by these two models is shown in figure 4.2, which we can conclude that the performance of CycleGAN is around ten percents better than Pix2Pix model in average. Additionally, all generated frontal faces by CycleGAN can be recognized, whereas there exits 0 confidence in synthesis by Pix2Pix model.

In order to further analyze the generated frontal facial from human perspectives, the best results achieved by Pix2Pix and CycleGAN are displayed in Figure 4.3. As we can notice, two model focus on contrasting facial features when generating the frontal image based on the side-pose image. Pix2Pix one can provide a more realistic hairstyle and precise nick detail, whereas eyes, mouth and nose are the spotlights of CycleGAN. Although these two models are able to generate a highly similar facial image to the original person, the precise texture of face is mainly not indicated in the final output like wrinkle, which results in a negative influence to our judgment their similarity.

Figure 4.3: Best result derived from of Pix2Pix model and CycleGAN with default loss ratio and train epoch is 125
We also explored the influence of different epoch on the generated results, which is indicated in figure 4.4. It demonstrates that "125 epochs" is a reasonable configuration to the model training since the confidence is increased from "60 epochs" to "125 epochs". On the contrary, the performance measured by best confidence and worst confidence decreased massively after 200 epochs. As a result, epoch configuration is settled down by "125 epochs" in further experiments.

![Figure 4.4: Compare similarity between ground truth and generated image by Cycle-GAN on different epochs but with default loss ratio. Higher similarity is better](image)

### §4.2 Loss Function Analysis

**Adversarial Loss, L1 Loss, Gradient Difference Loss**

The similarity result of the experiment on adjusting different ratios of loss functions was shown in figure 4.5. Based on the default ratio of adversarial loss and L1 loss in Pix2Pix model (1:120), we firstly adjusted the proportion of the L1 loss. With a low penalty coming from L1 loss, Pix2Pix trends to generate the frontal facial less similar to the original domain, which matches the theoretical analysis to the functionality of mean absolute error. If the L1 loss is removed from the loss penalty, the average similarity drops down 40 per cent than the default loss ratio of Pix2Pix model. On the other hand, although a large L1 penalty results in a highly similar image, the blurring issue of synthetic frontal images become more serious. As it is shown in figure 4.7, the shape of the nose is difficult to determine in the generated image by 120 L1 loss, compared to the adversarial loss one. The reason behind it is that the mean absolute loss guides the gradient descent of GAN into averaging the value between output and ground truth, which is same as smearing color.

After applying the GDL (gradient difference loss) into Pix2Pix model with keeping the original ratio, the average similarity remains unchanged. In order to analyze the influence of GDL penalty to model training, the proportion of L1 loss is decreased below than the adversarial loss; the ratio between L1 and GAL keeps untouched at the same time. We can notice that the average similarity grows to 38 among 40 test images. However, the obscured side of the input image is blurred in the synthesis (figure 4.7), which considered as a drawback of the GDL penalty.
Results and Evaluation

Figure 4.5: Compare average similarity between ground truth and generated image on different loss configurations among 40 test images. Two brown color bars are the state-of-art model (Pix2Pix, CycleGAN) mentioned in the above section. Higher similarity is better.

Symmetric Loss

The motivation to add the symmetric penalty to the model training is symmetry one of the characteristics in human faces. However, the results are beyond our expectation, which results in a negative influence on the performance of Pix2Pix as the average similarity decreases into 31 among 40 test images. The direct reason causes above phenomenon are that the symmetry loss is hard to converge during the training progress (as it is shown in figure 4.6). As a result, the symmetric loss is abandoned in the following experiments.

![Symmetric Loss Graph](image)

Figure 4.6: Monitoring changes of symmetric loss during whole training progress. The plot is generated by TensorBoard

Identity Loss

The final experiment on loss function, we explore the identity loss which aims to regularize the generator. Compared to the L1 penalty and GDL penalty, identity loss leads to a significant improvement on the frontal facial of Pix2Pix model, which the synthetic frontal face is most natural and sharp compared others (figure 4.7). Notably, hairstyle and glasses can be recovered in high quality. As a result, identity penalty helps to preserve facial identity information which is various from different people. On the other hand, adding identity loss to GAN learning will double the training
time in each epoch since it needs to calculate the ground identity image through the generator.

![Image of experiment results](image)

Figure 4.7: Experiment of comparing different loss configuration

### 4.3 Pair-GAN Analysis

The final experiment on loss function, we explore the identity loss which aims to regularize the generator. Compared to the L1 penalty and GDL penalty, identity loss leads to a significant improvement on the frontal facial of Pix2Pix model, which the synthetic frontal face is most natural and sharp compared others (figure 4.7). Notably, hairstyle and glasses can be recovered in high quality. As a result, identity penalty helps to preserve facial identity information which is various from different people. On the other hand, adding identity loss to GAN learning will double the training time in each epoch since it needs to calculate the ground identity image through the generator.

**Results**

Pair loss and weight sharing are the two significant hyper-parameters for Pair-GAN since they are designed to reduce the distinction of generated images of the same person by the left generator and the right generator. We expected the network architecture with weight sharing gains a better performance than using the loss penalty as the former one is close to a coercive specification. However, the result indicates
another opinion which the pair loss helps to improve the average performance, and weight sharing contributes to refresh the peak performance (Table 4.1). If both weight sharing and pair loss are employed in the Pair-GAN, it will decrease both peak and average performance, which results from the network is hard to converge.

Table 4.1: Results of different configurations on PairGAN. Where Adv means the proportion of adversarial loss; L1 means the proportion of L1 loss; GDL means the proportion of gradient difference loss; Id means the proportion of identity loss; Pair means the proportion of pair loss;

<table>
<thead>
<tr>
<th>Adv</th>
<th>L1</th>
<th>GDL</th>
<th>Id</th>
<th>Pair</th>
<th>Weight Share</th>
<th>Avg Sim</th>
<th>Max Sim</th>
<th>Min Sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>Disable</td>
<td>44.23</td>
<td>79.14</td>
<td>22.02</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>10</td>
<td>Disable</td>
<td>44.30</td>
<td>74.22</td>
<td>23.56</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>Enable</td>
<td>42.34</td>
<td>81.83</td>
<td>17.16</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10-5</td>
<td>10-2</td>
<td>Enable</td>
<td>42.79</td>
<td>70.61</td>
<td>16.43</td>
</tr>
</tbody>
</table>

where $\lambda_1$, $\lambda_7$ in formula 14, 15 refers to the weights of Adv; $\lambda_2$, $\lambda_8$ in formula 14, 15 refers to the weights of L1; $\lambda_3$, $\lambda_9$ in formula 14, 15 refers to the weights of GDL; $\lambda_4$, $\lambda_{10}$ in formula 14, 15 is set to 0; $\lambda_5$, $\lambda_{11}$ in formula 14, 15 refers to the weights of Id; $\lambda_6$ in formula 14, 15 refers to the weights of Pair;

Comparing with State-of-Art

Figure 4.8: Compare PairGAN’s performance with Pix2Pix and CycleGAN

In order to further analyze the performance of Pair-GAN, we include comparisons with the state-of-art models in this section. It is worth noticing that Pose-Invariant adversarial generative network (FT-GAN) is the state-of-art model in frontal facial generation [Zhuang et al., 2019], but they used CMU Multi-PIE database and also did not release the codes. Therefore, we compare our model with CycleGAN since the FT-GAN is built on CycleGAN and within 5 % better than it.

As it is shown on figure 4.8, the performance of Pair-GAN is around 13 % better than the CycleGAN and 25 % than the Pix2Pix model, measured by the average similarity of synthesis and ground truth among 40 test images. From the synthetic frontal images in 4.9, we can noticed that CycleGAN only focus and generate the
main facial features, and stacks onto the original side-pose image. On the contrary, the field consider to be recovered is broad on Pair-GAN; it not only focuses on the facial features, but also considers the other elements of facial images, such as hairstyle and neck.

In conclusion, the results of conducted experiments indicate the Pair-GAN wins a better performance than the Cycle-GAN in color FERET database. However, network architecture still required to be further developed. For instance, although pair loss and weight sharing are applied to minimize the difference of generated images by the two generators of Pair-GAN, this issue still has not been adequately resolved, which can be noticed in first two columns and last two columns of figure 4.9.

Figure 4.9: Compare generated images by CycleGAN and PairGAN
Conclusion

Based on the motivation of the dilemma of three-dimensional reconstruction, we raised the topic on synthesizing frontal face based on side-pose facial images in this work. At the beginning stage, we focus on exploring the pixel-to-pixel transformation on frontal facial generation, which finds Pix2Pix and CycleGAN models. To further develop the performance of Pix2Pix model, five different loss functions are tested and analyzed, including adversarial loss, L1 loss, gradient difference loss, symmetry loss, and identity loss. Since the improvement based on different loss penalty is minor, we continue to study the network architecture, which proposed the Pair-GAN based on the Cycle-GAN.

Through the analysis of experimental results on different loss function, we concluded that L1 loss, GDL loss, and identity loss are helpful to alleviate the common issue existing in CGAN, which the output face is unrelated to the input face. Additionally, the ratio of these four-loss, achieving the highest score in our experiment is 20 : 3 : 0.1 : 5 (adversarial : L1 : GDL : identity). On the other hand, the symmetric loss is challenging to converge in the experiments, which causes negative influences on the performance.

Furthermore, we obtained the result from the second experiment that Pair-GAN is proven to generate better results of frontal face than the Cycle-GAN in color FERET database; specifically, it improves around 13 % at average compared to the CycleGAN. Apart from that, we also explored various configuration of Pair-GAN, which results in either pair loss (soft penalty) or weight sharing (coercive specification) can positively contribute to, but not both.

5.1 Limitation and Future Work

We are aware that there are a few limitations in our experiments and several possible future works, which included the below:

1. The similarity results between the synthetic frontal face and ground truth are derived from 40 test images, which may result in a limitation to reflect the model performance.

2. We only employ one facial comparison API to measure the experimental results
quantitatively. As suggested in Huang et al. [2017b], they also test the identity preserving property of synthetic face in a gender perspective.

3. Add comparison between the Pair-GAN and the state-of-art model (FT-GAN). Since they didn’t release the codes and pre-trained model, we cannot recreate it from scratch in a short project time-frame.

4. Due to the limitation of time and computation, the hyper-parameters selected for Pair-GAN were not optimised in terms of performance.

5. Further tuning the Pair-GAN model by using a larger database (CMU multi-PIE) to improve its performance.

6. Add a classifier at the beginning of Pair-GAN’s generator, which aims to classify the input image is left or right. This procedure will make the result more convincing when compared to other models.
Bibliography


INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

<table>
<thead>
<tr>
<th>UniID:</th>
<th>u6212207</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAMILY NAME:</td>
<td>Shen</td>
</tr>
<tr>
<td>PERSONAL NAME(S):</td>
<td>Xuyang</td>
</tr>
<tr>
<td>PROJECT SUPERVISOR (may be external):</td>
<td>Jo Plested</td>
</tr>
<tr>
<td>COURSE SUPERVISOR (a RSCS academic):</td>
<td>Tom Gedein</td>
</tr>
<tr>
<td>COURSE CODE, TITLE AND UNIT:</td>
<td>COMP3740 Project Work in Computing 6 units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEMESTER</th>
<th>S2</th>
<th>YEAR:</th>
<th>2019</th>
<th></th>
<th>S1</th>
<th>YEAR:</th>
<th>2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT TITLE:</td>
<td>3D Facial Reconstruction – inferring missing face sections from face poses at an angle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LEARNING OBJECTIVES:
• Experience with deep learning techniques including Conditional Generative Adversarial networks.
• Experience with facial generation techniques.
• Experience with techniques for inferring missing face sections from face poses at an angle.

PROJECT DESCRIPTION:
• Construct a literature review on current state of the art and possible new methods for 3D facial reconstruction.
• Design and implement experiments to compare existing state of the art including potential improvements to infer missing face sections from face poses at an angle.
• Thoroughly test and evaluate comparisons.
• Write project report
ASSESSMENT (as per course’s project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th></th>
<th>% of mark</th>
<th></th>
<th>% of mark</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Honours (24 credit)</strong></td>
<td>(fixed)</td>
<td><strong>Projects (6-12 credit)</strong></td>
<td>…. / (fixed)</td>
</tr>
<tr>
<td>Assessed project components:</td>
<td>(85%)</td>
<td>Thesis (reviewer mark)</td>
<td>....45.....45-60%</td>
</tr>
<tr>
<td>Thesis</td>
<td>(85%)</td>
<td>Artefact (supervisor project mark)</td>
<td>....45.....30-45%</td>
</tr>
<tr>
<td>Presentation</td>
<td>(10%)</td>
<td>Presentation</td>
<td>(10%)</td>
</tr>
<tr>
<td>Critical Feedback</td>
<td>(5%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

Weekly

STUDENT DECLARATION: I agree to fulfil the above defined contract:

_________________________                     _____________
Signature                 Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

_________________________                     _____________
Signature                 Date

Reviewer:

Name: __________________________                     Signature: __________________________

Reviewer 2: (for Honours only)

Name: __________________________                     Signature: __________________________

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Honours / Projects coordinator approval)

_________________________                     _____________
Signature                 Date
Data Pre-processing

Files Structure:

data_preprocessing
  |   pre_processing.py:
  |       provides 4 functionalities:
  |          1. data profiling
  |          2. clean the object which doesn't have these poses [left, right, frontal]
  |          3. auto-crop, only keeps facial part
  |          4. generate train and test
  |  posefilters.py: detail implementation of the "data profiling" functionality
  |  autocrop.py: detail implementation of the "auto-crop" functionality
  |  merge.py: detail implementation of the "generate train and test" functionality

Package Requirement

This software has only been tested on Ubuntu 19.04 and MacOS 10.15.

Please install all the required package in your environment:

- Python >= 3.6
- Matplotlib >= 3.13
- Numpy >= 1.18.1
- opencv(cv2) = 4.2.0
- mtcnn = 0.1.0 (including Keras >= 2.0.0)

Run Programs

1. Prepare the Raw Data:
   - Either follow the instruction from NIST: https://www.nist.gov/itl/products-and-services/color-feret-database
2. **Training commands:**

```bash
$ python data_preprocessing/pre_processing.py -h

Data Pre-processing For color FERET database

optional arguments:
  -h, --help       show this help message and exit
  -p --path        raw dataset path (default: ./RawData)
  -t --task        task needs to run.
                   0: data profiling + cropped image + generate train-test;
                   1: data profiling + cropped image;
                   2: data profiling
  -m --model       select model, pair_gan, pix2pix, cyclegan (default: pair_gan)
  -rc --rmcache    whether remove cache or not (default: False)
  -r --rate        train and test split rate (default: 0.9)
```

**Quick Commands**

```bash
# run data profiling for raw data
$ python data_preprocessing/pre_processing.py --task 2

# crop image for raw data
$ python data_preprocessing/pre_processing.py --task 1

# generate train and test set for pair_gan model
$ python data_preprocessing/pre_processing.py

# generate train and test set for either pix2pix model or cyclegan
$ python data_preprocessing/pre_processing.py --model normal
```

**Pix2Pix Loss Function Explore**

*Code build on tensorflow official tutorial of Pix2Pix: [https://www.tensorflow.org/tutorials/generative/pix2pix](https://www.tensorflow.org/tutorials/generative/pix2pix)*
Files Structure:

```
Pix2Pix
|-- train.py:
   main file to run the training
   including: model init, loss config, optimizer, and etc
|-- vice_functions.py: contain several auxiliary function for train.py
|-- data_input.py: read train and test images from file
|-- pix2pix.py: store generator and discriminator of Pix2Pix
|-- /plot_saving: store the generated image during training progress
|-- /models_saving: store the saved model
```

Package Requirement

This software has only been tested on Ubuntu19.04, Python 3.7.6, Cuda-10.1.105 with a RTX-2060 Super 8Gb graphic memory, and 64 Gb system memory

Please install all the required package in your environment:

- Python >= 3.7
- tensorflow = 2.1.0
- tensor board = 2.1.0
- keras = 2.3.1
- opencv (cv2) = 4.2.0

Run Programs

1. **Prepare the Raw Data:**
   - Either download from our shared folder in google-drive: [https://drive.google.com/file/d/1vpYrD2v3EpltkqGpmJfWfECQrQpR7mXc/view?usp=sharing](https://drive.google.com/file/d/1vpYrD2v3EpltkqGpmJfWfECQrQpR7mXc/view?usp=sharing)
     around 288 Mb
   - Or follow instructions in data pre-processing to generate train-test set

2. **Training commands:**

   ```bash
   $ python Pix2Pix/train.py -h
   usage: train.py [-h] [-p --path] [-ep --epoch] [-r --recover-training]
   ```
Quick Commands

[-al -adv_loss] [-l1l -l1_loss] [-gl -gdl_loss]
[-sl -sym_loss] [-il -identity_loss]

Loss Function Exploration For Pix2Pix

optional arguments:
  -h, --help            show this help message and exit
  -p --path             dataset path (default: ./data1/)
  -ep --epoch           train epoch (default: 125)
  -r --recover-training continue training from saved model (default: False)

  -al -adv_loss         set weight for adversarial loss
  -l1l -l1_loss         set weight for l1 loss
  -gl -gdl_loss         set weight for gdl loss
  -sl -sym_loss         set weight for symmetric loss
  -il -identity_loss    set weight for identity loss

# run pix2pix training with all default setting
# default: 20 * adversarial loss + 3 * l1 loss + 0.1 * gradient difference loss + 0.05 * symmetric loss + 0 identity loss
$ python Pix2Pix/train.py

# run pix2pix training with other weights for loss functions
$ python Pix2Pix/train.py -al 20 -l1l 3 -gl 0.1 -sl 0.05 -il 0

# resume training from saved model
# please put the saved model under "models_saving" folder
$ python Pix2Pix/train.py -r True

Suggested Training Time

- Disable Identity Loss: 125 seconds / epoch
- Enable Identity Loss: 210 seconds / epoch

Pair-GAN Explore

Files Structure:

```
Pair_GAN
|-- train.py: main file to run the training including: model init, loss config, optimizer, and etc
|-- predict.py: predict images based on the saved model
|-- vice_functions.py: contain several auxiliary function for train.py, predict.py
|-- data_input.py: read train and test images from file
|-- pair_gan.py: store 2 generator and discriminator of Pix2Pix, including weight sharing
|-- /plot_saving: store the generated image during training progress
|-- /models_saving: store the saved model
|-- /logs: monitor the loss change during training progress
```

Package Requirement
This software has only been tested on Ubuntu19.04, Python 3.7.6, Cuda-10.1.105 with a RTX-2060 Super 8Gb graphic memory, and 64 Gb system memory

Please install all the required package in your environment:

- Python >= 3.7
- tensorflow = 2.1.0
- tensor board = 2.1.0
- keras = 2.3.1
- opencv (cv2) = 4.2.0

Run Programs

1. **Prepare the Raw Data: different from pix2pix model and cyclegan**
   - Either download from our shared folder in google-drive: [https://drive.google.com/file/d/1CG7uly-hGna1d1AksnCQblAhX58l7Z_u/view?usp=sharing](https://drive.google.com/file/d/1CG7uly-hGna1d1AksnCQblAhX58l7Z_u/view?usp=sharing)
   - around 217 Mb
2. **Training commands:**

```bash
$ python Pair_GAN/train.py -h
usage: train.py [-h] [-p --path] [-ep --epoch] [-r --recover-training]
[-al -adv_loss] [-l1l -l1_loss] [-gl -gdl_loss]
[-sl -sym_loss] [-il -identity_loss]
```

Model Training For Pair-GAN

optional arguments:

- **-h, --help**: show this help message and `exit`
- **-p --path**: dataset path (default: ./data2/)
- **-ep --epoch**: train epoch (default: 250)
- **-r --recover-training**: continue training from saved model (default: False)
- **-al -adv_loss**: set weight for adversarial loss
- **-l1l -l1_loss**: set weight for l1 loss
- **-gl -gdl_loss**: set weight for gdl loss
- **-sl -sym_loss**: set weight for symmetric loss
- **-il -identity_loss**: set weight for identity loss
- **-pl -pair_loss**: set weight for pair loss

# adversarial loss > 0
# l1 loss >= 0; symmetric loss >= 0; identity loss >=0 or -1; pair loss >= 0 or -1
# if set identity loss and pair loss as -1, the weight can be changed depend on epoch.

Quick Commands

# run PairGAN training with all default setting
# default: 10 * adversarial loss + 3 * l1 loss + 0 * gradient difference
loss + 0 * symmetric loss + max(5, min(12.5 - epoch / 25, 10)) * identity loss + max(2, min(46 / 3 - epoch / 15, 10)) * pair loss
$ python Pair_GAN/train.py

# run PairGAN training with other weights for loss functions
$ python Pair_GAN/train.py -al 20 -l1l 3 -gl 0.1 -sl 0.05 -il -1 -pl -1

# resume training from saved model
# please put the saved model under "models_saving" folder
$ python Pair_GAN/train.py -r True
**Suggested Training Time**

- 225 - 250 seconds / epoch

3. **Predict commands:**

```
$ python Pair_GAN/predict.py -h
usage: predict.py [-h] [-p --path] [-mp --model_path]
```

Model Predicting For Pair-GAN

optional arguments:
- -h, --help show this help message and exit
- -p --path dataset path (default: ./data2/)
- -mp --model_path path of saved model (default: ./Pair_GAN/models_saving/)

**Quick Commands**

```
# run PairGAN predict with all default setting
$ python Pair_GAN/predict.py
```

**Suggested Predicting Time**

around 20 FPS

**Pre-trained model**

- Please download from our shared folder in google-drive: [https://drive.google.com/file/d/1NzfufOqDRoBUSdqcuxHhoOeRzhqi4_x/view?usp=sharing](https://drive.google.com/file/d/1NzfufOqDRoBUSdqcuxHhoOeRzhqi4_x/view?usp=sharing)
  
  around 1Gb

**Validation**

**Files Stucture:**
**Package Requirement**

This software has only been tested on Ubuntu 19.04 and MacOS 10.15.

Please install all the required package in your environment:

- opencv(cv2) = 4.2.0

**Run Programs**

**Training commands:**

```bash
$ python Validation/similarity.py -h
usage: similarity.py [-h] [-p --path] [-r --refresh]

optional arguments:
  -h, --help            show this help message and exit
  -p --path             predication path which requires to get similarity
  -r --refresh          refresh the recorder file
```

**Quick Commands**

- # run similarity test on /Pair_GAN/prediction-15437298
  
  $ python Validation/similarity.py -p /Pair_GAN/prediction-15437298

- # clean previous results
  
  $ python Validation/similarity.py -p /Pair_GAN/prediction-15437298 -r True