Abstract
This project aims to achieve facial expression transfer directly between 2D RGB images by deep learning. Facial expression transfer is an operation to extract facial expression from one person, namely source, and generate the extracted expression on the face of another person, namely target, while maintaining the target’s facial identity. In this project, Facial Action Coding System (FACS) [1] is used to encode facial expression in a non-discrete and anatomical accurate manner. A website and an android mobile application were developed to provide service of facial expression transfer. The website and android application were built based on the OpenFace 2.0 toolkit [4] to extract facial expression and the GANimation model implemented by Pumarola et al. [3] to perform facial expression generation.

Acknowledgement
We would like to expression our deep gratitude to our supervisor, Dr. Dirk Schnieders. We changed our project topic and supervisor once in the middle of September. We would like to thank Dr. Dirk Schnieders for his patient and guidance for this project in such a hurry situation. We would also like to thank Miss Yeung Hei Tung for her advice on the user interface design of the android application.
**Table of Contents**

- List of Figures 4
- List of Tables 5
- Abbreviations 6
- 1. Introduction 7
  - 1.1 Objective 7
  - 1.2 Motivation 7
  - 1.3 Scope of Work 8
- 2. Related Work 8
  - 2.1 Facial Action Coding System (FACS) 8
  - 2.2 Facial Action Unit Intensity Estimation 8
  - 2.2 Generative Adversarial Network (GAN) 8
  - 2.3 Facial Expression Generation conditioned by FACS 9
    - 2.3.1 Direct Facial Expression Generation from 2D image to 2D image 9
    - 2.3.2 Facial Expression Generation using 3D Model 9
- 3. Methodology 9
  - 3.1 Facial Expression Representation under FACS 9
  - 3.2 Facial Expression Generation 11
    - 3.2.1 Data Collection 11
    - 3.2.2 Data Preprocessing 12
    - 3.2.3 Training of the GANimation Network 14
  - 3.3 Facial Expression Transfer 15
    - 3.3.1 Demonstration Website 15
    - 3.3.2 RESTful API server 15
  - 3.4 Android Mobile Application Development 16
    - 3.4.1 Android application 16
    - 3.4.2 Refined API server 18
- 4. Results and Limitation 18
  - 4.1 GANimation Training Results 18
  - 4.2 Software Implementation Result 22
    - 4.2.1 Demonstration Website Implementation Result 22
    - 4.2.2 Android Application Implementation Result 23
    - 4.2.3 API Server Implementation Result 24
- 5. Conclusion and Future Work 26
- 6. References 27
List of Figures

Figure 1. Haar features for face detection 12
Figure 2. An example of face detection using haar cascade 12
Figure 3. The structure of the GANimation network. 14
Figure 4. Request handling flow of the API server. 16
Figure 5a. Basic UI design of page for import image to extract facial expression. The import button is marked by label (1). The photo taking button is marked by label (2). The sample image credits to Philips Cavalcante. 17
Figure 5b. Basic UI design of page for import image to transfer facial expression. 17
Figure 5c. Basic UI design of page to perform facial expression transfer. The transfer button is marked by label (1). The export button is marked by label (2). The GIF button is marked by label (3). 17
Figure 6a. Basic UI design of page for AU editing (generation). User can select an AU to edit from the circular scrolling panel. 17
Figure 6b. Basic UI design of page for AU editing (generation). After selecting an AU to edit, a circular scrolling bar is provided for changing the AU value. 17
Figure 7. Request handling flow of the new API server. 18
Figure 8a. Loss values to evaluate the performance of the generator in producing fake image during the Mixed Dataset Training. 19
Figure 8b. Loss values to evaluate the performance of the generator in producing fake image during the EmotioNet Dataset Training 19
Figure 8c. Loss values to evaluate the performance of the discriminator in discriminating fake image during the Mixed Dataset Training. 19
Figure 8d. Loss values to evaluate the performance of the discriminator in discriminating fake image during the Mixed Dataset Training. 19
Figure 9. Single AU edition on AU17, AU23 and AU45 performed by the generator trained by the Mixed Dataset. The changes are not obvious and the edition of AU45 produces a lot of noise. 19
Figure 10. The distribution of AU values for each of the AUs in the Mixed Dataset 20
Figure 11. The distribution of AU values for each of the AUs in the EmotioNet Dataset 20
Figure 12. Single AU edition on AU17, AU23 and AU45 performed by the generator trained by the EmotioNet Dataset. 21
Figure 13. A batch of single AU value edition results 21
Figure 14. Implementation result of the facial expression transfer demonstration webpage. 22
Figure 15. Implementation result of the AU editing demonstration webpage 22
Figure 16. The import pages and the resulting page of the facial expression transfer activity 23
Figure 17. The implementation of AU editing activity. 23
List of Tables

Table 1. A list of used Action Units to represent facial expression in this project. The example images are from the FACS introduction site of the Carnegie Mellon’s School of Computer Science, except for the example image for AU45 which is shared by John R. Hershey on the website of ResearchGate. 10

Table 2. Built-in haar cascade file provided by OpenCV. 13

Table 3. Specification of Env_old used perform training with the mixed dataset. 14

Table 4. Specification of the Env_New used to perform training with the EmotioNet dataset. 15
Abbreviations

API  Application Programming Interface
AU   Action Unit
CDAAE Difference Adversarial Autoencoder
CelebA CelebFaces Attributes Dataset
CK database Cohn-Kanade (CK and CK+) database
FACS  Facial Action Coding System
FERET Facial Recognition Technology program
GAN  Generative adversarial network
GUFD Glasgow Unfamiliar Face Database
HTTPS Hypertext Transfer Protocol Secure
JSON  JavaScript Object Notation
LFW  Labeled Faces in the Wild database
URL  Uniform Resource Locator
1. Introduction

Synthesising realistic facial expressions from 2D images of human face could be beneficial to a wide range of industries including the film industry and the social media industry. Automatic facial expression generation technique could also help the development of other computer vision research areas such as face recognition and facial expression prediction.

Before the success of deep learning, researchers tried to produce new facial expressions by manipulating the facial landmarks with a geometric approach such as triangular geometric deformation [5] and frequency analysis [6], but the results were not satisfactory. However, facial expression generation technology has been substantially improved along with the maturity of deep learning.

In this Introduction section, the objective of this project will first be introduced. After that, motivation of this project will be discussed. Finally, the scope of this project will be described to state the focus of the project.

1.1 Objective

The main objective of this project is to achieve accurate and complex facial expression transfer anatomically by facial expression extraction and facial expression generation under an AU conditioned deep learning scheme, and develop software to provide such service. As the technology of facial expression extraction is mature, facial expression generation is the focus of this project.

1.2 Motivation

In 2014, Goodfellow et al. introduced generative adversarial networks (GANs) which are powerful for generative tasks [2]. Since then, many research studies in using various types of GANs for facial expression generation have been published such as CycleGAN [7], IcGAN [8] and StarGAN [9]. StarGAN made a huge breakthrough in facial expression generation. The synthesized facial expression results have become more realistic and natural. However, StarGAN is trained on the Radboud Faces Database (RaFD) [10] which consists of facial images with binary facial expression labels such as happy, sad, angry, fearful, etc. Due to the property of StarGAN, the generation results are limited to the corresponding set of discrete facial expression labels and interpolation between different facial expression is not application [2]. Therefore, StarGAN is not suitable for complex facial expression generation. Although ExprGAN [11] enables the intensity adjustment of a given set of facial expressions, the synthesized results are still limited to a number of discrete facial expressions.

Because of the above limitation mentioned, the Facial Action Coding System (FACS) [1] has started to gain attention in the area of facial expression synthesis. FACS is an anatomical facial expression measurement system which describes facial expressions using Action Units (AUs) which are independent actions of different sets of facial muscle [1]. The combination of deep learning and FACS makes complex facial expression generation possible. Research studies using GAN, autoencoder with FACS has successfully synthesized complex and realistic facial expressions [3] [12] [13]. Deep learning models conditioned by AU intensity can be applied to achieve facial expression transfer by combining with AU intensity estimation.

By combining the facial expression generation program GANimation [3] and the existing solution of facial expressions detection, the OpenFace 2.0 toolkit [4], facial expression transfer can be performed. Although the source code of the GANimation program and the OpenFace 2.0 toolkit is published on the internet, there is no open source software of facial expression transfer available. Therefore, this project aims to build an open source facial expression transfer software based on the GANimation program and the OpenFace 2.0 toolkit.
1.3 Scope of Work

In this project, direct 2D to 2D image facial expression transfer is studied. Transferring facial expression using 3D model is not in the scope of this project. The work of this project was separated into three stages which are facial expression generation, facial expression transfer and android application development.

The first stage was facial expression generation. The goal of this stage was to train the generative adversarial network called GANimation implemented by the team of Pumarola [3] to perform FACS conditioned facial expression generation. This stage consists of two tasks. The first task is data preprocessing. In order to train the GANimation network, a large amount of facial images was collected from several large scale open source facial image databases. Data cleansing and selection were performed to ensure the quality of data. After that, the data was reformatted and processed to fit the specific requirement of the GANimation network. After data preprocessing, the GANimation network was trained and tested. This stage was performed multiple times throughout the project to improve the result.

The second stage was facial expression transfer. In this stage, the trained generator in the GANimation network was used together with the OpenFace 2.0 toolkit to perform facial expression transfer. The OpenFace 2.0 toolkit was used to extract facial expressions under the FACS scheme. The GANimation generator model was used to generate new facial images with the extracted facial expressions. A website with simple user interface and an API server were developed to demonstrate the functionalities of facial expression generation and facial expression transfer.

The third stage was android mobile application development. In this stage, an android application with user friendly interface was developed. This application allows users to perform facial expression transfer and facial expression generation (edition). Detailed functionalities will be described in the methodology section.

2. Related Work

In this section, past studies, systems and tools related to this project will be introduced.

2.1 Facial Action Coding System (FACS)

FACS is an anatomical system which was developed by Ekman and Friesen for facial expression measurement [1]. In this system, facial expressions are divided into several independent sets of muscle movement. The concept of Action Units (AUs) is introduced to represent the correlation between each independent region of a facial expression and the involved facial muscle. For example, AU1 represent the movement of “Inner Brow Raiser” which involves the frontalis and pars medialis muscle.

2.2 Facial Action Unit Intensity Estimation

Baltrušaitis et al. described a method to detect AU occurrence and intensity in real time based on facial geometric features [4]. The problem of individual difference was addressed by using Support Vector Machines (SVM) and Support Vector Regression (SVR) under a person-specific normalization approach based on cross-dataset learning. OpenFace is a popular open source toolkit for facial behavior analysis, including facial action unit detection based on the research of Baltrušaitis et al. It is able to extract 18 kinds of facial action units in 5 discrete levels of intensity.

2.2 Generative Adversarial Network (GAN)

In 2014, Goodfellow et al. proposed GANs which are a type of unsupervised deep neural network model structured based on a zero-sum game framework [2]. A classic GAN is composed of a generative network and a discriminative network which are trained simultaneously. The generative
network is trained to create realistic sample while the discriminative network is trained to
determine whether a sample is produced by the generative network or from a real dataset. GANs
are widely used in generative tasks including facial expression synthesis.

2.3 Facial Expression Generation conditioned by FACS

2.3.1 Direct Facial Expression Generation from 2D image to 2D image

Zhou and Shi proposed a conditional difference adversarial autoencoder (CDAAE) for
photorealistic facial expression synthesis based on FACS [12]. The CDAAE is able to generate a
new facial expression based on an unseen input facial image which does not exist in the training
dataset while preserving the facial identity. This is achieved by adding a low-level feedforward
connection between the encoder and decoder to disambiguate identity changes and facial
expression changes [12].

Pumarola et al. introduced a novel GAN scheme conditioned by AU annotation for facial
expression generation [3]. The model produces a mapping from a single facial image and an AU
intensity vector to a new image of the same facial identity under the desired facial expression.
Different from the approach suggested by Zhou and Shi [12], this model achieved an unpaired
image to image translation. Instead of pairs of images of the same person under different facial
expressions, only images with AU annotations are required for model training, which makes the
model more general and flexible. Moreover, Pumarola et al. made the network more robust to
background and lighting condition changes by adding an attention layer which limits the network
to only manipulate regions of images that are related to producing the new facial expressions [3].

2.3.1 Facial Expression Generation using 3D Model

Although it is not in the scope of this project, it is still worth to mention that apart from
transformation between 2D images, 3D model can be used to perform facial expression
generation. Instead of directly generating 2D images, Liu et al. suggested an approach combining
3D Morphable Model (3DMM) [14] with GANs [13]. 3DMM facial expression parameters are first
extracted from the input images. The parameters are then used to generate AUs-conditioned
3DMMs. Finally, the output images are rendered from the 3DMMs. Under this approach, the
model is able to generate high-resolution facial images.

3. Methodology

This section will describe the detailed methodology used in this project. First, method used to
represent facial expression throughout this project will be introduced. After that, the methodology
of the facial expression generation stage will be introduced including the procedure of data
preprocessing and the environment of the training and testing conducted. Then, the methodology
of the facial expression transfer stage will be described to explain how the OpenFace 2.0 toolkit
was combined with the trained GANimation generator model. Implementation details of the
demonstration website and the API server will also be included. Finally, the android mobile
application delivered in this project will be introduced in terms of functionality, user interface and
implementation details.

3.1 Facial Expression Representation under FACS

In this project, a subset of Action Units is used to represent facial expressions. Specifically, 17
Action Units are used including AU1, AU2, AU4, AU5, AU6, AU7, AU9, AU10, AU12, AU14, AU15,
AU17, AU20, AU23, AU25, AU26, AU45. The descriptions of AUs used are shown in the Table 1.
Facial expressions are separated into these 17 movements. The intensity of the movements are
evaluated as real numbers ranging from 0 to 5 inclusively to indicate the level of the movements. The intensity value 0 means that the corresponding AU is absent and the intensity value 5 means that the corresponding AU is at its most intensive level. Thus, facial expressions are coded as 17 dimensional arrays of real numbers.

<table>
<thead>
<tr>
<th>AU number</th>
<th>AU name</th>
<th>Example</th>
<th>AU number</th>
<th>AU name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inner brow raiser</td>
<td><img src="218x331" alt="Image" /></td>
<td>14</td>
<td>Dimpler</td>
<td><img src="450x388" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td>Outer brow raiser</td>
<td><img src="450x497" alt="Image" /></td>
<td>15</td>
<td>Lip corner depressor</td>
<td><img src="218x267" alt="Image" /></td>
</tr>
<tr>
<td>4</td>
<td>Brow lowerer</td>
<td><img src="450x325" alt="Image" /></td>
<td>17</td>
<td>Chin raiser</td>
<td><img src="218x325" alt="Image" /></td>
</tr>
<tr>
<td>5</td>
<td>Upper lid raiser</td>
<td><img src="218x396" alt="Image" /></td>
<td>20</td>
<td>Lip stretcher</td>
<td><img src="450x446" alt="Image" /></td>
</tr>
<tr>
<td>6</td>
<td>Upper lid raiser</td>
<td><img src="218x396" alt="Image" /></td>
<td>23</td>
<td>Lip tightener</td>
<td><img src="450x499" alt="Image" /></td>
</tr>
<tr>
<td>7</td>
<td>Lid tightener</td>
<td><img src="450x606" alt="Image" /></td>
<td>25</td>
<td>Lips part</td>
<td><img src="218x606" alt="Image" /></td>
</tr>
<tr>
<td>9</td>
<td>Nose wrinkler</td>
<td><img src="218x551" alt="Image" /></td>
<td>26</td>
<td>Jaw drop</td>
<td><img src="218x551" alt="Image" /></td>
</tr>
<tr>
<td>10</td>
<td>Upper lip raiser</td>
<td><img src="218x662" alt="Image" /></td>
<td>45</td>
<td>Blink</td>
<td><img src="218x662" alt="Image" /></td>
</tr>
<tr>
<td>12</td>
<td>Lip corner puller</td>
<td><img src="450x606" alt="Image" /></td>
<td></td>
<td></td>
<td><img src="218x606" alt="Image" /></td>
</tr>
</tbody>
</table>

Table 1. A list of used Action Units to represent facial expression in this project. The example images are from the FACS introduction site of the Carnegie Mellon's School of Computer Science, except for the example image for AU45 which is shared by John R. Hershey on the website of ResearchGate.
3.2 Facial Expression Generation

In this section, source of data used to train and test the GANimation model will first be introduced. After that, the procedure of data preprocessing will be described.

3.2.1 Data Collection

The training of the GANimation network requires a large amount of 2D RGC facial images. Data used in this project was collected from several open source databases and separated into two datasets. In this report, the two datasets will be referred as mixed dataset and EmotioNet Dataset respectively. The mixed dataset consists of data from multiple databases including CK+ database, GUFD, Color FERET Database, LFW Database and CelebA Dataset. The data in the EmotioNet dataset was solely collected from the EmotioNet database. The basic information of the databases is described as follows:

1. **Cohn-Kanade AU-Coded Expression Database (CK+ Database)**

   Cohn-Kanade AU-Coded Expression Database is a facial expression database prepared by Kanade, Cohn, & Tian [15] and Lucey et al. [16]. It includes 486 sequences of actions from 97 actors and AUs are marked on peak expression. However, due to historical reasons, most of the images are in monochrome, and cannot be used in the training of GANimation. After filtering, the number of color images in CK database is under 1000.

2. **Glasgow Unfamiliar Face Database (GUFD)**

   Glasgow Unfamiliar Face Database was a database prepared for Glasgow Face Matching Test (GFMT) [17]. It contains about 6000 images in total, from 303 identities and 20 images for each.

3. **Color FERET Database**

   Color FERET Database is part of the Facial Recognition Technology (FERET) program [18], which aims to develop new technology for the automatic recognition of human faces. It contains 14,126 facial images of 1199 individuals.

4. **Labeled Faces in the Wild Database (LFW Database)**

   Labeled Faces in the Wild [19] [20] contains more than 13,000 images of faces collected from the Internet by detecting with the traditional Viola-Jones algorithm.

5. **CelebFaces Attributes Dataset (CelebA Dataset)**

   CelebFaces Attributes Dataset [21] is the largest dataset that we can access apart from the EmotioNet Database. It contains 202,599 face images from 10,177 identities obtained from the Internet.

6. **EmotioNet Database**

   The EmotioNet Database is a product of a research study conducted by Fabian Benitez-Quiroz, C., Ramprakash Srinivasan, and Aleix M. Martinez in the Ohio State University [22]. The database consists of a million 2D RGB facial images in the wild automatically annotated with AU occurrence and intensity by an accurate algorithm. This database is used in the original work conducted by the team of Pumarola [3].
3.2.2 Data Preprocessing

Before the training the GANimation network, face detection was performed to all of the facial images collected. If there was no face detected in an image, the image would not be used for the training. After face detection, the faces detected from the images were cropped and resized to 128 * 128 pixels. Then the cropped facial images were re-annotated using the OpenFace 2.0 toolkit as required by the GANimation project.

3.2.2.1 Face Detection, Cropping and Resize

To crop the training images properly, face detection was applied during the process. The processes of face detection for the mixed dataset and the EmotioNet dataset are different. For the mixed dataset, the default Haar feature-based cascade classifiers included in OpenCV is used. It is an algorithm using haar features to detect objects in a monochrome image. As shown in Figure 1, there are 3 kinds of haar features, edge features, line features and four-rectangle features. They represent different pattern of white and black area of the image.

![Figure 1. Haar features for face detection.](image)

Here is an example of how a cascade classifier detect faces. In the image in Figure 2, there is an edge feature between the eyes and the cheeks, and a line feature within the area of the eyes and the bridge of the nose. If these kinds of features located in a proper position and with a proper size, there is most likely a face in that area.

![Figure 2. An example of face detection using haar cascade.](image)
However, the algorithm is not perfect and missed detection may occurs. Therefore, OpenCV provided different kind of cascade classifiers for us to choose. Table 2 lists out all the haar cascade files provided by OpenCV and the highlighted items are for human face detection.

<table>
<thead>
<tr>
<th>Built in haar cascades file</th>
<th>Target to detect</th>
</tr>
</thead>
<tbody>
<tr>
<td>haarcascade_eye.xml</td>
<td>eye</td>
</tr>
<tr>
<td>haarcascade_eye_tree_eyeglasses.xml</td>
<td>glasses</td>
</tr>
<tr>
<td>haarcascade_frontalcatface.xml</td>
<td>cat face</td>
</tr>
<tr>
<td>haarcascade_frontalcatface_extended.xml</td>
<td>cat face</td>
</tr>
<tr>
<td>haarcascade_frontalface_alt.xml</td>
<td>human face</td>
</tr>
<tr>
<td>haarcascade_frontalface_alt2.xml</td>
<td>human face</td>
</tr>
<tr>
<td>haarcascade_frontalface_alt_tree.xml</td>
<td>human face</td>
</tr>
<tr>
<td>haarcascade_frontalface_default.xml</td>
<td>human face</td>
</tr>
<tr>
<td>haarcascade_fullbody.xml</td>
<td>full human body</td>
</tr>
<tr>
<td>haarcascade_lefteye_2splits.xml</td>
<td>left eye</td>
</tr>
<tr>
<td>haarcascade_licence_plate_rus_16stages.xml</td>
<td>Russian number plate</td>
</tr>
<tr>
<td>haarcascade_lowerbody.xml</td>
<td>lower human body</td>
</tr>
<tr>
<td>haarcascade_profileface.xml</td>
<td>profile face</td>
</tr>
<tr>
<td>haarcascade_righteye_2splits.xml</td>
<td>right eye</td>
</tr>
<tr>
<td>haarcascade_russian_plate_number.xml</td>
<td>Russian number</td>
</tr>
<tr>
<td>haarcascade_smile.xml</td>
<td>smile</td>
</tr>
<tr>
<td>haarcascade_upperbody.xml</td>
<td>upper human body</td>
</tr>
</tbody>
</table>

Table 2. Built-in haar cascade file provided by OpenCV.

For the EmotioNet dataset, an open source python library called face_recognition was used for more accurate face detection as the accuracy of model provided by the face_recognition library is 99.38% on the Labeled Face in the Wild benchmark. After face detection, the detected faces were cropped from the facial images and resized to 128 * 128 pixels.

3.2.2.2 AU Intensity Detection and Data Selection

As instructed by the GANimation project, OpenFace 2.0 toolkit was used to detect AU intensities from the facial images. In addition, OpenFace 2.0 toolkit performed face detection before detecting AU intensities. If no face was detected in an image, it would be excluded from the dataset. After AU intensity detection, data selection was performed to ensure there was sufficient data for each AU at each intensity level. After data selection, 215650 images were collected into the mixed dataset. 194085 images were used for training and 21565 images were used for testing. For the EmotioNet dataset, 260000 images were collected. 200000 images from the EmotioNet dataset were used for training and 60000 images were used for testing.
3.2.3 Training of the GANimation Network

This section will first give an introduction of the GANimation network implemented by the team of Pumarola [3]. Then describe the training environment used in this project.

3.2.3.1 Introduction to the GANimation Network

The team of Pumarola implemented a novel generative adversarial network under FACS condition to perform photorealistic facial expression generation [3]. Because of the structure of the network, paired facial images are not required for training and testing which makes the collection of training and testing data more flexible. The structure of the GANimation network is shown in Figure 3. Instead of directly output an image, the generator outputs an attention mask and a color mask. The attention mask indicate the level of relevance for each pixel of the original image in generating new image. The color mask is used together with the attention mask to generate new RGB facial expression. The generator is used bidirectionally. The generator is first used to generate new image with desired facial expression. After that, the generator is applied to the new generated image and tries to reverse its facial expression to the original one. The reversely generated image is used to compare with the original image in order to evaluate the generator.

![Figure 3. The structure of the GANimation network.](image)

3.2.3.2 Training Environment

The GANimation network is implemented in Pytorch 0.3.1 with CUDA support. A machine with a suitable GPU is required to train the network. In order to minimise the cost of this project, training was performed on virtual machines running on Google Cloud Platform (GCP) as GCP provide free trial with access to high performance GPUs.

Training was performed using the mixed dataset and the EmotioNet dataset respectively. There were two training environment used in this project which will be referenced as Env_old and Env_new in this report. The training of the GANimation network was first performed in Env_old. However, the training speed on Env_old was too slow and the training time was too long. Therefore, a new environment was created for faster training. The detailed training performance will be discussed in the Results and Limitations section. The specification of the two environments are as follows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>8 virtual CPUs</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>30GB</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>1 x NVIDIA Tesla K80</td>
</tr>
</tbody>
</table>

**Table 3. Specification of Env_old used perform training with the mixed dataset.**
3.3 Facial Expression Transfer

This section will first introduce the methodology of achieving facial expression transfer by combining the OpenFace 2.0 toolkit and the trained GANimation generator. A website with simple and clear user interface was developed to demonstrate the functionalities of both facial expression transfer and facial expression generation. To provide service of facial expression transfer and generation, an API server was built. The API server developed in this stage was an intermediate implementation and only served for the demonstration website but not the android application.

3.3.1 Demonstration Website

In this stage, a website was developed for demonstrating the result of facial expression transfer and for users to alter the AU values of a facial image and inspect how the changes of AU values affect facial expressions. Since it was just for demonstration purpose, the user interface was simple and the website was developed by HTML and Javascript only.

The website consists of two webpages for facial expression transfer and generation respectively. For facial expression transfer, users are able to import two facial images, source image and target image, and a result facial image of the target's face with the source's facial expression will be shown. For facial expression generation, users can import one facial image, adjust its AU values and observe the change of its facial expression in realtime.

3.3.2 RESTful API server

The API server was built to provide two services to the demonstration website which are facial expression transfer and facial expression generation. The API server was developed depending on the OpenFace 2.0 Toolkit and the GANimation network as shown in Figure 4.

For facial expression transfer, the API server takes two facial images as inputs. One image is for facial expression extraction, namely source image. Another image contains a face which the extracted facial expression is transferred to, namely target image. When the API server receives a request of facial expression transfer, it first calls the OpenFace 2.0 toolkit to extract AU values from the source image. Then, the API server sends a request to a python server with the trained GANimation generator preloaded. The python server takes the target image and the extracted AU values as inputs and generates the desired facial expression on a face in the target image. When the computation is finished, the python server responds to the API server with a cropped face image with desired facial expression together a tuple of integers (x, y, weight, height) indicating the cropped location and the face image size.

For facial expression generation, clients are required to send request to the API server with a target image and a set of desired AU values. The flow of handling facial expression generation request is similar to the one of facial expression transfer but the OpenFace 2.0 Toolkit is not called and the input target image and AU values from the clients are directly sent to the python server.
3.4 Android Mobile Application Development

In this stage an android mobile application was developed to provide more sophisticated service of both facial expression transfer and generation (by editing AU values) with a user friendly interface. In order to provide more functionalities and better performance, a new API server was built to provide more efficient and sophisticated service to the android application.

3.4.1 Android application

The android application aims to provide a user friendly platform for users to perform facial expression transfer and modifying facial expression by editing AU values. Figure 5a, 5b and 5c show the basic UI design of the facial expression transfer activity. As shown in Figure 5a, users will first be asked to import an image from which a facial expression will be extracted. Users can either import an image from the file system or take a photo using their phone camera.

![Figure 5](image)

After that, users can swipe left to the page shown in Figure 5b and import another image which will be the target of facial expression transfer. Finally, users can swipe left again to enter the page to perform facial expression transfer. As shown in Figure 5c, users can click the GIF button to choose either produce a static image or an animated image. When users click on the face button, facial expression transfer will be performed and the result will be shown. Users can export the result image to their file system by clicking the export button.

Apart from facial expression transfer, the application also provides a convenient interface for users to modify facial expressions by change AU values. Figure 6a and Figure 6b show the basic UI design of the AU editing activity. As shown in Figure 6a, a circular scrolling panel is provided for users to select an AU to edit. After selecting an AU, a circular scrolling bar is provided for changing the corresponding AU intensity value as shown in Figure 6b. The facial expression will be changed accordingly in realtime.

![Figure 6](image)
Figure 5a. Basic UI design of page for import image to extract facial expression. The import button is marked by label (1). The photo taking button is marked by label (2). The sample image credits to Philips Cavalcante.

Figure 5b. Basic UI design of page for import image to transfer facial expression.

Figure 5c. Basic UI design of page to perform facial expression transfer. The transfer button is marked by label (1). The export button is marked by label (2). The GIF button is marked by label (3).

Figure 6a. Basic UI design of page for AU editing (generation). User can select an AU to edit from the circular scrolling panel.

Figure 6b. Basic UI design of page for AU editing (generation). After selecting an AU to edit, a circular scrolling bar is provided for changing the AU value.
3.4.2 Refined API server

Instead of using NodeJS, a new API server was built using the Flask framework in python such that the trained GANimation generator model can be directly loaded in the API server and the python server is no longer required. By eliminating the communication overhead between the API server and the python server, the API server is able to provide more efficient services. Figure 7 shows the architecture of the refined API server.

![Figure 7. Request handling flow of the new API server.](image)

4. Results and Limitation

This section will first discuss the training results of the GANimation network. After that, implementation results of the demonstration website, android application and the API server will be described.

4.1 GANimation Training Results

In this section the results of the last two training will be discussed and compared. The two training processes are referred as Mixed Dataset Training and EmotioNet Dataset Training respectively. The information of the two training is shown in Table 5.

<table>
<thead>
<tr>
<th>Training name</th>
<th>Dataset Used</th>
<th>Number of Training Data</th>
<th>Number of Testing Data</th>
<th>GPU</th>
<th>CPU</th>
<th>Memory</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed Dataset Training</td>
<td>Mixed Dataset</td>
<td>194085</td>
<td>21565</td>
<td>1 x NVIDIA Tesla K80</td>
<td>8 virtual CPUs</td>
<td>30 GB</td>
<td>224 hours</td>
</tr>
<tr>
<td>EmotioNet Dataset Training</td>
<td>EmotioNet Dataset</td>
<td>200000</td>
<td>60000</td>
<td>1 x NVIDIA Tesla P100</td>
<td>16 virtual CPUs</td>
<td>104 GB</td>
<td>34 hours</td>
</tr>
</tbody>
</table>

Table 5. Information of Mixed Dataset Training and EmotioNet Dataset Training.

From the summary written by tensorboard, the EmotioNet Dataset Training is more satisfying that the Mixed Dataset Training. Figure 8a shows the performance of the generator in producing realistic fake images during the Mixed Dataset Training while Figure 8b shows the same performance of the generator during the EmotioNet Dataset Training. The corresponding loss function during the EmotioNet Dataset Training is lower which means the generator performs
better in the EmotioNet Dataset Training. The same can be proved by comparing the performance of the discriminator. Figure 8c and 8d show the performance of the discriminators in discriminating fake images during the two training respectively. As shown from the figures, the discriminator performs worse during the EmotioNet Database Training which indicates that the fake images produced by the generator are harder to discriminate and more realistic.

By observation, there is a limitation of the generator trained in the Mixed Dataset Training to generate facial expression for several AUs. In particular the generator cannot produce observable results in generating AU17, AU23 and AU45. And the noise when generating AU17 is huge. Figure 9 shows the single AU edition of AU17, AU23, AU45. It can be observed that even the AU intensity of the corresponding AU is at the highest level, there is no obvious change.

Figure 8a. Loss values to evaluate the performance of the generator in producing fake image during the Mixed Dataset Training.

Figure 8b. Loss values to evaluate the performance of the generator in producing fake image during the EmotioNet Dataset Training.

Figure 8c. Loss values to evaluate the performance of the discriminator in discriminating fake image during the Mixed Dataset Training.

Figure 8d. Loss values to evaluate the performance of the discriminator in discriminating fake image during the EmotioNet Dataset Training.

Figure 9. Single AU edition on AU17, AU23 and AU45 performed by the generator trained by the Mixed Dataset. The changes are not obvious and the edition of AU45 produces a lot of noise.
The reason of the above problem could be the insufficient training data of the above AUs with high intensity values. Figure 10 shows the distribution of AU values for each of the AUs in the Mixed Dataset. It can be observed that the standard deviation of several AUs including AU17, AU23 and AU45 are low comparing with others.

In order to solve the above problem, data selection is performed in the EmotioNet Dataset. Figure 11 shows the distribution of AU values for each of the AUs in the EmotioNet Dataset after data selection. As shown in the figure, the problem of low standard deviation for the mentioned AUs is solved.

To evaluate whether the problem is solved by attempted solution of data selection, the single AU edition on the involved AUs are performed again by the generator trained by the EmotioNet Dataset. Figure 12 shows the result. It can be observed that the problem of insufficient effect on editing those AUs is solved and the noise when changing AU45 is lower. Therefore, it can be concluded that data selection is effective in solving the above mentioned problem.
For better observation of the performance of the generator trained by the EmotioNet dataset. AU value edition with different intensity levels was performed on several AUs. Figure 13 shows a batch of single AU value edition results of a subset of AUs involved in this project. It can be observed that the trained generator is able to change the facial expressions according to the input AU values and produce desired facial expressions.

Figure 12. Single AU edition on AU17, AU23 and AU45 performed by the generator trained by the EmotioNet Dataset.

Figure 13. A batch of single AU value edition results
4.2 Software Implementation Result

This section will introduce the software implementation result of this project. First, the implemented demonstration website will be introduced. After that, implementation result of the android mobile application and the API server will be discussed.

4.2.1 Demonstration Website Implementation Result

The demonstration website has been implemented. It is an intermediate deliverable of this project. It provides simple interface for demonstrating the basic function of our software. Figure 14 shows the interface of the website demonstrating the basic function of facial expression transfer.

Figure 14. Implementation result of the facial expression transfer demonstration webpage.

Figure 15 shows the interface of demonstration page of AU editing which provides sliders for users to conveniently change to AU values. The change of the facial expression is shown instantly such that it is easy to observe and inspect how the change of AU values affects facial expressions.

Figure 15. Implementation result of the AU editing demonstration webpage
4.2.2 Android Application Implementation Result

The android application has been successfully implemented following the basic UI design described in the Methodology section. It is available on https://github.com/choiwaiyiu/fyp18066-android-app. Figure 16 shows the implementation result of the facial expression transfer activity. Users are able to import source and target images and perform facial expression transfer. A button is provided for users to choose whether the result is static or animated. An export button is provided for export the result image into the file system of the device.

The AU editing activity has also been successfully implemented. As shown in Figure 17, users can import an image and edit the AU values. When users changed an AU value using the circular scrolling bar, the change of facial expression will be shown instantly.

Figure 16. The import pages and the resulting page of the facial expression transfer activity. The face icon shown is modified based on icons made by www.freepik.com from www.flaticon.com

Figure 17. The implementation of AU editing activity. Icons shown are modified based on icons made by www.freepik.com from www.flaticon.com

23
4.2.3 API Server Implementation Result

This API server has been developed and is available on https://github.com/choiwaiyiu/facial-expression-transfer-flask-server. The API server provides several routes for different services and all of the operations have been tested to give correct response.

1. **Facial Expression Transfer (Static Image Result / Animated Image Result)**

<table>
<thead>
<tr>
<th>Method and URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Image</td>
<td>POST /api/facial-expression-transfer</td>
</tr>
<tr>
<td>Animated Image</td>
<td>POST /api/facial-expression-transfer-gif</td>
</tr>
</tbody>
</table>

**Request Header**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td></td>
<td>multipart/form-data</td>
</tr>
</tbody>
</table>

**Request Body**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>au-img</td>
<td>File</td>
<td>The source image from which the au values are extracted.</td>
</tr>
<tr>
<td>target-img</td>
<td>File</td>
<td>The target image to which the extracted facial expression is transferred to.</td>
</tr>
</tbody>
</table>

**Success 200**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>res-img</td>
<td>File</td>
<td>A full size image (not cropped face) of the facial expression transfer result. Could be a static or an animated image.</td>
</tr>
</tbody>
</table>

**Bad Request 400**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MissingField</td>
<td>au-img or target-img is not provided.</td>
</tr>
<tr>
<td>NoFaceDetected</td>
<td>No face detected from au-img or target-img.</td>
</tr>
</tbody>
</table>

2. **Facial Expression Generation (Static Image Result / Animated Image Result)**

<table>
<thead>
<tr>
<th>Method and URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Image</td>
<td>POST /api/facial-expression-generation</td>
</tr>
<tr>
<td>Animated Image</td>
<td>POST /api/facial-expression-generation-gif</td>
</tr>
</tbody>
</table>

**Request Header**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td></td>
<td>multipart/form-data</td>
</tr>
</tbody>
</table>
### Request Body

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aus</td>
<td>String</td>
<td>String form of 17 dimension array of au values of the desired facial expression.</td>
</tr>
<tr>
<td>target-img</td>
<td>File</td>
<td>The target image to which the extracted facial expression is transferred to.</td>
</tr>
</tbody>
</table>

### Success 200

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>res-img</td>
<td>File</td>
<td>A full size image (not cropped face) of the facial expression generation result. Could be a static or an animated image.</td>
</tr>
</tbody>
</table>

### Bad Request 400

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MissingField</td>
<td>aus or target-img is not provided.</td>
</tr>
<tr>
<td>NoFaceDetected</td>
<td>No face detected from target-img.</td>
</tr>
</tbody>
</table>
5. Conclusion and Future Work

This project aims to achieve facial expression transfer and build software to provide such service. Our facial expression transfer software was built on top of the GANimation network. In order to produce satisfying results, it was essential to train the GANimation network and reproduce the successful result of the original work conducted by the team of Pumarola [3]. Throughout the project, training of the GANimation network has been performed with different datasets under different environments. Method such as data selection from the databases has been applied to improve the training results. Although the training results were not satisfying at first, we were able to produce results close to the original work at the end of this project.

In addition to implementing the function for facial expression transfer by training the GANimation model and combining it with the OpenFace 2.0 Toolkit, a website and an android application were also developed. The website is simple as it was only for demonstration purpose while the function of the android application is more sophisticated. The android application provides a set of services that helps users to understand the Facial Action Coding System (FACS) and to enjoy the service of facial expression transfer. At first, the android application was an extra deliverable but now has become one of the main deliverables of this project. Although the goal of this project has been achieved, there are some possible further improvements which could make the application more efficient. For example, for the AU editing activity, it is possible for the client to only send the target image once to the API server at the beginning of the activity. Then the API server could cache the target image. In this case, instead of sending an image to the server every time the user changed an AU value, the client only needs to send a set of AU values to the server such that the communication overhead can be reduced and the service could become more efficient.

As there are a lot of possible ways to improve the performance of the android application, the future work of this project should be improving the efficiency of the software as much as possible and try to implement real-time facial expression transfer which allows users to transfer their expression to another person in real-time.
6. References


