Facial Expression Transfer with Machine Learning

14 April, 2019

Supervisor: Dr. Dirk Schnieders
CHENG Tak Kin (3035240116)

with

CHOI Wai Yiu (3035240568)
Abstract

This project aims to study the usage of deep learning and Facial Action Coding System (FACS) [1] in facial expression transfer. In this project, facial expression transfer will be achieved using the technology of facial expression extraction and facial expression generation under a FACS conditioned scheme which improves the complexity and anatomical accuracy of the output. A software will be developed using a Generative Adversarial Network (GAN) model [2] based on Action Units (AUs) annotations to achieve photorealistic facial expression synthesis. The software will be built on top of the GANimation model proposed by Pumarola et al. [3] and the software will be further combined with the OpenFace 2.0 toolkit [4] to achieve facial expression transfer.

Acknowledgment

We would like to express our deep gratitude to our supervisor, Dr. Dirk Schnieders. We had changed our project 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 thanks for his support on consulting us and applying for the access of the database.

We would also like to thanks Miss Yeung Hei Tung for her help on designing the user interface of the mobile application.
# TABLE OF CONTENTS

1. **INTRODUCTION** ......................................................................................................................... 5  
   1.1 **OBJECTIVE** .......................................................................................................................... 5  
   1.2 **MOTIVATION** ....................................................................................................................... 5  
   1.3 **SCOPE OF WORK** ............................................................................................................... 6  

2. **RELATED WORK** ..................................................................................................................... 7  
   2.1 **FACIAL ACTION CODING SYSTEM (FACS)** ................................................................. 7  
   2.2 **GENERATIVE ADVERSARIAL NETWORKS (GANs)** ..................................................... 7  
   2.3 **FACIAL EXPRESSION SYNTHESIS BASED ON FACS** ............................................... 7  
      2.3.1 **2D Action Unit Representation** .................................................................................. 7  
      2.3.2 **3D Action Unit Representation** ................................................................................ 8  
   2.4 **FACIAL ACTION UNIT INTENSITY ESTIMATION** ...................................................... 8  
   2.5 **OPENFACE 2.0** ............................................................................................................... 9  
   2.6 **GANimation** .................................................................................................................... 10  

3. **METHODOLOGY** ..................................................................................................................... 12  
   3.1 **FIRST EXPERIMENT OF TRAINING THE GANimation MODAL** .............................. 12  
      3.1.1 **Datasets** ................................................................................................................... 12  
      3.1.2 **Data Pre-processing** ................................................................................................ 13  
      3.1.3 **Training** .................................................................................................................... 15  
   3.2 **SECOND EXPERIMENT OF TRAINING THE GANimation MODAL** ............................. 16  
      3.2.1 **Dataset** ..................................................................................................................... 16  
      3.2.2 **Data Pre-processing** ................................................................................................ 16  
      3.2.3 **Training** .................................................................................................................... 21  
   3.3 **SOFTWARE IMPLEMENTATION** ...................................................................................... 22  
      3.3.1 **Web Application Implementation of Experiment 1** ............................................... 22  
      3.3.2 **Mobile Application Implementation of Experiment 2** ......................................... 26  

4. **RESULTS** .............................................................................................................................. 29  
   4.1 **LOSS VALUES OF TENSORBOARD** .................................................................................. 29  
   4.2 **RESULT OUTPUTS** ......................................................................................................... 33  
      4.2.1 **Comparison of both experiments** ............................................................................. 33  
      4.2.2 **Limitation** .................................................................................................................. 37  

5. **CONCLUSION** ....................................................................................................................... 41  

6. **REFERENCES** ....................................................................................................................... 42
List of Figures

Figure 1. A demo of image processed by OpenFace ......................................................... 9
Figure 2. Illustration of how GANimation generate a new image ........................................ 10
Figure 3. Training process of GANimation ............................................................................ 11
Figure 4. haar features for object detection ........................................................................... 13
Figure 5. An example of face detection using haar cascade .................................................. 14
Figure 6 Distribution of AU12 and AU45 of EmotioNet ....................................................... 19
Figure 7. Sample response body from the server ................................................................. 23
Figure 8. UI of AU editor of the web demo ........................................................................... 23
Figure 9. UI of facial expression transfer of the web demo ................................................... 24
Figure 10. Flow of the server of web application handling a request ..................................... 25
Figure 11. UI of facial expression transfer of the mobile application ..................................... 26
Figure 12. UI of AU editor of the mobile application ............................................................. 27
Figure 13. Flow of the server of mobile application handling a request .................................. 28
Figure 14. Loss values of output AU values for the generator .............................................. 29
Figure 15. Loss values of the attention mask output of new generated image for the generator .................................................................................................................. 30
Figure 16. Loss value of the attention mask output of real image existing in the training dataset for the generator .................................................................................................... 30
Figure 17. Loss value of the quality of new generated image by the generator ................. 31
Figure 18. Loss value of AU value discrimination (d_real) and discriminating new generated image (d_fake) for the discriminator ................................................................. 31
Figure 19. Loss value of discriminating real image existing in training dataset for the discriminator .............................................................................................................................. 32
Figure 20. Sample results of altering AU17, AU23 and AU45 of experiment 1 .................. 33
Figure 21. Result of altering AU17 by applying model of experiment 2 ............................... 34
Figure 22. Result of altering AU23 by applying model of experiment 2 and an image with value 0 on AU23 ............................................................................................................. 35
Figure 23. Result of altering AU45 by applying model of experiment 2 ............................... 36
Figure 24. Reverse result of generating low Blink intensity .................................................... 37
Figure 25. Expression image of mouth closing.......................................................... 38
Figure 26. Result of tooth generation ........................................................................... 38
Figure 27. Result of removing tooth from generated images ........................................... 39
Figure 28. Result of removing tooth from real images ..................................................... 40
List of Tables

Table 1: List of AUs can be extracted by OpenFace .................................................. 9
Table 2. Built in haar cascade files of OpenCV .............................................................. 14
Table 3. Specification of the virtual machine used in experiment 1 .............................. 15
Table 4. sd of AU values of dataset of experiment 1 ...................................................... 17
Table 5. sd of AU values of EmotioNet......................................................................... 18
Table 6. sd of AU values of dataset used in experiment 2 ........................................... 20
Table 7. Specification of the virtual machine used in experiment 2 ......................... 21

Abbreviations

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
LFW Labeled Faces in the Wild database
1. Introduction

Modifying facial expression is a large topic within computer vision. It involves the understanding of facial expression, facial characteristic understanding of the transferrer and transferee, and the technique of generating a realistic facial image.

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.

1.1 Objective

The main objective of this project is to achieve accurate and realistic facial expression transfer by combining facial expression extraction and facial expression generation under the FACS Action Units conditioned deep learning scheme. As the technology of facial expression extraction is mature, facial expression generation is the focus of this project.

1.2 Motivation

In 15th European Conference on Computer Vision, Pumarola et al. introduced GANimation [3]. It is a deep learning modal GANimation using GAN under FACS to generate human face images with new facial expression. The high quality of it motivated us to achieve facial expression transfer of 2D images on top of it.

Generative adversarial networks (GANs) is a type of deep learning model introduced by Goodfellow et al. in 2014 [2], which are decided for generative tasks. Before GANimation is introduced, many other research studies in using various types of GANs for facial expression generation have been published such as CycleGAN [7], IcGAN [8] and StarGAN [9]. Within them, 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 marked with binary emotional labels such as happy, sad, angry, fearful, etc. It makes the generation results are limited to the corresponding set of emotional facial expression and complex facial expression generation is not applicable [2].
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] [11] [12]. 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

There deliverable of this project is a software doing facial expression transfer using deep learning. At the early stage, the plan was to build a standalone mobile application. However, after further research, we found that it is difficult to be finished in limited time, and it is out of the focus of our project. Thus, the final deliverable would be a basic server-client mobile application, and the project will focus on improving the training result of the deep learning model.
2. Related Work

2.1 Facial Action Coding System (FACS)

FACS is an anatomical system which is developed by Ekman and Friesen in 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 Generative Adversarial Networks (GANs)

Generative jobs were hard for deep learning because there are no absolute rules to determinate how a result is realistic. However, 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] to solve the problem. 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. Which means that, the task of determining a result is realistic or not is also done by deep learning. Currently, GANs are widely used in generative tasks including facial expression synthesis.

2.3 Facial Expression Synthesis based on FACS

2.3.1 2D Action Unit Representation

Zhou and Shi proposed a conditional difference adversarial autoencoder (CDAAE) for photorealistic facial expression synthesis based on FACS [11]. 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 [11].

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 [11], 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 at 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].

2.3.2 3D Action Unit Representation

Instead of directly generating 2D images, Liu et al. suggested an approach combining 3D Morphable Model (3DMM) [13] with GANs [12]. 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.

2.4 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.
2.5 OpenFace 2.0

OpenFace is an integration of facial landmark detection and tracking [14] [15], eye gaze tracking [16] and facial action unit detection [17]. Figure 1 is a demonstration of these functionalities of OpenFace. Detected facial landmark is marked as red dots, 3D rotation of the face is presented by the blue box, and direction of eye gaze is showed by the two green lines.

![Image of OpenFace demonstration](image)

**Figure 1. A demo of image processed by OpenFace**

In this project, the facial action unit detection function of OpenFace will be applied. OpenFace can detect 18 types of Action Units as shown in Table 1.

<table>
<thead>
<tr>
<th>AU number</th>
<th>Description</th>
<th>AU number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inner brow raiser</td>
<td>14</td>
<td>Dimpler</td>
</tr>
<tr>
<td>2</td>
<td>Outer brow raiser</td>
<td>15</td>
<td>Lip corner depressor</td>
</tr>
<tr>
<td>4</td>
<td>Brow lowerer</td>
<td>17</td>
<td>Chin raiser</td>
</tr>
<tr>
<td>5</td>
<td>Upper lid raiser</td>
<td>20</td>
<td>Lip stretcher</td>
</tr>
<tr>
<td>6</td>
<td>Cheek raiser</td>
<td>23</td>
<td>Lip tightener</td>
</tr>
<tr>
<td>7</td>
<td>Lid tightener</td>
<td>25</td>
<td>Lips part</td>
</tr>
<tr>
<td>9</td>
<td>Nose wrinkler</td>
<td>26</td>
<td>Jaw drop</td>
</tr>
<tr>
<td>10</td>
<td>Upper lip raiser</td>
<td>28</td>
<td>Lip suck</td>
</tr>
<tr>
<td>12</td>
<td>Lip corner puller</td>
<td>45</td>
<td>Blink</td>
</tr>
</tbody>
</table>

Table 1: List of AUs can be extracted by OpenFace

Except AU28, OpenFace can extract these AU in a 5-level scale, namely AU intensity. It describes the detail status of the muscle and action related that AU. For example, a 0.2 level of AU26 means the jaw is just starting to drop, and a 1.0 level means the jaw drop is fully performed. By the intensity, GANimation can learn and reproduce images in detail. GANimation make uses of all these AUs except AU28, totally 17 types of AUs.
2.6 GANimation

GANimation is a GAN model introduced by Pumarola et al to apply new facial expression to 2D images using deep learning and the new facial expression image generator that will be applied in this project. Although its source code is available on GitHub, there is no pre-trained model provided. Thus, this project also include the training part of GANimation.

As mentioned in section 2.2, GAN contains a generative network and a discriminative network. In GANimation, the generative network is further separated into two networks, one for attention mask and one for color mask. Figure 2 shows how these two masks can be used to generate a new image.

![Figure 2. Illustration of how GANimation generate a new image](image)

In Figure 2, $y_i$ means the new expression to generate, $y_o$ means the originale expression, and so $I_{yo}$ means the original image and $I_{yf}$ means the image with new expression. $G_A$ is the generative network of attention mask, and $G_C$ is the generative network of color mask. Therefore, inputting the original image $I_{yo}$ into network $G_A$ ($G_A(I_{yo} | y_i)$) will give an attention mask $A$, and input them into $G_C$ ($G_C(I_{yo} | y_i)$) will give a colour mask. Attention mask is to indicate the weight to apply the color mask into the original image. The equation $(1-A) \cdot C + A \cdot I_{yo}$ means apply more color from the original image if the attention mask has a high value (light color) in that pixel ($A \cdot I_{yo}$), and apply more from color mask if the attention mask has a low value (dark color) in that pixel ($(1-A) \cdot C$). And combining the two will give the result image $I_{yf}$. For example, in Figure 2, the attention mask is dark around the area of the mouth, so more color of the color mask will be picked. On the other hand, the four corners are light, so no change will be made to the original image $I_{yo}$ and just copy the pixel to the new image $I_{yf}$.
Figure 3. Training process of GANimation

Other than the generative network and a discriminative network approach of GAN, GANimation also have a unique process of training as shown in Figure 3. In each step, GANimation will try to generate a new image with random expression. After that, the result will not only be piped into the discriminative network, but also the generative network again, with the original expression. Using the generated image and original expression as input means that, it is trying to regenerate the original image. And this is how GANimation learn, if we stop at generating a non-existing image, there is no indicator for it to determine the expression is right or not, but when it is trying to reproduce the real original image, it can rate and adjust itself by using the original image as a reference.
3. Methodology

This section will explain the methodology used to produce the open source facial expression transfer software of this project. The utilities scripts used in this section are written in Python and NodeJS.

3.1 First Experiment of Training the GANimation modal

The first experiment was done in the first semester.

3.1.1 Datasets

A large amount of data is required for training the GANimation model. There are several datasets available on the Internet. Initially, it was decided that the EmotioNet database and CK+ database will be used in the current stage. Nevertheless, the access to EmotioNet database was not granted for this project at the time of the first experiment. Therefore, some alternative databases are used. The following is a list of alternative databases used in this project and description of the databases.

3.1.1.1 CK+ database

Cohn-Kanade AU-Coded Expression Database is a facial expression database prepared by Kanade, Cohn, & Tian [18] and Lucey et al. [19]. 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, it turns out that number of color images in CK database is under 1000.

3.1.1.2 GUFD

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

3.1.1.3 Color FERET Database

Color FERET Database is part of the Facial Recognition Technology (FERET) program [21], which aims to develop new technology, for the automatic recognition of human faces. It contains 14,126 facial images of 1199 individuals
3.1.1.4 LFW

Labeled Faces in the Wild [22] [23] contains more than 13,000 images of faces collected from the Internet by detecting with the traditional Viola-Jones algorithm.

3.1.1.5 CelebA

CelebFaces Attributes Dataset [24] is the largest dataset that we can access. It contains 202,599 face images from 10,177 identities obtained from the Internet.

3.1.2 Data Pre-processing

To be used to train GANimation, the images should be in size 128*128 pixels, and their AU indensities must be labeled by OpenFace.

3.1.2.1 Crop and resize the images and face detection

To crop the training images properly, face detection is applied during the process. 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 4, 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.

![Haar Features](image)

(a) Edge Features

(b) Line Features

(c) Four-rectangle features

**Figure 4. haar features for object detection**

Here is an example of how a cascade classifier detect faces. In the image of Figure 5, 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 5. An example of face detection using haar cascade

However, the algorithm is not perfect and missed detection may occur. Therefore, OpenCV provided different kinds 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 2. Built in haar cascade files of OpenCV

<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>
3.1.2.2 AU intensities detection

As suggested by GANimation, OpenFace 2.0 is used to detect the AU intensities of face images. Additionally, OpenFace 2.0 won’t give any output if it cannot recognize it is a face image. In this case, we would try to use another haar cascades file to detect face from the original image, which can minimize the chance of miss detection. After detection, there are 215650 images. 90% (194085) of them are used to train and 10% (21565) are used to test.

3.1.3 Training

After following the instruction of GANimation to pack the training data into a .pkl file, the training of the GANimation model was performed.

GANimation used CUDA support of PyTorch 3.1. Thus, a machine with suitable GPU is required to train the model. In order to minimize the cost of this project, virtual machine rental service from cloud computing platform is used. Among Microsoft Azure, Amazon Web Service (AWS) and Google Cloud Platform (GCP), GCP is chosen to host the project server as it is the only platform providing a GPU instance on free trial. Table 3 shows the specification of the virtual machine.

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

The original GANimation stated that they used 2 days to train with GeForce 1080Ti GPU. However, our training process lasted over 9 days (224 hours), which is much longer than the original one, while Tesla K80 is a more powerful server use GPU. We guess the bottleneck exists in CPU or memory. We will adjust the specification according to this information of then virtual machine in the next experiment.

Also, since the training process is done on the virtual machine on the cloud, the training process is attached into the ssh session, which means it will be terminated if the connection is disconnected. Thus, we need to set the ssh client to keep the connection alive and connect it on a desktop computer. We found the approach is inefficient and will improve on the next experiment.
3.2 Second Experiment of Training the GANimation modal

After the first presentation, and after further discussions with our supervisor, we found some shortcomings of the first model. Thus, we planned to retrain the model to improve the quality.

3.2.1 Dataset

Thanks for our supervisor, we had successfully accessed the EmotioNet dataset. Rather than a few compressed files of images, the dataset just contains hyperlinks from the internet, and we need to download them one by one, which is very slow. It costed us 3 days by using two machines to download them parallelly. Moreover, a significant amount of links was not available and responded with http status code 404, or it is just not an image of human face. Thankfully, the base number of EmotioNet dataset is large and we can still secure enough data.

3.2.2 Data Pre-processing

To be used to train GANimation, the images should be in size 128*128 pixels, and their AU indensities must be labeled by OpenFace.

3.2.2.1 Face detection, crop and resize

In the first experiment, we had used Haar feature-based cascade classifiers of OpenCV to detect faces. In the second one, the Face Recognition package [25] is used to achieve the task. It makes use of machine learning to recognize faces. Although it is not good at detecting Asian faces (only mentioned in Chinese documentation), its overall performance is still better than the Haar feature-based cascade method. It hit an accuracy of 99.38% on the Labeled Faces in the Wild database we mentioned.

Regarding the time performance, although the Face Recognition package itself is slower than the Haar feature-based cascade method, it supports multicore processing and can compute jobs parallelly to fully utilize a multi core CPU.

After face detection, OpenCV is used to resize and crop the face out as same as experiment 1.
3.2.2.2 AU intensities detection and data selection

As same as experiment 1, OpenFace 2.0 is used to detect the AU intensities and filter the images which is mis-detected. We finally detected AU intensities of 864,370 images. Then, we need to pick 200,000 images to train the model.

To help data selection, we use the database MongoDB to do statistics. We choose the official MongoDB Atlas service, which provide 500MB free storage online and we can access it locally through internet to minimize our setup cost. It also provides data visualization service.

The AU values OpenFace 2.0 detected has a range of 0 to 5, with precision of 2 decimal places. For easy calculation, we rounded the values to nearest 0.5. None of the images have an AU value reaching 5.0 after rounding. Therefore, there will be 10 levels of value, 0.0 to 4.5.

At first, we decide descriptive key names “AUx” and “AUx_rounded” (x is the AU number) for MongoDB documents. However, it almost fully used the 500MB storage after we inserted the values of EmotioNet dataset. And we cannot insert the values of the old dataset for further analyzation. Thus, since key lengths can impact the document size for a small document, we chose shorter key names “x” for original value and “xr” for rounded value, and successfully cut the storage usage to about 300MB.

To determine the strategy of data picking, we need to analyse the dataset of experiment 1 first. Table 4 shows the standard deviation(sd) of AU values of the dataset.

<table>
<thead>
<tr>
<th>AU</th>
<th>sd</th>
<th>AU</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU1</td>
<td>0.59612626524</td>
<td>AU14</td>
<td>0.6951026526</td>
</tr>
<tr>
<td>AU2</td>
<td>0.4634682138</td>
<td>AU15</td>
<td>0.4035686572</td>
</tr>
<tr>
<td>AU4</td>
<td>0.5533035351</td>
<td>AU17</td>
<td>0.4060795155</td>
</tr>
<tr>
<td>AU5</td>
<td>0.6047101598</td>
<td>AU20</td>
<td>0.530510075</td>
</tr>
<tr>
<td>AU6</td>
<td>0.7438906198</td>
<td>AU23</td>
<td>0.1988374049</td>
</tr>
<tr>
<td>AU7</td>
<td>0.8450710369</td>
<td>AU25</td>
<td>0.7900577559</td>
</tr>
<tr>
<td>AU9</td>
<td>0.3909753326</td>
<td>AU26</td>
<td>0.5102071786</td>
</tr>
<tr>
<td>AU10</td>
<td>0.6950645292</td>
<td>AU45</td>
<td>0.1825096838</td>
</tr>
<tr>
<td>AU12</td>
<td>0.9679101752</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Form Table 4, we found that the sd of AU23 and AU45 are significantly low, which is the two AUs that the model of experiment 1 not functioning. Thus, our goal in this stage is to balance their sd.
After analyzing the data of experiment 1, we knew that an ideal set of AU values should not have a low sd. Table 5 shows sd of AU values of the dataset before rounding.

**Table 5. sd of AU values of EmotioNet**

<table>
<thead>
<tr>
<th>AU</th>
<th>sd</th>
<th>AU</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU1</td>
<td>0.6556026739</td>
<td>AU14</td>
<td>0.7320542976</td>
</tr>
<tr>
<td>AU2</td>
<td>0.534755695</td>
<td>AU15</td>
<td>0.4511296593</td>
</tr>
<tr>
<td>AU4</td>
<td>0.5820565833</td>
<td>AU17</td>
<td>0.4823638797</td>
</tr>
<tr>
<td>AU5</td>
<td>0.7306297402</td>
<td>AU20</td>
<td>0.5740361957</td>
</tr>
<tr>
<td>AU6</td>
<td>0.7617512649</td>
<td>AU23</td>
<td>0.2497179056</td>
</tr>
<tr>
<td>AU7</td>
<td>0.8746483787</td>
<td>AU25</td>
<td>0.8017179065</td>
</tr>
<tr>
<td>AU9</td>
<td>0.4406325761</td>
<td>AU26</td>
<td>0.5599466488</td>
</tr>
<tr>
<td>AU10</td>
<td>0.7358301909</td>
<td>AU45</td>
<td>0.2534699927</td>
</tr>
<tr>
<td>AU12</td>
<td>0.9598882938</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparing with the old dataset, it is obviously that AU12 keeps the highest sd, AU45 and AU23 are still the lowest two. We can assume that the characteristics of AU value’s distribution are shared between different datasets. And now, we need to pick data and adjust the dataset to bring the sd of AU23 and AU45 into an acceptable level.
To know how to raise their sd, we need to know the actual distribution of AU values. Figure 6 shows the distribution of AU12, which having highest sd (0.960), and AU45 which having second low sd (0.253). Blue bars are for AU12 and green bars are for AU45.

![AU12 and 45 of EmotioNet](image)

**Figure 6 Distribution of AU12 and AU45 of EmotioNet**

We can observe that no matter which AU, their distributions are almost the same. Most of the data concentrated on value 0, after a quick drop on value 0.5, the count drops slowly while the AU value increases. The different between a high sd AU and a low sd AU is that, the zero to non-zero ratio of a high sd AU is lower than a low sd AU. Thus, an image with non-zero value of AU23 and AU45 will have the highest priority to pick.
Table 6 shows the final sd of AUs of the dataset that used in the second experiment.

**Table 6. sd of AU values of dataset used in experiment 2**

<table>
<thead>
<tr>
<th>AU</th>
<th>sd</th>
<th>AU</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU1</td>
<td>0.6673449852</td>
<td>AU14</td>
<td>0.7740528662</td>
</tr>
<tr>
<td>AU2</td>
<td>0.5226214359</td>
<td>AU15</td>
<td>0.6419011294</td>
</tr>
<tr>
<td>AU4</td>
<td>0.6764352626</td>
<td>AU17</td>
<td>0.6432167802</td>
</tr>
<tr>
<td>AU5</td>
<td>0.6863843276</td>
<td>AU20</td>
<td>0.664781848</td>
</tr>
<tr>
<td>AU6</td>
<td>0.8180365373</td>
<td>AU23</td>
<td>0.3926892916</td>
</tr>
<tr>
<td>AU7</td>
<td>0.9744852839</td>
<td>AU25</td>
<td>0.7526741685</td>
</tr>
<tr>
<td>AU9</td>
<td>0.6296460272</td>
<td>AU26</td>
<td>0.6443904361</td>
</tr>
<tr>
<td>AU10</td>
<td>0.7899694741</td>
<td>AU45</td>
<td>0.4483277744</td>
</tr>
<tr>
<td>AU12</td>
<td>0.8728716134</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Similar as experiment 1, we picked 200,000 images to train the model. In Table 6, we can see that the sd of AU23 raised to 0.39, and sd of AU45 raised to 0.45, which is reach an acceptable level of around 0.4, just like AU9, AU15 and AU17 in the old dataset of Table 4.

Also, we had reported that AU17 also needs to be improved in interim report, and the sd of AU17 is raised from 0.41 of Table 4 to 0.64 of Table 6.

Besides of the training data of 200,000 images, we also picked 60,000 images randomly for testing.
3.2.3 Training

As stated in section 3.1.3, we tried to setup a virtual machine with more virtual cores and memories. However, since Google Cloud Platform limited the CPU and memory of a machine with a NVIDIA Tesla K80 GPU, we also need to upgrade the GPU to NVIDIA Tesla P100. Table 7 is the whole specification of the virtual machine.

Table 7. Specification of the virtual machine used in experiment 2

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>16 virtual CPUs</td>
</tr>
<tr>
<td>Memory</td>
<td>104 GB</td>
</tr>
<tr>
<td>GPU</td>
<td>1 x NVIDIA Tesla P100</td>
</tr>
</tbody>
</table>

This time, the training process last about 1.5 days (34 hours), which is much better than that of experiment 1.

As mentioned in section 3.1.3, we avoided using a keep-alive connection to train in experiment 2. Instead, we make use of the "screen" Linux command, which can detach the training process from current session, and resume it on other new session.
3.3 Software Implementation

A server-client approach is used in the deliverable. In this way, the setting and installed library on the virtual machine can be reused, and the specification requirement of the client side can be minimized. We had built a web demo application with the result of experiment 1, and an Android mobile app for experiment 2.

3.3.1 Web Application Implementation of Experiment 1

3.3.1.1 Client implementation

Python program and web application were two implementation method considered during the design of the client. Web application was chosen for the purpose of easy deployment. Plain HTML and Javascript is used and the deliverable is hosted on the FYP account.

At first, it was planned to crop the face image with OpenCV.js on the frontend before sending the request to minimize the request size. However, OpenFace was not installed in the client and was not able to be used to audit the result as in data processing stage. Therefore, the quality of face detection would be poor. Thus, the client was designed to send two raw images to the server.

The response from the server consists of 4 integers representing the position of the face (x, y, width, height), and a base 64 string of the result image data in 128*128 as shown in Figure 7. According to this information, only the area of the face will be updated with the new expression.
Figure 7. Sample response body from the server

There are two demo pages, AU editor and facial expression transfer. There are 17 sliders in the AU editor, representing the 17 AUs. User can edit the AU of the image by inputting the image and AU values. Figure 8 is a screenshot of the demo editor.

Figure 8. UI of AU editor of the web demo
The second one is facial expression transfer. The UI as shown in Figure 9 is relatively simple when compared to the AU editor. The user can upload two images and apply the facial expression of one to another.

**Figure 9. UI of facial expression transfer of the web demo**
3.3.1.2 Server implementation

NodeJS is used to manage internet requests, and Python to provide the actual service. The Python script is modified from the testing script of GANimation, it can detect faces and crop images as mentioned in 3.2.1, and use the GANimation modal to perform facial expression modification. Since it takes times for the Python program to load the GANimation modal, the Python program is pre-started, and the modal is preloaded such that the response time can be minimized. It communicates with the NodeJS program using socket.

Then, AU intensities detection is required from another image. To achieve it, the program will start a subprocess of OpenFace, and check for its output.

The actual flow of the server handling requests is described in Figure 10. The NodeJS server and the Python program will keep running and waiting for requests. The OpenFace program will only be fired whenever it is needed.

![Figure 10. Flow of the server of web application handling a request](image-url)
3.3.2 Mobile Application Implementation of Experiment 2

3.3.2.1 Mobile Application Implementation

iOS and Android are two popular mobile OS. We had chosen to build an Android application because building an iOS application needs to run XCode on a machine running macOS, and not all of our members own one. While the IDE of Android application development, Android Studio can be executed on either Windows, macOS, or even Linux systems, which is suitable for our team to cooperate.

As same as the web demo, the mobile application contains two functions: AU editing and facial expression transfer. What’s more, it can generate an animated GIF image for facial expression transfer. Figure 11 is the UI of the facial expression transfer.

![Figure 11. UI of facial expression transfer of the mobile application](image)

Submit button modified from icons by Freepik from www.flaticon.com

There are three views, expression image, face image and result image (From left to right of Figure 11). User can swipe left and right to switch between these three screens. The left button is used to upload image on expression image view and face image view, and disabled on result image view. The middle button is used to take photo on expression image view and face image view, and start sending to server on result image view. The right button is used to save the image of current view into local storage. The left most “GIF” switch is used to determined generating an animated GIF or static image. An animated GIF of switching
between original expression and modified expression will be generated if the switch is lighted on.

Figure 12. UI of AU editor of the mobile application

Icons modified from icons made by Freepik from www.flaticon.com

Figure 12 is the UI of AU editor. As same as that of facial expression transfer, there are three buttons, upload image, take photo and save image on the bottom of the screen. There is a circular scrolling list of the 17 AUs. After scrolling to the AU, and press the icon in the middle, an ark slider will be toggled. The corresponding will be modified if the user changes the value of the slider. Also, the value will be saved, and the user can combine the changes of different AUs.

Regarding the data flow, it is inefficient that sending the whole image whenever a change is made. Therefore, the whole image will only be sent to server once for face detection. After that, the mobile application will crop and resize the face image internally according to the server response, and it will be sent to server with new expression every time instead of the original image to minimize the network load.
3.3.2.2 Server implementation

In the server of the web application, NodeJS is used to handle network request. In the server of the mobile application, we rewrite the API handler with Python by using the Flask framework, so that we can combine the API handler and AU modifier program into one, and do not need to communicate with socket. The new flow of the server handling requests is described in Figure 13.

The process of fetching AU intensities is kept. OpenFace is fired as a child process and the output of AU intensities is read from an output csv file.

![](image)

**Figure 13. Flow of the server of mobile application handling a request**

Regarding GIF generation, the server will detect the AUs from both expression image and face image. And then a sequence of each AU between both images will be computed for the transition images. The server will then apply all sequences and generate multi images, and combine them into an animated GIF. The starting frame and ending frame will be repeated to last longer, so that the user can clearly recognize the transfer result.
4. Results

In this section, the results of the GANimation training process as well as the application of the model in facial expression transfer will first be described.

4.1 Loss values of tensorboard

GANimation is implement with Pytorch and apply tensorboardX, which makes the loss values can be visualized by tensorboard even though the model is not trained by Tensorflow. Since experiment 1 and 2 have a similar training data size, they both took about 6,000,000 training steps. As mentioned in section 3, the training process of experiment 1 took more than 224 hours, and experiment 2 took 34 hours. In this subsection, the loss values of the two experiments will be compared. The following graphs are all in a pair, the left one is from experiment 1 and the right one is from experiment 2. The actual value is marked as light orange while the smoothed value is in dark orange. The comparation will be done base on the smoothed value because the actual values are dispersed hard to compare.

![Figure 14. Loss values of output AU values for the generator](image)

From Figure 14, a decrement in loss value of AU values for the generator can be observed on both graphs which indicates an improvement of the performance of the generator during the training process. Overall, loss value of experiment 2 is higher than that of experiment 1. Which means experiment 2 may has a worse result than experiment 1 in this aspect.
Figure 15. Loss values of the attention mask output of new generated image for the generator

Figure 15 shows the loss values of the attention mask of generated image. Two graphs have a similar pattern and no huge difference. They both keep decrease slowly which means they keep improving.

Figure 16. Loss value of the attention mask output of real image existing in the training dataset for the generator

Figure 16 show the quality of the attention masks of the real image existing in the training dataset. While the loss value of experiment 1 decrease stably, the value of experiment 2 once reach 0.95 and bounced back to 0.098. It may be over-trained and should be stopped at step 4.500M, or it will decrease again like a sin curve if we keep training it.
“mskd_fake” means “masked fake image”, which is the result generated image. Both experiments give similar pattern again. Although the absolute value of loss of experiment 2 is large than that of experiment 1, the flow of it is closing to 0, while experiment 1 is moving far from 0. Which means further train on experiment 2 may improve it, but further train on experiment 1 may be over.

Figure 18. Loss value of AU value discrimination (d_real) and discriminating new generated image (d_fake) for the discriminator
Figure 17 shows the performance of the discriminator in fake image discrimination and real image discrimination. They are highly related to 16. While the loss value of real image discrimination conducted by the discriminator and the loss value of fake image generation conducted by the generator are directly proportional with each other, both of them are in inverse proportion with the loss value of fake image discrimination of the discriminator no matter in experiment 1 or 2.

As same as the graphs of experiment 2 in Figure 16 and 15, there is a peak closest to 0 on step around 4.500M. Maybe it is better to stop training there.

![Graphs of loss values](image)

**Figure 19. Loss value of discriminating real image existing in training dataset for the discriminator**

In Figure 18, the loss value of discriminating real image keep decreasing and means they keep improving. However, at the end of experiment 2, it seems starting to bounce back. Again, it may be better to stop it earlier.
4.2 Result outputs

Although the result of loss value analysis is not so good, experiment 2 can still outer perform experiment 1 because generative job cannot just valuate by data. In this section the result of two experiments will be compared and limitation will be introduced.

4.2.1 Comparation of both experiments

![Sample results of altering AU17, AU23 and AU45 of experiment 1](image)

**Figure 20. Sample results of altering AU17, AU23 and AU45 of experiment 1**

Figure 19 is the sample results of altering AU17, AU23 and AU45 of experiment 1. There are 4 images, original image, altering AU17, altering AU23 and altering AU45 from left to right. When altering AU17 Chin raiser, there is noise near the chin. Which means the model knows where to modify (correct attention mask) but do not know how to change it (wrong color mask).

As mentioned in section 3.1.1 and 3.2.1, the dataset used in experiment 1 has a low sd on AU23 Lip tightener and AU45 Blink. We can see the lip of the right 2 image of AU23 is tightened a bit, but still not significant. And almost no changes can be observed from the right most image of AU45.

After the improvement of datasets, experiment 2 gave the results of Figure 20, 21, and 22.
Figure 20 shows altering AU17 Chin raiser with our mobile application. It can be observed that the chin of the image being rough naturally. The second left image of Figure 19 is modifying the chin to be rough too, but the quality of it was low and it is noise in the view of human. On the other hand, Figure shows a natural generated image. The improvement of dataset stated at section 3.2.2 actually helps, and it also means that the discriminator of GAN works better than that of experiment 1, as discriminator is the key of GAN to generate a realistic input.
It is a comparison of a generated image of AU23 altered by the model of experiment 2 on the left, and another generated image with value 0 of AU23 on the right in Figure 21. By observation, the lips of image of intensity 0 in AU23 is dark, thick vertically and narrow horizontally. On the other hand, the lips of image of high intensity in AU23 is light in color, thin vertically and wide horizontally. When comparing to the image on the second right in Figure 19, this change is obvious. Which proves that the sd contral of section 3.2.2 improves the training result.
Figure 20 shows altering AU45 Blink with the second model. Eyes in the original image are totally disappeared. Significant change is obvious when comparing to the right most image of Figure 19.

Again, as stated at section 3.2.2, an improvement of distribution of AU45 in the new dataset is performed, and it helps to improve the result. In Figure 22, the slider is not crossing the middle point, but the eyes in the image are already closed. By Figure 5 from section 3.2.2, the Blink intensities are concentrating in the low zone, the highest value is only 3.5 from 4 images. Therefore, a high value in Blink is meaningless, and an AU intensity of about 2 is enough for closing the eyes.
4.2.2 Limitation

Although experiment 2 had shown a huge improvement from experiment 1, there is still its limitation. Which is mainly about generating new eyes and removing tooth.

4.2.2.1 Eyes generation

As shown in Figure 22 of section 4.2.1, the model performs well at removing eye to perform blink. However, on the other hand, it is not good at creating non-exist eye. Eyes generation is strongly related to AU45 Blink. A low intensity of AU45 means no blink, eyes are opened and eye balls are shown in the image. If intensity is low, lid will cover the eye and eye balls will not be shown in the image.

![Figure 24. Reverse result of generating low Blink intensity](image)

Figure 23 is the result by reverting the intensity of AU45 to zero of Figure 22. We can see that it tries to create white and black near the eyes. However, it is blurred, and the quality is low.
4.2.2.2 Tooth generation and removal

![Image of mouth closing](image1)

**Figure 25. Expression image of mouth closing**

In the following section, we will discuss about tooth generation and remove, which is mainly affected by AU25 Lips part. A low intensity of AU25 means a closed mouth, and high intensity means separated lips. If lips are separated, mouth is opened, and tooth are shown in the image. Figure 24 is an image of Donald Trump and it will be used with the image of Barack Obama used in previous sections to valuate the model.

![Image of Donald Trump and Barack Obama](image2)

**Figure 26. Result of tooth generation**

Figure 25 is the result of transferring a smile face to the face of Donald Trump and Barack Obama. We can see that tooth is created with an acceptable quality.
Figure 27. Result of removing tooth from generated images

Figure 26 is the result of applying the expression of Figure 24 to the images of Figure 25. We can see that tooth is completely removed which is a good result. We guess that the reason that quality is good when applying expression in a generated image is that reproducing the original image base on a generated image is the way that how the generator is trained, as mentioned in section 2.5.
Figure 28. Result of removing tooth from real images

Figure 27 is the result of applying the expression of Figure 24 to two real images, the left two are real image and right 2 are generated image. The tooth is not fully removed and looks pink. It is because the white color of tooth and red color of lips are overlapped. In fact, if we view the generated image of Donald Trump far away, it looks like a success transfer that only lips left but no tooth. In this case, the color mask works well and realistic color is generated, but the attention mask is not good and the color from color mask cannot apply to the image properly.

To summarize, it seems that the model works better with an original image of lower AU intensities, such as opened eyes (low AU45 Blink) and closed mouth (low AU25 Lips part).
5. Conclusion

The goal of this project is to implement a facial expression transfer software, which is fulfilled with the web application as well as the mobile application. The final product is built on top of GANimation, OpenFace 2.0 toolkit, and the EmotioNet database. OpenFace is used to mark AU intensities of the images from EmotioNet, and then GANimation model is trained with the marked data. After the modal is trained, it is ready to perform facial expression transfer. For an expression image and a face image, we fetch the AU intensities of the expression image by using OpenFace, and then pipe the AU intensities and the face image into the GANimation network, and a face image with new expression is generated.

Within this project, we have explored the FACS conditioned scheme, and a modern deep learning approach GAN for generative tasks. At first, we planned to implement a standalone mobile application. However, we finally found that there is too much works for moving GANimation and OpenFace 2.0 to execute on a mobile device. As this is not the focus of our theme facial expression transfer, we gave up the standalone plan and use client-server approach in the final product.

Through the two training processes, we have learnt the importance of data pre-processing, which is called the most important and most time-consuming part of machine learning. In the first experiment, we did not filter any data because the base of data is not large enough. It results in the bad performance on AU17, AU23 and AU45. After we granted the access to EmotioNet through our supervisor, the base of data is large enough for us to choose and discard. Finally, we successfully raise the standard deviation of AU17, AU23, AU45 from 0.41, 0.20 and 0.18 of experiment 1 to 0.64, 0.40 and 0.45 of experiment 2. It turns out that this improvement brings positive impact to the training result.

Facial expression is a high-level concept. Although FACS can helps representing facial expression, the discrete Action Unit concept limited the variation of facial expressions that it can represent. Also, OpenFace can only detect a few of AUs, which is not fully utilized FACS. Using deep learning to represent facial expression without FACS may come up with a new representation system that is more flexible. Also, the meaning of GAN is far larger than achieving difficult generative tasks, but also generating data for further machine learning. Therefore, transferring facial expression using deep learning without FACS can be a direction of future research.
6. References


