

Detecting Human Emotions From 3D Mesh

Jialin Yang

U5894100

A report submitted for the course
Comp 4560 Advanced Computing Project
Supervised by: Prof. Tom Gedeon, Dr. Zakir Hossain
and Moshiur Farazi
The Australian National University

October 2019

© Jialin Yang 2019

Except where otherwise indicated, this report is my own original work.

Jialin Yang
25 October 2019

Acknowledgments

Many thanks to my supervisor Prof. Tom Gedeon for gave me the chance to participate in such an ambitious project with the best researchers in the world. The whole journey was fantastic and benefited me a lot.

Special thanks to Dr. Zakir Hossain and Moshiur Farazi for guided and inspired me. Your patient help is like the light in the dark, guiding me forward. It would be impossible for me to complete this project without your participation.

Next, I would like to take this opportunity to thank my parents for their unconditional support and love in the last 22 years.

Last, I would like to thank my girlfriend Shirley Yuan for taking care and support me unconditionally in the last 8 years. You make my life marvelous!

Abstract

Ever since the artificial neural network model had been introduced in 1943 by Warren McCulloch and Walter Pitts[McCulloch and Pitts, 1943], it had been widely applied to the tasks which standard programming cannot handle, such as image classification or linear regression. In this project, we would focus on the human emotion detection task using the artificial neural network. The standard human emotion detector had two major components, the data-set, and the classifier. The data-set contained many images of human faces with emotions, and it would be the input for the classifier. As for the classifier, the convolutional neural network was commonly used. Some of the traditional human emotion detectors had achieved an over 90% accuracy on some data-set, but they performed terribly when the quality of input image was low, or the input image was a side face of a human. Thus, we proposed the 3D mesh generating process to improve the traditional neural network. Our 3D mesh pre-processing step would take all input images and generate a 3D mesh model for each of them. Even the input was a side face, the 3D mesh generator could still generate a 3D mesh with good quality. Then we adjusted the position and the angles of the 3D mesh and let it face toward the camera. After that, we fed those pre-processed images into the classifier. Our project aimed to examine whether introducing 3D mesh pre-processing steps would increase the overall performance of the human emotion detector used the artificial neural network. Our experiments tested a standard human emotion classifier with our proposed method implemented, and the results indicated that under majority situations, it would increase the overall performance of the classifier. However, when the model compressed or down-sampled the input images too much, using our pre-processing method would not be helpful. This is due to our pre-processing method had already eliminated most of the information that irrelevant to human emotion detection from the input image, so further compress or down-sampled would make useful information lost. Thus the performance of the classifier using our pre-processing method under those situations would be unsatisfied. In the future, researchers could test our pre-processing method on non-deep learning method, which would further prove the effectiveness of applying our method.

Keywords: Artificial neural network, Convolutional neural network, 3D mesh generator, human emotion detection, ablation study

Contents

Acknowledgments	iii
Abstract	v
1 Introduction	1
1.1 Problem Statement and Motivations	1
1.2 Project Scope	2
1.3 Report Outline	3
2 Background and Related Work	5
2.1 Artificial Neural Network and Supervised Learning	5
2.2 Image Classification Task Using Convolutional Neural Network	6
2.3 Human Emotion Detection	8
2.4 3D Face Alignment	9
3 Design and Implementation	11
3.1 Experiment Design	11
3.2 Data-set	12
3.2.1 Cohn-Kanade AU-Coded Expression Data-set	12
3.2.2 Data-set Preparation	13
3.3 Image Pre-processing	14
3.3.1 Texture Mapping	14
3.4 Rendering	15
3.5 Ablation Study	16
4 Experimental Methodology	19
4.1 Programming Language	19
4.2 Image Pre-processing	19
4.3 Convolutional neural network settings	20
4.3.1 Image crop and compression	20
4.3.2 Data and Label matching	21
4.3.3 Activation function	22
4.4 Ablation study setting	22
4.4.1 Base model	23
4.4.2 Hyper-parameters in Ablation study	23

5	Results	25
5.1	Base model	25
5.2	Ablation study	26
5.2.1	Compression Size	26
5.2.2	Kernel Size	27
5.2.3	Max pool size	28
5.2.4	Adding normalization	29
5.3	Summary	30
6	Conclusion	31
6.1	Conclusion	31
6.2	Future Work	32
	Bibliography	33
	Appendix I	35
.1	Project Title	35
.2	Project Description	35
.3	Project Outcome	35
	Appendix II	36
	Appendix III	39
.4	Code files description	39
.5	Correctness test	39
.6	Description of how experiments were conducted	40
	Appendix IV	41

List of Figures

2.1	The structure of a standard artificial neural network	6
2.2	The architecture of a standard convolutional neural network, which consists of two convolutional layers, two max pooling layers, and one fully connected layer	7
2.3	Some working examples of 3D face alignment where the top was images of different people facing different directions ,and the bottom was plotted face shapes in 3D space by 3DDFA.	9
3.1	Proposed Experiment Process for our project. Input two data-sets into the classifier and run the classifier twice with each of them. Record the results and modify the classifier before repeat the above process.	12
3.2	Example of sequence in Cohn-Kanade data-set. The sequence contains a poser's emotion from the neutral(leftmost) to the peak(rightmost). . .	14
3.3	Example of 3D mesh model with and without texture mapping	15
3.4	The process of the ablation study in our project. We select a base model at first and test its prediction accuracy with the two data-set. Next, we choose a range of hyper-parameters we would like to test and repeat the test for each of them.	17
4.1	The process of the image pre-processing step in our project. We first select the peak image for each sequence in the Cohn-Kanade data-set and organize those peak images into a folder. Pick one image from this folder and generate its 3D mesh model. Repeat this process until all images have their 3D mesh model generated.	20
4.2	Examples of cropped images from original and rendered data-set. Original image was cropped to 480*480 while the rendered image was cropped to 600*600	21
5.1	Test result of model using original and rendered data-sets for compression size from 30 to 150	26
5.2	Test result of model using original and rendered data-sets for kernel size from 1 to 9	28
5.3	Test result of model using original and rendered data-sets for max pool size from 1 to 5	29

List of Tables

3.1	An example of Cohn-Kanade dataset, where "Subject" represent the poser's Id and "Session" indicate the sequence index. FACS code determines the emotion but needs to be converted to numeric labels before training.	12
3.2	An example of The guide converting FACS code into Emotion terms . .	13
4.1	Hyper-parameters' settings for base model	23
4.2	The chosen hyper-parameters for ablation study in our project	23
5.1	Hyper-parameters' setting for base model	25
5.2	Test result of base model using original and rendered data-sets over 5 times	25
5.3	Hyper-parameters' setting for model under investigation of compression size	27
5.4	Hyper-parameters' setting for model under investigation of kernel size	27
5.5	Hyper-parameters' setting for model under investigation of max pool size	29

Introduction

1.1 Problem Statement and Motivations

Since Ivakhnenko and Lapa[Schmidhuber, 2015] had introduced the first functional multi-layers Artificial Neural Network in 1965, it had been widely applied to solve tasks that cannot be handled by traditional programming, such as data mining and pattern recognition. Traditional programming required programmers to know the solution to the problem in advance, and then they can implement these methods in code. Traditional programming worked well for some simple problems, but when the problem became complicated, such as image classification, this method no longer works because the programmer did not know the solution as well. In other words, the programmer's capabilities limited the performance of traditional programming. Things were different when using the artificial neural network. The artificial neural network did not require the programmer to know the solution to the problem in advance, and all they need to do was fed the artificial neural network with data and set the initial hyperparameters. It would discover the patterns behind the data and improve its model while training. Thus, unlike traditional programming followed user's instructions to solve problems, the artificial neural network analyzed the problem and came up with its solutions.

For human emotion recognition, the artificial neural network first learned the features and details of the human face from many samples and then used the learned knowledge to classify the emotions. Nowadays, some human emotion detection model had achieved over 90% prediction accuracy. Such models only considered the case where the input was the human's front face, and the classification result would be significantly affected if the input image condition changed. For example, an image contains only the side face of a human.

To get more details from only a side face and improve the performance of the human emotion detector under this condition, we introduced the 3D mesh generate pre-processing step for the standard human emotion detector. The 3D mesh generator was trained using the data-set, which contained human faces at different angles to make it learned how the human's eyes, nose, and lips located in 3D space. After training, the model could predict the other half of the face based on only one side

of the face. Furthermore, for some extreme situations, such as parts of human faces were covered by glasses or a mask, the 3D mesh generator can still predict the hidden part from the visible part of the face.

Since this 3D mesh generator could generate a 3D mesh model from face images under various conditions, we wondered whether introducing this pre-processing step would increase the overall performance of standard human emotion detectors and whether this step would become an essential part of emotion recognition. That was the motivation for our project, but we did not just stop there. If the 3D mesh generate pre-processing step has been proved to be effective, we could extend the emotion detector to detect emotions on a 3D object or even on live videos, which would make a big step toward the fully intelligent system. Imagine that the robot can detect changes in your expression and change the way they speak accordingly.

1.2 Project Scope

Our project aimed to examine whether introducing 3D mesh pre-processing steps would improve the overall performance of the Human emotion detector. To test our hypothesis, we set up a 3 steps experiment for this project. The first step is to set up a 3D mesh generator at first and apply it to all input image data. Once finished, we rotate these 3D mesh models and let them facing the front at the approximately same angle. After that, we use a software called "meshlab" to render all 3D models back to the 2D images, which would be the input for the next part. Meanwhile, we categorize all the original images into one folder while put all render images into another folder. Prepared the data-set in a separate folder would help us test and conclude later steps. One thing that needs to notice when generating a 3D model of the input image is, due to low resolution or noisy background of some images, the 3D model generate was not good enough to reveal the emotions on the original face. If that happens, we need going back to the original image and change the brightness and sharpness of it, which would output a much better 3D mesh.

The second goal is to set up an emotion detector used the deep learning approach. In this project, we choose the Convolutional Neural Network(CNN) for this task, which is one of the most common supervised learning approaches. Supervised learning is the machine learning task of learning a function that maps an input to an output based on given input-output pairs[Russell et al., 2010]. The general aim of using supervised learning is to train a model that maps an input to the expected result as close as possible. Choosing CNN for our task is because CNN could extract features from the image and pass the features to the fully connected layers before yielding the output. CNN has been widely applied in image classification tasks, such as the mnist handwritten word recognition task[Maitra et al., 2015]. In this step, the model would be trained based on the data-set prepared in the first step.

The third goal is to evaluate the result. Since we have two data-sets contain both the original image and rendered image, we then passing each data-set through the emotion detector mentioned above. During the process, we might change the hyper-parameters or adding new techniques to the emotion detector, but no matter what changes, we would test the new model using those two data-set. By doing that, if we can observe the performance of the emotion detector using the rendered image is always better than using the original image under different situations, we can then conclude that our pre-processing method is beneficial to the emotion detector.

In general, this paper can be understood as feasibility proves of using 3D mesh generator in emotion detection used deep learning approach task by comparing the performance between models with and without 3D pre-processing step.

1.3 Report Outline

After the introduction, there are another 5 chapters as listed below:

List of Chapters:

Chapter 2: Background and Related Work

In this chapter, we would like to introduce some project related background information for the reader to better understand the project. Meanwhile, we would also mention some previous studies or technologies that we implement in our project.

Chapter 3: Design and Implementation

In this chapter, we would first discuss how we designed the experiment in our project to test our hypothesis. Next, we would introduce the data-set we used for this project and how we pre-process the data-set before input. At last, we would talk about how we applied ablation study in our project and why it made our result robust.

Chapter 4: Experimental Methodology

In this chapter, we would talk about how we conduct the experiment, which involved the experiment environment, classifier setup, and deploy of ablation study.

Chapter 5: Results

In this chapter, we would show the results of our ablation study and related hyper-parameters settings. Besides, we would also compare and explain the results.

Chapter 6: Conclusion

In this chapter, we would talk about the conclusion we made from our experiments' results and what could be improved in the future.

Background and Related Work

This chapter provides preliminary background information about the project, which required to know beforehand to understand the project. In section 2.1, we introduced the Artificial Neural Network and its usage. In section 2.2, we talked about using Convolutional Neural Network for image classification task and its advantages compared to other methods. In section 2.3, We discussed the 3D face construction from the input face image or videos.

2.1 Artificial Neural Network and Supervised Learning

To understand the project, knowing what the Artificial neural network(ANN) is and how does it work are essential because the whole project was built on that. ANN is one of the bio-inspired products created by humans, and the idea of ANN comes from the architecture of the human brain[Schmidhuber, 2015]. Warren McCulloch and Walter Pitts published the first computational model of ANN in 1943[McCulloch and Pitts, 1943]. Their work transforms the workflow of the nervous system inside the human brain into a computational math model, which opened a new chapter for machine learning and gave a new solution to problems that cannot be done in the past, such as object recognition and classification. As shown in Figure 2.1, there are four components in a basic ANN model, which are neurons, weight, activation function, and propagation function. The single neuron would take a signal as input, followed by passing through the activation function before outputting the result. Neurons are connected, and each connection provides the output from one neuron as the input for the others[McCulloch and Pitts, 1943]. Besides, each connection is also assigned with a weight, which represents the relative importance between neurons[Zell, 1997]. One neuron could be connected with multiple inputs and output neurons. When the neuron has multiple inputs, the propagation function would take the weighted sum of all inputs and pass to the neuron. Since the weights would affect the importance of that input, we can adjust this weight to let the model output the result we expected, which is called the "learning." The back-propagation algorithm was proposed by multiple scientists in the early 1960s[Schmidhuber, 2015], and it was first applied to ANN by Werbos in 1974[Werbos, 1975]. The back-propagation algorithm calculates the output

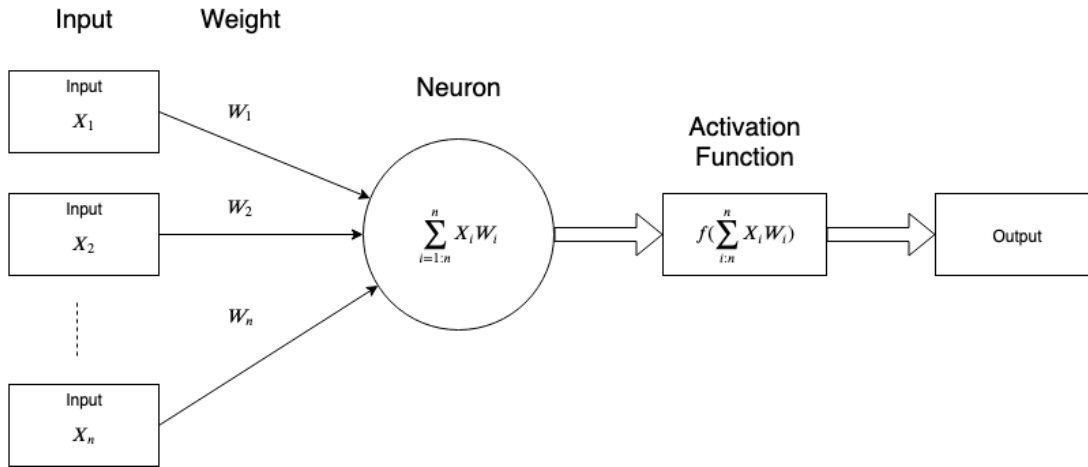


Figure 2.1: The structure of a standard artificial neural network

error and updates the weight based on that error from the last layers to the first layers. The whole process is recursive, iterative, and efficient when training the multi-layer neural network.

Supervised learning is a universal machine learning type that maps an input to the output based on example input and output pairs[Russell et al., 2010]. Generally speaking, a correct label is required to be given at first, and the result of supervised learning would train to match this label, which is the learning process. Some standard supervised learning methods are linear support vector machines (SVMs)[Cortes and Vapnik, 1995] and k-nearest neighbors (kNNs)[Altman, 1992], which had been successfully applied to dealing pattern-recognition problems in biology and medicine[Bzdok et al., 2017]. Our project used the supervised learning model, where emotion labels work as supervisors and inputs are the face images.

2.2 Image Classification Task Using Convolutional Neural Network

Traditional programming refers to the manually created program which takes input data and running the program on the computer before outputting the result[Parthasarathy, 2019]. The program needs to be prepared beforehand to test, which means all logic needs to be understood at first before programmers encode into the program. However, when we have little knowledge about input data, use traditional programming would not help us solve the problem. For example, Given 100 images contains either cat or dog, create a program that identifies the animal on each input image. For these kinds of problems where we have little knowledge about the input data, using the deep learning approach is encouraged. Rather than traditional programming, the

deep learning approach takes both the input and output data and creates a program by learning from the input and output pairs. Thus, when facing a problem that requires the computer to learn the features and programming itself, choosing the deep learning approach is a wise decision.

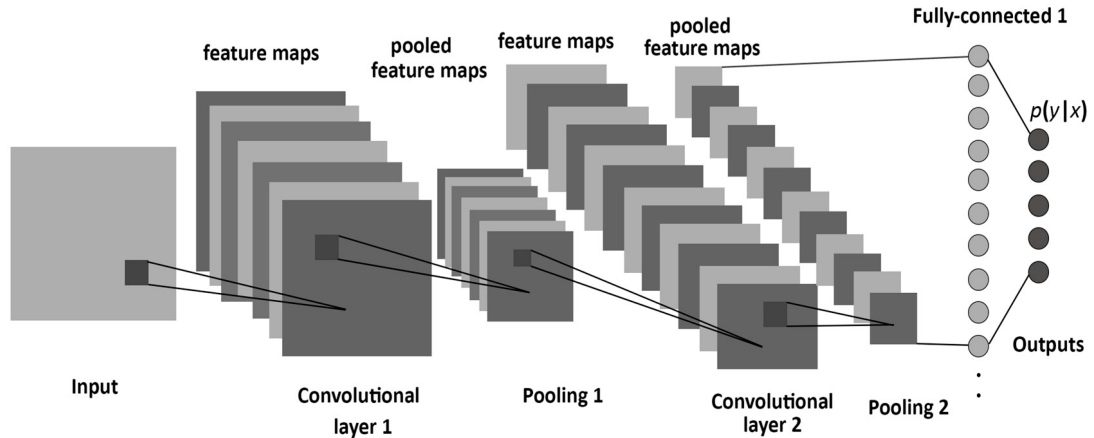


Figure 2.2: The architecture of a standard convolutional neural network, which consists of two convolutional layers, two max pooling layers, and one fully connected layer

There are different kinds of deep learning approach out there, but each of them has their strengths. When the problem we are solving using images as input, we normally choose the convolutional neural network(CNN) as our solution[Collobert and Weston, 2008]. CNN was inspired by Hubel and Wiesel’s early research on the cat’s visual cortex[Wang et al., 2017]. Compared to the basic neural network, the Convolutional neural network have extra convolutional layers and pooling layers[Xing and Yang, 2016]. A convolutional layer contains a set of filters, whose height and weight are smaller than those of input image. Those filters is convolved with the input image to create an activation map made of neurons[Ke et al., 2018]. After that, all activation map are stacking along the depth dimension[Ke et al., 2018]. Since the width and height of filters are smaller than the input image, each neuron in the activation map is connected to a small local region of the input image[Ke et al., 2018]. This feature of the convolutional layers allows the CNN to learn filters which maximally respond to a local region of the input, which helps the CNN identify the correlation between pixels in the original image. In other words, convolutional layers could extract features from the input image and train the parameters inside filters to better generalize those features. Once the features have been learned, we could pass the features into fully connected layers to help us classify those features into classes we expected. Above are the reasons why we were using CNN for image classification tasks rather than traditional programming.

2.3 Human Emotion Detection

In the previous section, we went through how we use CNN to classify images and why CNN is a useful tool for that task. In this section, we would further break down this task and introduce some backgrounds about emotion recognition.

When recognizing the emotion, we observed that human was reporting consistent recognition results, which indicates that there is a similar feature pattern behind the data we percept. Emotions were expressed in a combination of facial movement, body movement, and voice, and we could rely on this information to detect emotion.

Let's first talk about emotion detection using facial information. Humans and other animals have excellence control in their facial muscles. Emotions were expressed as a patterned movement of facial muscles that related to internal and affective states[psy]. Since the movement of facial muscles was patterned, we could train the neural network to classify human emotions by learning them. As mentioned in the previous part, CNN was widely applied to such training tasks. Convolutional and pooling layers extract and select the most significant features from the image before using fully connected layers for classification. There were many emotion detection papers applied to CNN or CNN based neural networks, and the results were outstanding compared to other methods. For example, Jie Wu's model had achieved 94.64% accuracy for CK+ data-set and 73.112% for FER2013 data-set[WuJie1010, 2019], which proves the effectiveness of CNN in this task.

Besides facial movement, vocal information can also be used for emotion detection. Humans and other animals react rapidly to conspecifics' signals to survive[Sauter and Eimer, 2010]. Disa and Martin's research on "detection of emotion from human vocalization" showed the evidence that non-verbal vocalization, such as screams, would trigger the emotion detection within 150 milliseconds, which was as fast as detecting emotions from facial images. This observation gave us evidence that common supramodal brain mechanisms may be involved in the rapid detection of affectively relevant visual and auditory signals. To further prove the effectiveness of using vocal information to classify the emotions, more experiments are required.

Body movement and gestures could also be used to recognize emotions. When we were expressing some emotions, despite our facial movement, we also did a lot of body movements and gestures. Ginevra, Santiago, and Anotonio's research had shown us the possibilities of using body information to classify emotions. Rather than analyze and categorize body or gestures' shape, they were using non-propositional movement qualities (e.g., amplitude, speed, and fluidity of movement) to refer emotions[Castellano et al., 2007]. Their experiment considered 4 emotions (Anger, Joy, Pleasure, and Sadness), and the experiment result showed the effectiveness of classifying emotions using non-propositional movement qualities. Compared to emotion detection using facial information, this method's performance is still unsure when

more emotions considered.

Above, we have discussed the possibilities of using facial movement, vocal or bodygestures to classify human emotions. Though vocal and bodygestures could be used for emotion classification, we chose using facial information to do this task because its performance had been tested and proved by many researchers and facial data was easier and cheaper to collect compared to vocal and non-propositional movement qualities.

2.4 3D Face Alignment

Face alignment is a process of transforming a face model to fit the face image, which is a necessary pre-processing step for many face analysis tasks and CNN[Zhu et al., 2019], face alignment accuracy has been improved significantly but this was built on the assumption of yaw angle is below 45 degrees, and all face landmarks are displayed. This finding implies that if the face in the image is not facing ultimately toward the camera, the face alignment result would be wrong. In 2019, Xiangyu Zhu introduced a brand new 3D total solution for face alignment called 3DDFA[Zhu et al., 2019], which enabled the face alignment applies to those images whose landmarks were hidden or yaw angles were greater than 45 degrees(up to 90 degrees). Inside 3DDFA, there is a trained CNN which has learned features from thousands of faces. If some of the landmarks of input face were missing or covered, CNN would be able to "imagine" the missing or covered parts, which make the 3D model generate accurately fit the image. 3DDFA was also extended to face alignment of the face in the video. 3DDFA is a newly introduced method and has never been applied to the emotion detection field. In the emotion detection field, the detection result is terrible when the input face image is not facing the camera, where some landmarks or features missed. However, applying 3DDFA to the traditional emotion detection task would solve the problem mentioned above by fit a 3D model to the input image. Our project is to test whether 3DDFA would improve the performance of the traditional emotion detection method.

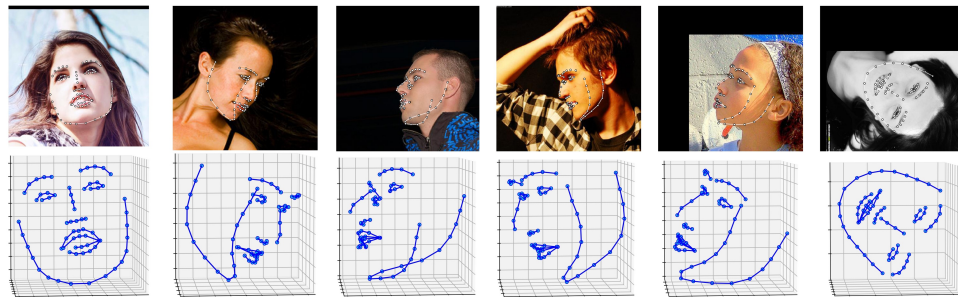


Figure 2.3: Some working examples of 3D face alignment where the top was images of different people facing different directions ,and the bottom was plotted face shapes in 3D space by 3DDFA.

Design and Implementation

This section would address two topics, how the experiment was designed to test our hypothesis effectively and reliably and how we implement those ideas into real code.

3.1 Experiment Design

Our project is to examine the hypothesis, whether introduce the 3D mesh generating process would increase the overall performance of human emotion detection. To examine this hypothesis, we designed the experiment process, as shown in Figure 3.1. At first, we prepared two data-sets. One is the original face images without any pre-processing, and another is the face image rendered from the 3D mesh. Once the two data-sets are ready, we then forward them to the classifier, the convolutional neural network(CNN). Here noticed, The data-sets would be fed separately to the classifier rather than mix together. When the classifier loads all the images from the input data-set, it splits these images into the training set and validation set. The classifier trained itself using the train set and predicted the emotions of the images in the validation set. Then we compared the predict emotions with the labeled emotions and calculated the accuracy in percentage before recorded. Once finished, we set the classifier to the original state before repeating the same process for another data-set.

Once the training for both data-sets finished, we compared their prediction accuracy and recorded the result. These steps mentioned above was one complete cycle of our experiment loop. Next, we move back to the classifier and update the CNN model before repeat the whole "fed-train-predict-compare" process again. This process would be repeated until there was no further improvement in the performance of the classifier. To ensure there were enough loops for us to analyze, we manually select a base model for the classifier and improve it uniformly per loop. From this experiment, we were aiming to observe how data-set rendered from 3D mesh would affect performance under different models. By observing the comparison with the performance of the model using the original image under different situations, we can conclude the effectiveness of using 3D mesh pre-processing in the human emotion detector. Since the experiment was conducted using different classifier models, the

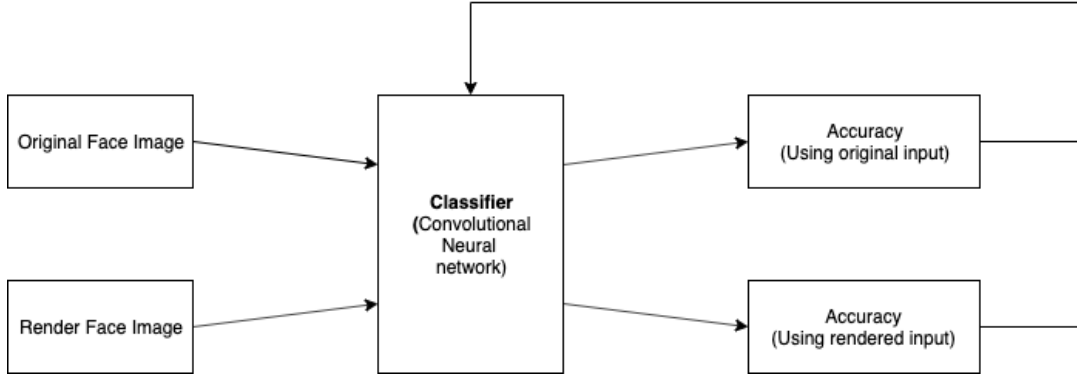


Figure 3.1: Proposed Experiment Process for our project. Input two data-sets into the classifier and run the classifier twice with each of them. Record the results and modify the classifier before repeat the above process.

findings and conclusion we made could be extended to general situations.

3.2 Data-set

3.2.1 Cohn-Kanade AU-Coded Expression Data-set

In 2000, the first version of the Cohn-Kanade(CK) data-set was released to promote research into automatically detecting individual facial expressions[Lucey et al., 2010]. The data-set includes 487 sequences from 97 posers. Each sequence starts with a neutral expression and ends with peak expression of that emotion[ck]. In our project, we were aimed to generate the 3D model from a face image. Thus only the peak expression was picked from each sequence. Although this action made the training and 3D mesh generating process more manageable, the lack of training samples would significantly lower the classification performance of the proposed model. Since we were not meant to develop a perfect emotion detector but to examine our hypothesis "Whether introducing 3D mesh pre-processing would increase the performance of the emotion detector", this disadvantage could be discarded currently.

Subject	Session	FACS code
10	001	1+2+20+21+25
11	001	1+2+25+27
14	002	1+4+15+17+45b
26	003	2 4+R7+14+17d+24

Table 3.1: An example of Cohn-Kanade dataset, where "Subject" represent the poser's Id and "Session" indicate the sequence index. FACS code determines the emotion but needs to be converted to numeric labels before training.

Emotion	Prototypes	Major Variants
Surprise	1+2+5B+26	1+2+5B
	1+2+5B+27	1+2+26
		1+2+27
		5B+26
		5B+27
Fear	1 +2+4+5*+20*+25	1+2+4+5*+L or R20*+25, 26, or 27
	1+2+4+5*+25	1+2+4+5*
		1+2+5Z, with or without 25, 26, 27
		5*+20* with or without 25, 26, 27
Happy	6+12*	
	12C/D	

Table 3.2: An example of The guide converting FACS code into Emotion terms

There were a total of 6 emotions recorded in the CK data-set, surprise, fear, happy, sadness, disgust, and anger. For each sequence, a combination of AU scores was attached, as shown in Table 3.1. FACS code (a combination of AU scores) cannot be directly used as the labels to train the emotion detector, so we have to convert FACS code into emotion terms before training.

In the original CK data-set, a complete guide of converting FACS code into emotion terms was given (example shows in Table 3.2). By matching the FACS code into the closest form of prototypes, we can determine the emotion label for that sequence. In our project, we use the numeric number to represent the emotions, 0 for the surprise, 1 for fear, 2 for happy, 3 for sadness, 4 for disgust, and 5 for anger. Repeating the conversion process for the whole 487 samples, we then obtained the labels that could be used for model training.

3.2.2 Data-set Preparation

The CK data-set contains 487 sequences from 97 posers, and each sequence contains one single poser's emotion from neutral to the peak. One example was showed in Figure 3.2. Our project aimed to improve the performance of the human emotion detector when detecting emotions from one person's face image. At the current stage, we were not considered to read emotion from videos. Thus only the peak emotion image of each sequence was required. We manually selected the peak emotion image from each sequence and packed them into one folder, which would be the input for our experiment. By picking only the peak image from each sequence, we shrink

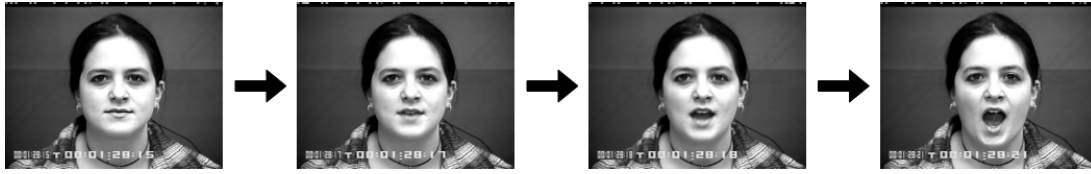


Figure 3.2: Example of sequence in Cohn-Kanade data-set. The sequence contains a poser's emotion from the neutral(leftmost) to the peak(rightmost).

the data-set size from 5876 to 487. Though it saved our time and efforts spent on pre-processing later, the sample number became insufficient. Insufficient samples meant we were impossible to train a perfect 6 emotions detection model.

3.3 Image Pre-processing

In this section, we would introduce how we pre-processed the original data-set, and this is the place where we applied the 3D mesh generating technique.

Since the 3D mesh generator can only handle one image in the meantime, we have to feed one image to the 3D mesh generator manually. Once the 3D mesh model was generated, we saved the model and fed the next image. The process had to be repeated until all images had to go through this process.

Since we were using the 3D mesh models to detect human emotions, we have to make sure the 3D mesh model clearly depicted the emotion showed on the original image. During the training, we noticed the 3D mesh generated highly relied on the qualities of the original image. The original Cohn-Kanade images were shot in 2002 when the camera resolution is relatively low. Thus the image qualities were not satisfied. Even though the qualities were unsatisfied, the majority of the images were good enough for the 3D mesh generator to construct a 3D mesh model. Some of the images were too dark, or the contrast between face and background was too abstract, so the 3D mesh model generated cannot depict the original emotions. To solve that, we went back to the original image and used the image editing tool to increase its brightness and contrast manually. We tested and modified the settings multiple times until we got a satisfied 3D mesh model. Once all the models were prepared, we then forward the 3D mesh model to the next rendering process.

3.3.1 Texture Mapping

The 3D mesh generator we used was good but not perfect. During the pre-processing, we observed some of the images, though they have a good quality, the 3D mesh models generated were still unsatisfied. Soon, we realize this was due to the emotions of the original image was not noticeable. As shown in Figure 3.3, the left image is

the original image of Elon musk, and the mid one is the 3D mesh model generated from it. It was evident that the 3D mesh model was not captured the emotion of the original image since the original image is not apparent enough for the generator to understand. Under these situations, we cannot use this 3D mesh model for training, and it was also not possible for us to change the structure of the 3D mesh generator. Soon, we came with the idea, texture mapping. Though the 3D mesh generator cannot locate the position of eyes and mouth correctly under this situation, it still can distinguish the face from the background. Using the face coordinates data, we can apply it to the original image and crop the texture within the coordinates before mapping this texture to the 3D mesh generated. The right image of Figure 3.3 showed the 3D mesh model after applying the texture mapping.

For the image only have the side face of a person, the texture showed on the original image was only for one side of the face. Under this situation, the 3D mesh generator would apply the mirror-symmetric of the visible side of the face to the other side of the 3D mesh model. Though the result was not natural and good enough for us, it was enough for the classifier to detect emotion from it.

To achieve texture mapping, we did not require to write any code to support it since the 3D mesh generator would automatically store the texture information with the 3D mesh model. When rendering the 3D mesh model into a 2D image, we would apply the textures.

3.4 Rendering

In this section, we would discuss how we were rendering the 3D mesh model to a 2D image. It is commonly asked by many people that why won't we input the 3D mesh model into the classifier directly? Doing that was due to insufficient data-set. As mentioned in the data section, each sequence of the Cohn-Kanade data-set contained one poser's emotion from neutral to peak, and we only extracted and used the peak emotion from each sequence. By doing that, we shrink the Cohn-Kanade data-set

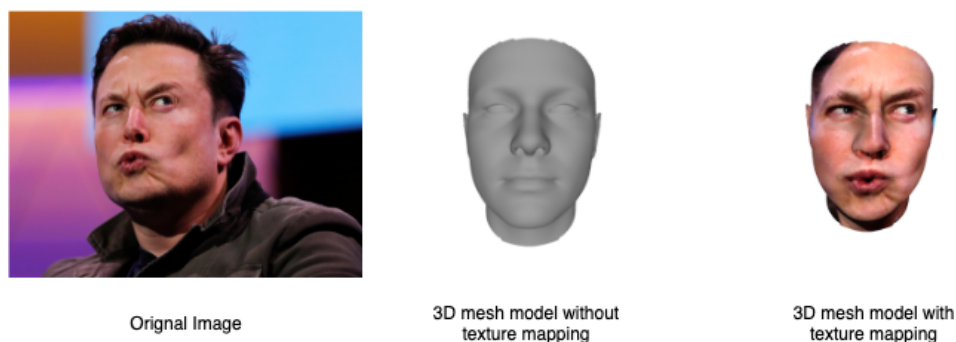


Figure 3.3: Example of 3D mesh model with and without texture mapping

from 5876 images to only 487 images. Though it decreased our training time and cost, the training samples were insufficient. Training a 3D model classifier required tons of training samples due to the massive amount of training parameters. Even the whole Cohn-Kanade data-set was used, it is still insufficient. Thus, as a result, we decided to render the 3D mesh model to the 2D image to decrease the training parameters. Since we were not aiming to build a perfect emotion detector but to examine our proposed hypothesis, 487 images were enough for us to conduct experiments.

We used the 3D mesh editor "meshlab" to render the images. We first loaded all 3D mesh models into the meshlab and processed them one by one. For one 3D mesh model, we adjusted its facing angles and positions to make it face toward us at the center of the screen, which is the real advantage of using the 3D mesh model. If the original image were only the side face of a human, the traditional human emotion detector would not be able to predict its emotion because the majority of the facial features were invisible to the detector. Using the 3D mesh generator can counter this problem by predicting the other side of the face from the visible side based on the knowledge learned from the AFLW-2000 3D data-set. Then adjust the 3D mesh model angles and positions to let it face toward the camera. Then rendered it back to a 2D image, and the traditional classifier could detect its emotion.

3.5 Ablation Study

Ablation study had been widely applied to the neuroscience area to investigate a complex system. In the past, it was used to help people understand the structure and organization in the brain, such as mapping of features inherent to external stimuli onto different areas of the neocortex[Meyes et al., 2019]. In recent years, with the increase of the abilities of the artificial neural network, its structure had become very complicated. To further improve the performance of the artificial neural network, we must first understand how each part and every hyper-parameters affecting it before adding new functions. Thus, we could introduce ablation study to the area of artificial neural network.

In our project, we were aiming to examine whether introducing 3D mesh generating pre-process would increase the overall performance of the human emotion detector. Though our purpose was not creating a perfectly human emotion detector, we could start up with a base model and use ablation study to help us find the best settings for each hyper-parameter. The whole process started with picking a base model and test the model with the two data-sets. Next, I picked one hyper-parameter while kept other hyper-parameters fixed and prepared a select range for this hyper-parameter. Tested the model with the two data-sets and repeated the test for every setting within the range. Then, Compared the results and found the best-performed setting for this hyper-parameter and kept this hyper-parameter in this setting. I

moved to the next hyper-parameter and repeated the above process. The whole process was shown in Figure 3.4.

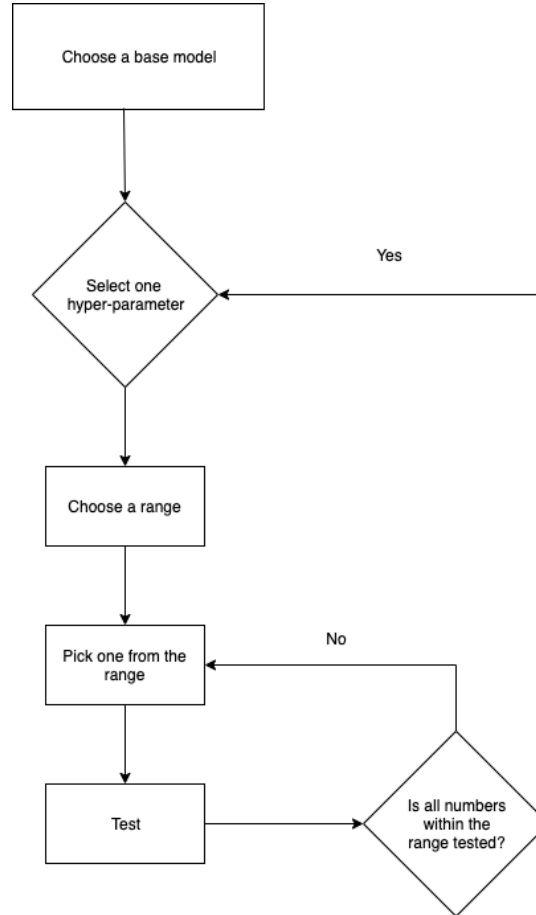


Figure 3.4: The process of the ablation study in our project. We select a base model at first and test its prediction accuracy with the two data-set. Next, we choose a range of hyper-parameters we would like to test and repeat the test for each of them.

During applying the ablation study, we tested the performance of the human emotion detector with the original image data-set and rendered image data-set under every possible circumstance of the current model. Thus, the results and conclusion we got from this experiment would be robust. Also, since we had tested how each hyper-parameter affected the human emotion detector’s performance and found out it is the best setting, the final model would be the best-performed. If other researchers want to make further testings, they could start with our final model and introduce new techniques.

Experimental Methodology

In this Chapter, we would introduce more details about how we conduct our experiment and set up the testing environment.

4.1 Programming Language

Our project had two main sections. The first section was a 3D mesh generator. The 3D mesh generator contained two steps, where the first step was locating the position of the features on the face, and the second step was constructing a 3D mesh from the position data. The first step was done by a pre-trained neural network model, which was implemented in python. The second step was using MatLab to build a 3D mesh model from the position data.

The second section of our project is a human emotion detector. The human emotion detector is a convolutional neural network trained on our prepared data-set, which was implemented using python and PyTorch.

4.2 Image Pre-processing

This section would give more practical details of how we perform image pre-processing in our project.

First of all, we picked the peak emotion image from each sequence in the Cohn-Kanade data-set and organized these peak emotion images into a new folder named "peak_image". These images were in the same order as the sequence order in the Cohn-Kanade data-set. Next, Prepared the environment and dependent libraries for the 3D mesh generator before initializing it. Once the 3D mesh generator got readied, we manually fed one image from the folder "peak_image" into it, and the 3D mesh model generated would be saved into the same folder. The 3D mesh model was stored in the ".obj" file with the textures. Repeat the fed and stored process for the next image in the "peak_image" folder until all images got their ".obj" model. The

above process was showed in Figure 4.1

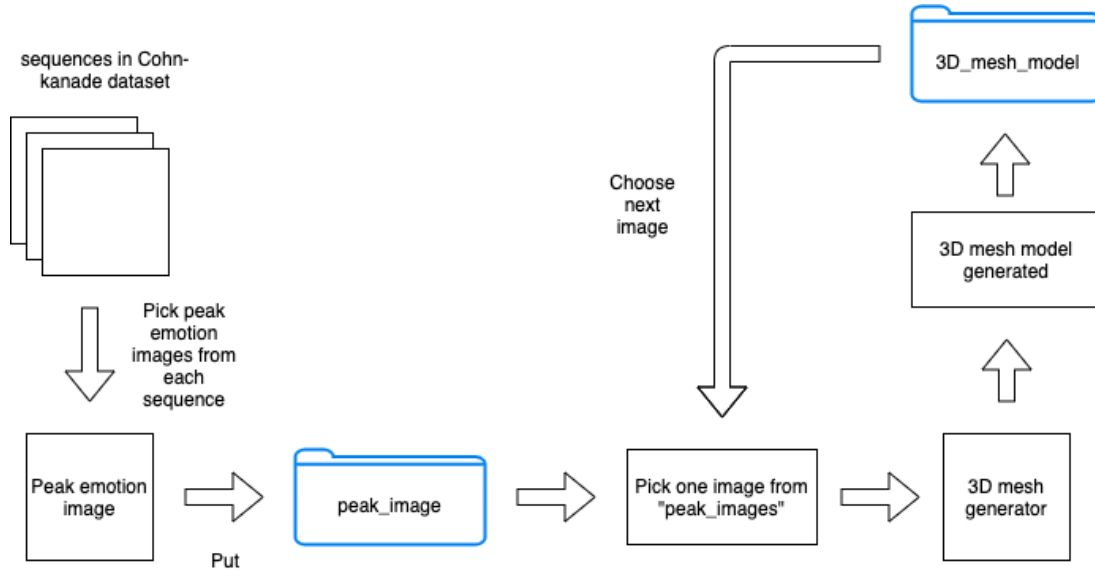


Figure 4.1: The process of the image pre-processing step in our project. We first select the peak image for each sequence in the Cohn-Kanade data-set and organize those peak images into a folder. Pick one image from this folder and generate its 3D mesh model. Repeat this process until all images have their 3D mesh model generated.

Once all images had their 3D mesh generated, we then use meshlab to rendered these 3D mesh models back into 2D images. Opened the meshlab and imported all ".obj" generated in the last step into the meshlab. Next, we adjusted the position of the 3D mesh model and let it face toward us. Then used the render function in meshlab and selected rendered with textures before output a 2D image. Check the 2D image output, whether depicting the emotion on the original image. If yes, went ahead and rendered the next image. If not, went back to the 3D mesh generating steps and adjusted the brightness or sharpness of the image before repeated the above process.

4.3 Convolutional neural network settings

In the previous step, we pre-processed the input image and prepared the data-set for the convolutional neural network. In this section, we would talk about setting of the convolutional neural network in this project.

4.3.1 Image crop and compression

The image in the original data-set had a resolution 490*640, and the image in the rendered data-set had a resolution 926*1585. We were aiming to compare the per-

formance of the classifier using these two data-set so the size of images in the two data-set should be the same. Also, compress the image size could reduce the training parameters as well. For images in both data-set, we first cropped them and left only the face at the center of the images because doing this would decrease the background's effect on the classification result. One example was showed in Figure 4.2.

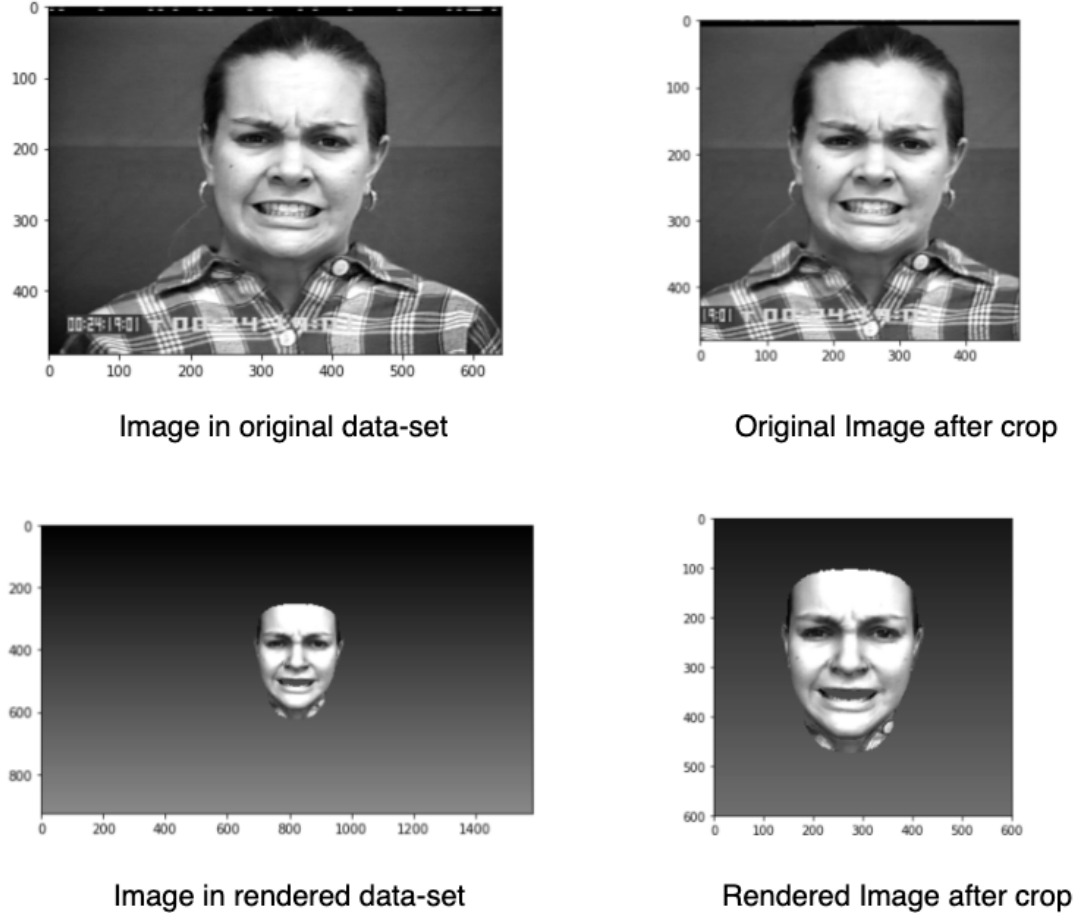


Figure 4.2: Examples of cropped images from original and rendered data-set. Original image was cropped to 480*480 while the rendered image was cropped to 600*600

After the crop, we now had original image in 480*480 and rendered image in 600*600. Then we use "rescale" in python to compress image. The compress coefficient would be tested in the ablation study part to find the best setting.

4.3.2 Data and Label matching

In our project, image data and corresponding labels were prepared separately. When we fed the image data into the classifier, we need to shuffle it to ensure the reliability of the final result. However, when shuffled the image data, labels were not shuffled. Thus the labels cannot match the original image, which made the training impossible.

To solve this problem, we set up a dictionary in python and use the image name "XXX.png" as the keyword. Each keyword would lead us to find its corresponding label. The dictionary was set up before shuffled the image data, so, no matter how random the shuffled results were, we can still found the correct label for each image.

4.3.3 Activation function

In this project, we used ReLU, the rectified linear unit, as our activation function. The ReLU was a non linear activation function and its formula was showed in Equation 4.1.

$$f(x) = \max(0, x) \quad (4.1)$$

The non-linear activation function is vital in neural network training because most of the function we expected the neural network to learn is non-linear. If the non-linear activation function is not allowed, the neural network would become a simple linear classifier and could be simplified by multiplying the weight matrices together. Thus, the neural network cannot handle a complex task, like the image classification in our project. As a result, we would choose the non-linear activation function for our project. Some standard non-linear functions like "tanh", "ReLU" and "sigmoid" were widely used in training the convolutional neural network. ReLU became the winner among these non-linear activation function due to several reasons. The most important reason was the model used ReLU trained much faster than models used other non-linear activation functions and remained a similar accuracy. The other man advantage of ReLU was it not saturated. The gradient of ReLU is always very high when the neuron activated, and as long as the neuron was alive, continuous updates were reasonably efficient. But for tanh and sigmoid activation functions, the gradient became very small when the input is very high or small. This behavior caused the vanishing gradient problem, which was a problem where the lower layer in a neural network trained very slow since the gradient decreased exponentially through the layers. The non-saturated feature of ReLU successfully countered this problem and made the neural network training more efficient. That was the reason we select ReLU as the activation function in our project.

4.4 Ablation study setting

In this section, we would describe the settings of our classifier when applying the ablation study.

Hyper-parameters	Value
Compression size	30*30
Kernel size(convolution layers)	1
Batch size	15
Learning rate	0.001
Max pool size	2
Number of images in train set	390
Number of images in test set	97
Training epochs	50

Table 4.1: Hyper-parameters' settings for base model

4.4.1 Base model

We must select a base model at the first of the ablation study, and the further test would be based on the base model settings. In our project, the base model was defined for the classifier, the convolutional neural network. We chose the settings, as shown in Table 4.1, for hyper-parameters in our classifier.

4.4.2 Hyper-parameters in Ablation study

In the ablation study, it required us to chose a range of hyper-parameters and investigated how each of them could affect our classifier's accuracy. In our project, the hyper-parameters in table 4.2 would be tested in our experiment. Furthermore, we would also test the performance of the model, adding normalization.

Hyper-parameters for testing
Compression size
Kernel size(convolution layers)
Max pool size

Table 4.2: The chosen hyper-parameters for ablation study in our project

Results

5.1 Base model

In the beginning, we chose a base model for the classifier, and the future experiments would be conducted based on this base model. The settings of hyper-parameters for the base model was shown in table 5.1. We tested this model using original and rendered data-set 5 times, and the results were displayed in Table 5.2. From Table 5.2, we can see the base model with the rendered data-set was slightly higher than the base model with the original data-set. At this stage, the performance difference here could not prove our contributions since the features of both data-set were compressed a lot by the base model. The remaining features were not enough for the classifier to learn, and it made a random guess at this situation.

Hyper-parameters	Value
Compression size	30*30
Kernel size(convolution layers)	1
Batch size	15
Learning rate	0.001
Max pool size	2
Number of images in train set	390
Number of images in test set	97
Training epochs	50

Table 5.1: Hyper-parameters' setting for base model

Data-set	1	2	3	4	5	Average
Original data-set	0.2680	0.2371	0.2886	0.2577	0.2783	0.2659
Rendered data-set	0.3093	0.3196	0.2886	0.2783	0.2783	0.2948

Table 5.2: Test result of base model using original and rendered data-sets over 5 times

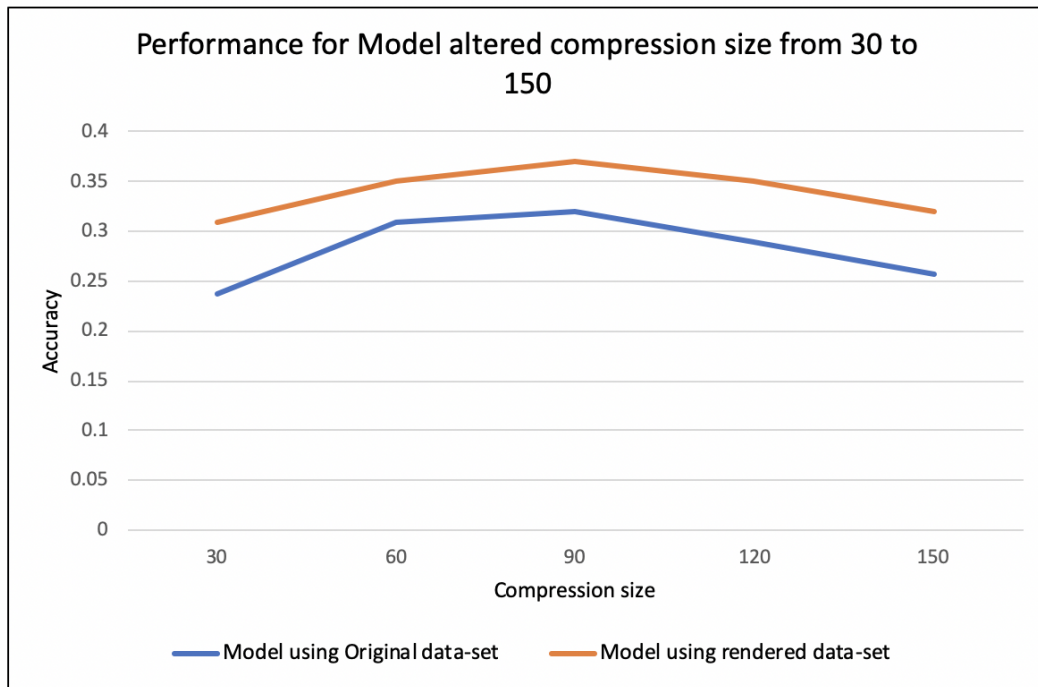


Figure 5.1: Test result of model using original and rendered data-sets for compression size from 30 to 150

5.2 Ablation study

In this section, we would show the experiment results for our ablation study. Each subsection investigated one hyperparameter effect to the classifier performance, and the results were also described.

5.2.1 Compression Size

In this subsection, we altered the compression size and observed the model's accuracy using original and rendered data-set. The model had a hyper-parameters setting, as shown in Table 5.3. The experiment result was showed in Figure 5.1, and from the figure, we observed that the model using rendered data-set was always achieved higher accuracy than the model using the original data-set. Also, both lines peaked when the compression size is around 90 and kept decrease after that. Here, we observed that when the compression size increased, the training parameters would increase as well. However, since we do not have enough data-set to train that amount of parameters, the performance was expected to decrease as a result. Thus, the trend showed in Figure 5.1 proved our analysis.

Hyper-parameters for testing	Value
Compression size	30, 60, 90, 120, 150
Kernel size(convolution layers)	1
Batch size	15
Learning rate	0.001
Max pool size	2
Number of images in train set	390
Number of images in test set	97
Epoch	50

Table 5.3: Hyper-parameters' setting for model under investigation of compression size

5.2.2 Kernel Size

From the last subsection, we observed compression size 90 achieved the best performance for the model used both data-sets. Though the performance of the model with compression size 60 was slightly lower, consider training efficiency, we set compression size to 60 in this section. In this part, we altered the kernel size and observed the model's accuracy using both data-sets. The setting of hyper-parameter was shown in Table 5.4. The experiment result was showed in Figure 5.2. From the figure, we noticed that the performance of the model using the rendered data-set was higher than the model using the original data-set. As the kernel size increase, the relations between pixels in the image would become more apparent. Thus, the classifier could learn more knowledge from the output of the image applied kernel and increase the prediction performance. The experiment results proved this trend, which proved the correctness of our model.

Hyper-parameters for testing	Value
Compression size	60
Kernel size(convolution layers)	1, 3, 5, 7, 9
Batch size	15
Learning rate	0.001
Max pool size	2
Number of images in train set	390
Number of images in test set	97
Epoch	50

Table 5.4: Hyper-parameters' setting for model under investigation of kernel size

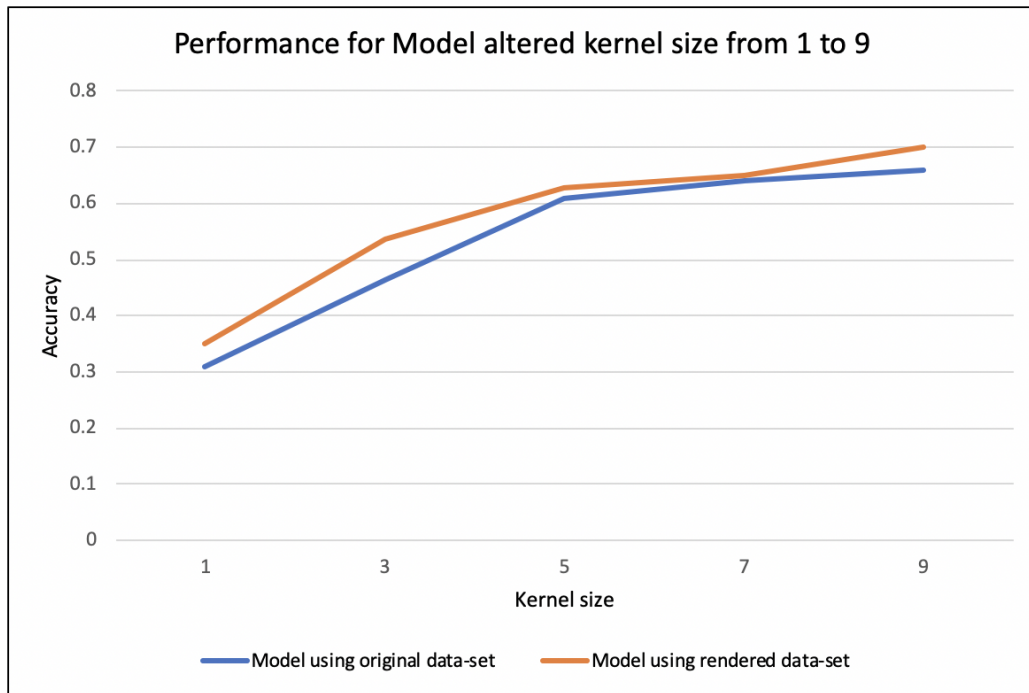


Figure 5.2: Test result of model using original and rendered data-sets for kernel size from 1 to 9

5.2.3 Max pool size

From the last sections, we chose the kernel size 9 based on the performance. In this section, we would test the max pool size from 1 to 5, and the setting of the model, as shown in Table 5.5. Max pooling was used to down-sample an input image. As the increase of the max pool size, more information from the original input would be compressed. The experiment result was showed in Figure 5.3. From the figure, we observed that the performance of the model using rendered data-set was lower than the model using the original data-set after max pool size 3. As mentioned above, a large max pool size meant more information compressed. Since our input image was compressed to 60×60 already, a large max pool size would shrink the input into a tiny matrix, which resulted in most critical features lost. The advantage of the rendered image was it got rid of information that unrelated to emotion detection, which meant information remained was important in detecting emotions. In that case, compress rendered image would lead to more key information lost compared to compress the original image. Thus, the model using the original data-set would achieve a better performance compared to the model using rendered data-set. This situation should be considered when concluded.

Hyper-parameters for testing	Value
Compression size	60
Kernel size(convolution layers)	9
Batch size	15
Learning rate	0.001
Max pool size	1, 2, 3, 4, 5
Number of images in train set	390
Number of images in test set	97
Epoch	50

Table 5.5: Hyper-parameters' setting for model under investigation of max pool size

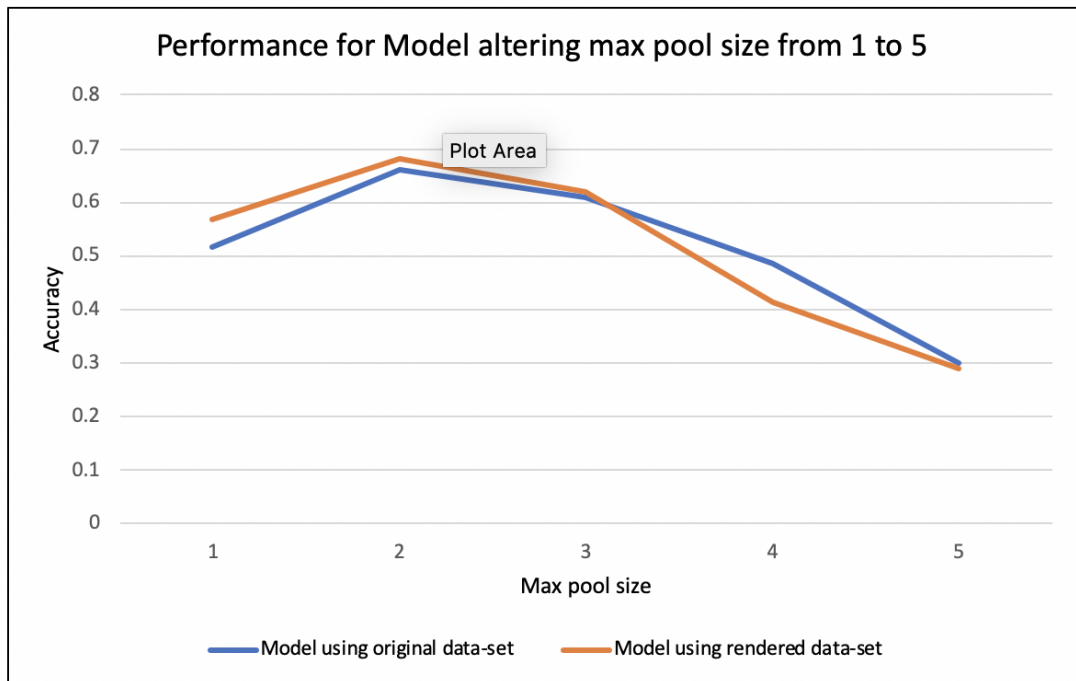


Figure 5.3: Test result of model using original and rendered data-sets for max pool size from 1 to 5

5.2.4 Adding normalization

We chose max pool size 2 when the input image was compressed to 60*60. In this section, we would normalize the output from each convolutional layer before fed into the next layer. As the base model, we test this new model with two data-sets 5 times. The average prediction accuracy for the model using the original data-set was 67.01% and the 69.07% for the model using rendered data-set. It proved once again using rendered images would increase the overall performance of the classifier. We were not satisfied with this result because we compressed the input image into 60*60,

which was the most efficient but not the best. Then we repeated our experiment and compressed the image into $90 * 90$ with a max pool size 3. Max pool size 3 was trying to down-sample the input into a similar size as the previous model. The final prediction accuracy of the model using the original input was 64.95%, but the prediction accuracy of the model using rendered input rose to 74.22%. Since we decrease the compressing rate, the original image would have more noisy features remained but for the rendered image, more useful features would be kept. Thus the classifier would learn more knowledge by using rendered data-set, and the experiment result also confirmed our methods.

5.3 Summary

So far, we have tested how the hyper-parameters such as compression size, kernel size, max pool size, and normalization affected the performance of the classifier. The best classifier prediction accuracy we achieved was 74.22% when using rendered data-set. The performance of our model was lower than other models used the same data-set, but this was due to only peak image from each sequence was considered. Limited data-set indeed affected our classifier prediction accuracy, but we proved in most situations, using images from 3D mesh pre-processing would increase the performance of the classifier. But when the input image was compressed or down-sample too much, using our pre-processing would not help since more key features would lose during the process than the original image.

Conclusion

Summarize your work, state your main findings and discuss what you are going to do in the future in Section 6.2.

6.1 Conclusion

In our project, we have performed the ablation study to investigate how each hyper-parameters could affect the performance of the classifier and found out the best settings for each hyper-parameters. During the experiment, we fed the classifier with our two data-set, the original images, and rendered images from 3D mesh pre-processing. From the experiment, we observed that the model using rendered data-set would always achieve a better performance than the model using original data-set except when the input image was either compressed or down-sampled too much. When compressed or down-sampled, the features of the input image would be lost as a result. Since the image rendered from our 3D mesh pre-processing had already eliminated most of the information that irrelevant to emotion detection such as the background, poser's hair, and teeth, further compress or down-sample would cause the critical information lost. The features in the original image were scattered all over the image, so compress or down-sample would not cause a severe performance drop.

The best performance of our model so far was 74.22%, which was good for a 6 labels classification classifier using only 487 samples. The performance proved that our model was learning features from the training data-set, so our convolutional neural network model was correct. Though the performance was good, there were still spaces we can further improve.

Our project aim was not to build a perfect classifier but to examine our hypothesis, whether introducing 3D mesh pre-processing step would increase the overall performance of the human emotion detector. Observed from the current experiment, we found that 3D mesh generate pre-processing would increase the average performance of the human emotion detector used artificial neural network under most situations. Thus our hypothesis was proved by this experiment.

6.2 Future Work

As mentioned above, there were still some hyper-parameters were not tested, so in the future, researchers could continue this experiment and altered the hyper-parameters, which had not been tested in the report. By doing that, we would have more results to examine our hypothesis and the classifier's performance could be further improved.

Besides, the classifier in our experiment was implement using the artificial neural network, and in the future, researchers could extend the experiment to other non deep learning emotion detector and test whether 3D mesh pre-processing would increase the overall performance of those human emotion detectors. By doing that, we could enlarge the coverage area of our proposed method and make our 3D mesh pre-processing a common step in the human emotion detection task.

In this project, we rendered the 3D mesh model into a two image before input to our classifier and this was meant to reduce training parameters due to insufficient training samples. In the future, researchers could choose other dataset with more training samples. Thus, without rendering to 2D image, researchers could test our classifier with 3D mesh as input and compare the results. Although the training time of the classifier using the 3D mesh as the input will be significantly longer than the training time of the classifier using the 2D image as the inputwe expected the classifier with 3D mesh as input would have a better prediction accuracy.

The convolutional neural network in this project used the most common structure in the current world, so the experiment results proved the effectiveness of the 3D mesh pre-processing for majority human emotion detector. But researchers could implement our pre-processing technique to other pre-trained model which used a different structure and test whether our hypothesis still be true under those environments.

Bibliography

<http://www.pitt.edu/~emotion/ck-spread.htm>. (cited on page 12)

Psychology. <https://psychology.iresearchnet.com/social-psychology/emotions/facial-expressions-of-emotion/>. (cited on page 8)

ALTMAN, N. S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46, 3 (1992), 175–185. doi:10.1080/00031305.1992.10475879. <https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879>. (cited on page 6)

BZDOK, D.; KRZYWINSKI, M.; AND ALTMAN, N., 2017. Machine learning: a primer. *Nature Methods*, 14 (Nov 2017), 1119 EP –. <https://doi.org/10.1038/nmeth.4526>. (cited on page 6)

CASTELLANO, G.; VILLALBA, S. D.; AND CAMURRI, A., 2007. Recognising human emotions from body movement and gesture dynamics. In *Affective Computing and Intelligent Interaction*, 71–82. Springer Berlin Heidelberg, Berlin, Heidelberg. (cited on page 8)

COLLOBERT, R. AND WESTON, J., 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML '08* (Helsinki, Finland, 2008), 160–167. ACM, New York, NY, USA. doi:10.1145/1390156.1390177. <http://doi.acm.org/10.1145/1390156.1390177>. (cited on page 7)

CORTES, C. AND VAPNIK, V., 1995. Support-vector networks. *Machine Learning*, 20, 3 (Sep 1995), 273–297. doi:10.1007/BF00994018. <https://doi.org/10.1007/BF00994018>. (cited on page 6)

KE, Q.; LIU, J.; BENNAMOUN, M.; AN, S.; SOHEL, F.; AND BOUSSAID, F., 2018. *Computer Vision for Human-Machine Interaction*, 127–145. Academic Press, United Kingdom. ISBN 9780128134450. (cited on page 7)

LUCEY, P.; COHN, J.; KANADE, T.; SARAGIH, J.; AMBADAR, Z.; AND MATTHEWS, I., 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. 94 – 101. doi:10.1109/CVPRW.2010.5543262. (cited on page 12)

MAITRA, D. S.; BHATTACHARYA, U.; AND PARUI, S. K., 2015. Cnn based common approach to handwritten character recognition of multiple scripts. In *2015 13th*

-
- International Conference on Document Analysis and Recognition (ICDAR)*, 1021–1025. doi:10.1109/ICDAR.2015.7333916. (cited on page 2)
- MCCULLOCH, W. S. AND PITTS, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5, 4 (Dec 1943), 115–133. doi:10.1007/BF02478259. <https://doi.org/10.1007/BF02478259>. (cited on pages v and 5)
- MEYES, R.; LU, M.; DE PUISEAU, C. W.; AND MEISEN, T., 2019. Ablation studies in artificial neural networks. *CoRR*, abs/1901.08644 (2019). <http://arxiv.org/abs/1901.08644>. (cited on page 16)
- PARTHASARATHY, S., 2019. Difference between traditional programming versus machine learning from a pm perspective. <https://productcoalition.com/difference-between-traditional-programming-versus-machine-learning-from-a-pm-perspective-3802b02bc7f6>. (cited on page 6)
- RUSSELL, S. J.; NORVIG, P.; AND DAVIS, E., 2010. *Artificial intelligence: a modern approach*. Prentice Hall. (cited on pages 2 and 6)
- SAUTER, D. A. AND EIMER, M., 2010. Rapid detection of emotion from human vocalizations. *Journal of Cognitive Neuroscience*, 22, 3 (2010), 474–481. doi:10.1162/jocn.2009.21215. <https://doi.org/10.1162/jocn.2009.21215>. PMID: 19302002. (cited on page 8)
- SCHMIDHUBER, J., 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61 (2015), 85 – 117. doi:<https://doi.org/10.1016/j.neunet.2014.09.003>. <http://www.sciencedirect.com/science/article/pii/S0893608014002135>. (cited on pages 1 and 5)
- WANG, F.; ZHANG, J.; WEI, Q.; ZHENG, X.; AND LI, L., 2017. Pdp: parallel dynamic programming. *IEEE/CAA Journal of Automatica Sinica*, 4, 1 (Jan 2017), 1–5. doi:10.1109/JAS.2017.7510310. (cited on page 7)
- WERBOS, P., 1975. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University. <https://books.google.com.au/books?id=z81XmgEACAAJ>. (cited on page 5)
- WUJIE1010, 2019. Wujie1010/facial-expression-recognition.pytorch. <https://github.com/Wujie1010/Facial-Expression-Recognition.Pytorch>. (cited on page 8)
- XING, F. AND YANG, L., 2016. *Machine learning and its application in microscopic image analysis*, 97–127. ISBN 9780128040768. doi:10.1016/B978-0-12-804076-8.00004-9. (cited on page 7)
- ZELL, A., 1997. *Simulation neuronaler Netze*. Addison-Wesley. (cited on page 5)
- ZHU, X.; LIU, X.; LEI, Z.; AND LI, S. Z., 2019. Face alignment in full pose range: A 3d total solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41, 1 (Jan 2019), 78–92. doi:10.1109/tpami.2017.2778152. <http://dx.doi.org/10.1109/TPAMI.2017.2778152>. (cited on pages 9 and 39)

Appendix I

.1 Proeject Title

Detecting Human Emotion From 3D Face Mesh

.2 Project Description

Identifying human emotions from images or videos is a challenging problem. The current emotion detection algorithm solely depends on 2D images, which perform poorly if the whole face is not visible in the image or the person is not facing the camera directly. Generating a 3D Face Mesh that learns to fit a face model to an image can be helpful in challenging situations. In this project, we aim to create a 3D Face model that can be used to detect human emotion from images or videos.

Learning objects includes:

- Experience with face and emotion recognition
- Experience with computer vision for emotion recognition problems
- Experience with deep learning for face related computer vision tasks

.3 Project Outcome

The final model should be able to:

Goal 1(G1): Generate a 3D Mesh of a human face from image and/or videos with different pose and camera angles.

Goal 2(G1): Use the generated 3D mesh to detect emotion.

Appendix II

Attached was my independent study contract and due to size limitation, the independent study contract would display from next page.



INDEPENDENT STUDY CONTRACT

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

SECTION A (Students and Supervisors)

UniID:	u5894100		
FAMILY NAME:	Yang	PERSONAL NAME(S):	Jialin
PROJECT SUPERVISOR (may be external):	Zakir Hossain and Moshir Farazi		
COURSE SUPERVISOR (a RSCS academic):	Tom Gedeon		
COURSE CODE, TITLE AND UNIT:	COMP4560 Advanced Computing Project		

SEMESTER ☒ S1 YEAR: 2019_ ☒ S2 YEAR: 2019_

PROJECT TITLE:

Detecting human emotion from 3D face mesh

LEARNING OBJECTIVES:

- Experience with face and emotion recognition
- Experience with computer vision for emotion recognition problems
- Experience with deep learning for face related computer vision tasks

PROJECT DESCRIPTION:

Identifying human emotion from images or videos is a challenging problem. Current emotion detection algorithm solely depend on 2D images which performs poorly if the whole face is not visible in the image or the person is not facing the camera directly. Generating a 3D Face Mesh that learns to fit a face model to an image can be helpful in the challenging situation. In this project, we aim to create a 3D Face model that can be used to detect human emotion from images and/or videos.

The final model should be able to:

Goal 1(G1): Generate a 3D Mesh of a human face from image and/or videos with different pose and camera angles.

Goal 2(G1): Use the generated 3D mesh to detect emotion.

For achieving G1:

Research School of Computer Science

12

Form updated Jun-

ASSESSMENT (as per course's project rules web page, with the differences noted below):

Assessed project components:	% of mark	Due date	Evaluated by:
Thesis	45%		
Artefact	45%		
Presentation	10%		

MEETING DATES (IF KNOWN):

Weekly

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

Signature: [Signature]

Date: 18 April 2019

SECTION B (Supervisor):

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

Signature: MRF [Signature] T.D. Gedeon

Date: 18 April 2019

Reviewer

Name:

Signature:

REQUIRED DEPARTMENT RESOURCES:**SECTION C (Course coordinator approval)**

Signature:

Date:

SECTION D (Projects coordinator approval)

Signature:

Date:

Appendix III

.4 Code files description

The artefact submitted had two folders inside.

- "3DDFA"
 - All the codes contained in this folder was taken from the external resources and the original author was Dr. Guo Jianzhu[Zhu et al., 2019]. This folder was our 3D mesh generator and more usage details could be found at Appendix IV.
- "classifier"
 - "CNN_4560.ipynb". This was the classifier model and all modules was implemented by myself. This was the human emotion detector in our project and more usage details could be found at Appendix IV.

.5 Correctness test

For the 3D mesh generator, followed the usage guide in Appendix IV and manually compared the generated 3D mesh model with the original image. By looking at the 3D mesh model and its original image, we could identify the poser on them as the same person with the same emotion. Thus, we can prove the correctness of the 3D mesh generator.

As for the classifier, we used the convolutional neural network. For the artificial neural network, the correctness cannot be determined by whether the code compiles correctly. Instead, run our classifier with the data-set and corresponding labels and checked the prediction accuracy of the test set at last. Accuracy depicted whether our model learned anything from the data-set or not. Also, observe the loss change during the training and loss decrease represents the model has improved itself through training. Thus, accuracy and loss decrease could be used to prove the correctness of

the artificial neural network.

.6 Description of how experiments were conducted

There were three steps in our experiment as listed below:

- 3D mesh generating
 - Test environment: Linux only
 - Test hardwares:
 - * Processor: Intel(R) Core(TM) i7-7700HQ @ 2.80Hz
 - * Memory: 16 GB 2133 MHz LPDDR3
 - * Graphic Card: NVIDIA GeForce GTX 1050 Ti
 - Data-set: Modified Cohn-kanade data-set with only peak emotion image in each sequence. The data-set can be found in "ck" folder.
 - Follow the usage in Appendix IV and ran the 3D mesh generator on every images in "ck" folder, which should have 487 in total. Organize all the ".obj" model generated into one folder and input this folder to next "rendering" step.
- Rendering
 - Test environment: Linux, MacOS, Win10
 - Test hardwares were same as in "3D mesh generating" step
 - Followed the usage in Appendix IV and rendered every ".obj" model from last step into 2D regular images. Next Input the 2D regular image into the "classification" step.
- Classification
 - Test environment: MacOS, Linux
 - Test hardwares:
 - * Processor: 2.2 GHz 6-Core Intel Core i7
 - * Memory: 16 GB 2400 MHz DDR4
 - * Graphic Card: Radeon Pro 555X 4 GB
 - We have two data-sets for this step. The data-set in "obj487_original" was the original images and the data-set in "obj487" was the rendered images from our 3D mesh generator. Also, the labels data was named "4560_ck_labels.csv" and all of them were already prepared under "classifier" data-set. Modify the parameters as described in reports and ran the classifier. Repeat this step until all parameters tested.

Appendix IV

Due to size limitation, the figures were showed in the next page.

The code contains three parts, 3D mesh generator, rendering and Convolutional neural network model

3D mesh generator. External work.

Original repo: <https://github.com/cleardusk/3DDFA>

Requirements:

PyTorch >= 0.4.1 (PyTorch v1.1.0 is tested successfully on macOS and Linux.)

Python >= 3.6 (Numpy, Scipy, Matplotlib)

Dlib (Dlib is optionally for face and landmarks detection. There is no need to

use Dlib if you can provide face bounding bbox and landmarks.

Besides,

you can try the two-step inference strategy without

initialized

landmarks.)

OpenCV (Python version, for image IO operations.)

Cython (For accelerating depth and PNCC render.)

Platform: Test only on linux

Required dataset: 3DDFA/ck

Same as Cohn kanade dataset but pick only peak emotion image for each sequence.

Installation instructions:

sudo pip3 install torch torchvision # for cpu version. more option to see

<https://pytorch.org>

sudo pip3 install numpy scipy matplotlib

sudo pip3 install dlib==19.5.0 # 19.15+ version may cause conflict with pytorch in linux, this may take several minutes

sudo pip3 install opencv-python

sudo pip3 install cython

Usage:

1. Build cython module (just one line for building)

cd utils/cython

python3 setup.py build_ext -i

2. Run the main.py with arbitrary image as input

python3 main.py -f ck/SXXX/SXXX_XXX.png (replace XXX with the sequence you want to test)

The generated 3D mesh model will have the same name as the image but in .obj and .ply file under the same folder.

All 3D mesh model generated had been already put into Folder OBJ and you can download at

https://drive.google.com/file/d/1KCr3rJ258EdtG07NxIi_b_YfI1o96BDQ/view?usp=sharing.

Rendering

Required software: meshlab
Can be download at <http://www.meshlab.net/>
Platform: Linux, win10, macOS

Required dataset:

".obj" models can be obtained from running "3D mesh generator" on all input images

Or you can directly download all 3D mesh model at
https://drive.google.com/file/d/1KCr3rJ258EdtG07NxIi_b_YfI1o96BDQ/view?usp=sharing.

Usage:

1. Open meshlab
2. File -> import mesh -> select one .obj file from OBJ folder -> open
3. Rotate 3D mesh and let it face front
4. Go to bottom right panel and select "Vert" for Shading, "Vert" for Coloraturas, "Double" for Back-face
5. File -> Save snapshot -> Save

In our project, we had rendered all .obj inside the OBJ folder and the rendered image was put into the folder /classifier/obj487. You can direct use this folder as the input for next part.

Convolutional neural network model

Requirement:

Jupyter notebook. Can be download at <https://jupyter.org/>
PyTorch >= 0.4.1 (PyTorch v1.1.0 is tested successfully on macOS and Linux.)
Python >= 3.6 (Numpy, pandas, scipy, skimage)

Required dataset:

Original image input: classifier/obj487_original folder
Rendered image input: classifier/obj487 folder
Labels: 4560_ck_labels.csv

Usage:

1. Go into folder "classifier" and open the "CNN_4560.ipynb" with Jupyter notebook
2. Run through the blocks. The model would print loss for each epoch and saved model at last. Then test the saved model with test data-set and it will print out the prediction accuracy.
3. Details about how to modify parameters or change input dataset are showed as comments inside "CNN_4560.ipynb"