

Copyright

By

Hui Wang

2018

ECAMERA: A REAL-TIME FACIAL EXPRESSION RECOGNITION SYSTEM

by

Hui Wang, BE

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

DECEMBER, 2018

ECAMERA: A REAL-TIME FACIAL EXPRESSION RECOGNITION SYSTEM

by

Hui Wang

APPROVED BY

Jiang Lu, PhD Chair

Thomas L Harman, PhD, Committee Member

Xiaokun Yang, PhD, Committee Member

APPROVED/RECEIVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

Said Bettayeb, PhD, Associate Dean

Ju Kim, PhD, Dean

Dedication

I would love to dedicate my thesis to my parents who have always loved me unconditionally, always been a constant source of support and encouragement during the challenges of my whole graduate life. Also to my little sister whom I am truly grateful for having in my life.

Acknowledgments

I would first like to thank my thesis advisor Dr. Jiang Lu of the Computer Engineering department at University of Houston – Clear lake. The door to Prof. Lu's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. I sincerely thank him for his guidance and support throughout this study and specially for his confidence in me.

I would like to thank the members of my thesis committee, Dr. Thomas Harman, and Dr. Xiaokun Yang, for their time and their helpful advice. I would also like to thank the professors at the College of Science and Engineering at UHCL whose classes I took during my master's program for ultimately contributing to my academic growth.

To all my friends, thank you for your understanding and encouragement in many, many moments of crisis. Your friendship makes my life a wonderful experience. I cannot list all the names here, but you are always on my mind.

Finally, I must express my very profound gratitude to my parents for their love and support throughout my life. Thank you both for giving me strength to reach for the stars and chase my dreams. My sister deserves my wholehearted thanks as well.

This thesis is only a beginning of my journey!

ABSTRACT

ECAMERA: A REAL-TIME FACIAL EXPRESSION RECOGNITION SYSTEM

Hui Wang
University of Houston-Clear Lake, 2018

Thesis Chair: Jiang Lu, PhD

According to a report from the United Nations, “There is approximately 20 percent of youth experiencing a mental health condition each year on a global level”. Young generation is at great risk for a variety of mental-health conditions. Mental health problems affect about 1 in 10 children and young people, including depression, anxiety and conduct disorder, and are often a direct response to what is happening in their lives. Facial expression is one of the direct reflections during daily life to show their emotions which is a key factor to mental status. Therefore, the design of a real-time facial expression recognition system to look after our young generation is an urgent and vital thing to do.

This project is devoted to build an e-camera which is a facial expression recognition system based on Raspberry Pi from a live Pi Camera feed and get results in real time processing. Although computer vision and facial expression recognition technology have made significant progress in recent years with many professional systems available for real-world applications, it still gains strong interest to implement such a system on a smaller device at a reasonable price such as a single-board computer. The proposed system combines image pre-processing and Convolutional Neural Network

(CNN) to build the facial expression recognition model. In pre-processing, the Haar-Cascade is implemented for face detection. Moreover, 68 facial landmarks are collected for expression feature extraction. Then, CNN is used for training and testing of face expressions classification. The trained CNN model is saved in Raspberry Pi for real-time facial expression recognition. All the computing algorithms are performed on the eCamera. Only the face expression recognition results are delivered to users. The two public datasets, JAFFE and CK+, are used in simulations to evaluate our proposed expression recognition procedures. The initial real-time experimental testing is also provided in results. Compared with other previous models that build upon OpenCV, our proposed model shows a great improvement in accuracy with robust computing.

Overall, compared with above methods, our work: presents a higher accuracy in the JAFFE[1] and CK+ databases [55]; a more robust evaluation methodology; recognition of seven expression categories instead of only five or six as done by Song et al.[2].

TABLE OF CONTENTS

List of Tables	x
List of Figures	xi
Chapter	Page
CHAPTER I: INTRODUCTION.....	1
1.1 Background.....	1
1.2 Applications	2
1.3 Significance.....	4
1.4 Challenges.....	5
1.5 Contributions.....	6
1.6 Organization.....	7
CHAPTER II: RELATED WORK	8
2.1 Methodology	8
2.2 Brief History of Neural Networks.....	10
2.3 Wireless Applications	12
CHAPTER III: SYSTEM SETUP	13
3.1 Hardware.....	13
3.1.1 Raspberry Pi.....	13
3.1.2 Camera Module.....	14
3.1.3 Hardware Setup.....	15
3.2 Software	16
3.2.1 Python and OpenCV	16
3.2.2 Keras	16
3.2.3 TensorFlow	17
3.3 System Diagram.....	18
CHAPTER IV: METHODS.....	21
4.1 Face Detection	22
4.2 Feature Extraction.....	23
4.3 Facial Expression Recognition	25
4.3.1 Convolutional Neural Network.....	25
4.3.2 CNN Architecture	28
4.3.3 CNN Training Process	31
4.3.4 Confusion Matrix.....	32
CHAPTER V: VISUALIZATION	34

5.1 Methods of Visualizing a CNN model.....	34
5.1.1 Preliminary Methods.....	34
5.1.2 Activation Maps.....	37
5.2 Layer Activation	40
CHAPTER VI: RESULTS.....	48
6.1 Simulation	48
6.2 Implementation	53
CHAPTER VII: CONCLUSION.....	55
REFERENCES	56

LIST OF TABLES

Table 1.1: Machine learning and computer vision real world applications	3
Table 2.1: Important contributions in the history of neural networks	11
Table 4.1: Confusion Matrix	33
Table 5.1: Plotting model architecture	35
Table 6.1: Confusion matrix for seven expressions on the JAFFE dataset	50
Table 6.2: Confusion matrix for seven expressions on the CK+ dataset	51
Table 6.3: Confusion matrix for seven expressions on the combined dataset of CK+ and JAFFE	52
Table 6.4: Confusion matrix for five expressions on the CK+ dataset	53
Table 6.5: Experimental Results	54

LIST OF FIGURES

Figure 3.1: Raspberry Pi 3 Model B+	13
Figure 3.2: Pi camera connected to Raspberry Pi	14
Figure 3.3: Facial expression recognition system overview.	19
Figure 4.1: Flow Chart.....	21
Figure 4.2: Cropped Face from Face Detection Stage	23
Figure 4.3: Visualization of the 68 facial landmarks	23
Figure 4.4: Feature Extracted based on 68 Facial Landmark.	24
Figure 4.5: Architecture of the Convolution Neural Network.....	30
Figure 4.6: Loss and Accuracy Over Time	32
Figure 5.1: Output of 30 th filter of the first convolutional layer	37
Figure 5.2: Learned features from all eight convolutional layers	38
Figure 5.3: Visualization of features in a fully trained model	41
Figure 6.1: Samples from dataset.....	48

CHAPTER I: INTRODUCTION

1.1 Background

The association of scientific fields in computer vision, machine learning, and the artificial intelligence technologies is more and more important in real life by its numerous human-computer interaction cutting edge applications, such as Face ID on iPhone X and a humanoid robot. The human-computer interaction is believed to produce a high impact effect when the machine could positively communicate and interact with a human. Computer vision is actually the study of visual data. With a large number of sensors around the world, the amount of visual data has been growing exponentially. So it's critical for us to develop algorithms that can utilize and understand the data.

Vision is one of the most important and prioritized senses we human beings have. We can constantly look at the world around us and quickly identify what we see. We can recognize what the expression is on a person's face instantly. However, machines are not capable of this high-level generalization, so we need to train them to mimic the human brain and obtain the ability to understand human performance. With the huge and rapid development of human-computer interaction applications[3], machines are required to have the ability to read human facial expressions in order to identify and sense human emotion.

With the power of high performance computing technology, the combination of deep learning and computer vision has been pushed to a new stage. Machine learning allows perceptrons to learn and represent data with neural-like architecture to understand information. Deep learning is one of machine learning techniques. One of the most significant deep learning schemes used in conjunction with computer vision research is Convolutional Neural Networks(CNN). CNN models have shown tremendous capacity and ability in making a good progress in the field of computer vision along with several

other fields like natural language processing and speech recognition. In this project, we use CNN to classify facial expressions.

The face is the most important object to humans. Facial expressions are strongly linked to the person's inner emotions. Researchers have used human observation of recorded videos to try to assess emotional response. Although human assessment has many limitations, for example, the same expression may seem differently by different person, facial expression recognition technology offers an opportunity to overcome some of these limitations, delivers a much greater level of insight about personal sentiment and reactions.

Facial Action Coding System (FACS)[4] is the most commonly and widely used guide for human facial expression recognition since its publication 40 years ago. Based on an anatomical analysis of facial action through the codification of facial expression in 44 “action units” (AUs), through observational and electromyography study of facial behavior, they determined how the contraction of each facial muscle, both singly and in unison with other muscles, change the appearance of the face. The paper talks about six basic emotions: anger, disgust, fear, happiness, sadness and surprise. This project proposes to deal with these six emotions, and plus the neutral face, total seven categories of expression recognition.

1.2 Applications

Recently, deep learning methods have been shown to outperform previous state-of-the-art techniques in several tasks, as well as the abundance of complex data from different sources (e.g., visual, audio, medical, social, and sensor).[5] There are plenty of applications in various computer vision tasks, such as object detection, face recognition, human pose, and gesture recognition.

For machines, there are plenty of ways to understand human emotions, like text, voice, and of course, facial expression. Facial expression is the most perceptual intuition and easiest way to observe and record compared with other sensory methods.

Applications of facial expression recognition are spread across many different fields like the medical field, E-learning, monitoring, marketing, and entertainment. Table 1.1 shows some of the existing applications related to machine learning and computer vision.

Table 1.1: Machine learning and computer vision real world applications

Fields	Applications
Medical field	Rehabilitation and Companion
	Psychological health care[6]
E-learning	Adjust the presentation style of an online tutor[7]
	Detect the state of the learner
Monitoring	Car driver fatigue monitoring[8]
Entertainment	Music player that can recognize mood and emotions of the user[9]
Marketing	Impact of ads[10]
	Attention and engagement
	Emotion vital in purchasing decisions

Here are some detailed introduction, explaining how the technologies are used on specific products:

1. Psychology and Computer Vision

We all know that facial expression plays an important role in cognition of human emotions. The original idea of this paper is to take advantage of computer vision

to create a product that can help children who may potentially have psychological diseases. The eCamera we built can be used to detect his/her face, recognize and record his/her expression. The recognition result can be saved and sent to a psychologist for further analysis based on long term monitoring.

2. Driver Fatigue Monitoring

There are some existing products for driver drowsiness detection. Different techniques were used in [11] to complete the driver drowsiness detection, Eye Blinking Based Technique, Yawning Based Technique, and Head Nodding Detection. Among these techniques, eye blinking and yawning are related to facial expression recognition.

3. Marketing

The technology of facial expression recognition has already been used to understand how consumers are satisfied and engaged with the brand content and advertising, and how these emotions then influence brand awareness and purchase intent. The key insight is that at the emotional level, we subconsciously form opinions that have a huge influence on our purchase decisions. This will help to transform marketing and advertising by reading human expressions and then adapting consumer experiences in real time.

1.3 Significance

Facial expressions are indeed powerful, biologically based reactions to the current situation and environment surrounding. Abnormal facial expressions often occur in different situations. Mental health is a major key component in a child's healthy development. Children need to be healthy in order to learn, grow, and lead productive lives. Although security cameras installed on school buses, hallways, classrooms,

bathroom entrances, etc. have been widely exploited to address issues such as safety and surveillance, so parents can know their children's situation wherever and whenever with their mobile devices, they have been little employed as part of the mental care. For instance, some daycares install baby monitors to let parents know their babies are safe and sound. That is not enough. Parents and faculties should pay more attention to students' mental or psychological health. Children's emotional health is as important as their physical health and safety. Healthy mental status allows children and young people to develop the resilience to cope with difficulties and uncertainties of their lives and grow into well-rounded, healthy adults.

One of the simplest, yet maybe most effective, ways to learn about child mental condition is observation by. Observing their actions, expressions, and temperament when they eat, sleep, and play. Although there are cameras everywhere on campus, it is still a heavy workload to watch all the videos to locate your kid among all the students, not even to see his/her expression at a specific time. With the help of our proposed facial expression recognition system, this can be easily fulfilled. For example, one does not have to go through the whole recorded video, just seeing the analysis results you can find that your kid has been sad for a continues period of time, there may be little or no joy or interest in normal age-appropriate activities, then he or she may in an unhealthy mental condition.

1.4 Challenges

Automatic recognition of facial expressions through only a facial picture is a complex issue even to human, not just because of significant variations in the faces with respect to person's identity, environment illumination and head pose, also perception is subjective. What we think we see is actually filtered through our own mind's eye.

Experiment in[12] shows that even culture can exert a great influence on the production and perception of basic emotion signal. There are nearly no pure expressions of one emotion, always predominant expression with admixtures of different emotions. Some studies present that a human can make more than 20 distinct facial expressions[13]. We confront the fact that natural images result from the interaction of multiple factors related to scene structure, illumination, and imaging. For facial images, these factors include different facial geometries, expressions, head poses, and lighting conditions.

Facial expression recognition typically has a series of related problems:

- Find the faces in a video frame;
- Crop out face/facial parts area;
- Pick up unique features of the face that can be used to distinguish from other expressions;
- Compare the unique features of the facial expressions to determine what the emotion is.

This gives rise to our problem statement to be able to infer emotions in real time using a live video feed with the following challenges:

- To be able to provide solid results in real time applications;
- To be an intelligent, independent and integrated system that can do both image acquisition and facial expression recognition;
- To be able to process the face expression recognition procedures on edge devices.

1.5 Contributions

In this project, we proposed a Raspberry Pi based facial expression recognition system to efficiently process images and videos with different illumination captured by Pi

Camera in real-time. As the size of Raspberry Pi 3 is very small, light weight and low power consumption, it can be easily used for many applications such as monitoring the elderly or children at home, monitoring critical patients in ICU, for therapists and patients to keep track of the process and effects of their mental therapy treatment plans and so forth. Additionally, it adds wireless LAN & Bluetooth connectivity making it the ideal solution for our design. In real-world applications, the system can use motion detection algorithm, which enables live streaming camera along with detection of motion to trigger Pi Camera intermittently to take face images waiting to be recognized. This will significantly decrease storage usage. The live video also can be viewed in real-time if there's a screen attached to Raspberry Pi. We adopt deep learning techniques and Convolutional Neural Networks to achieve facial expression recognition.

1.6 Organization

The rest of the thesis is organized as follows: Chapter II summarizes the recent related work: different methods have been used for face detection, feature extraction and classification. A brief history of CNN development is presented. We list the contribution of CNN and its contributors. Chapter III shows system setup process. For hardware, steps for Raspberry Pi initialization and basic environment setting are presented. For software, we illustrate the main python libraries and APIs used in this project. Chapter IV presents the methods that are used. Chapter V shows the visualization of trained CNN. The simulation and experimental results are given in Chapter VI. Finally, Chapter VII concludes the thesis.

CHAPTER II: RELATED WORK

Many studies and algorithms on facial expression recognition and analysis have been carried out for a long time. Pantic and Rothkrantz [4] pointed out that there are three basic problems for facial expression analysis: detecting face in an image or video frame, extracting facial expression feature information, and at last classifying the expression. In the following, we will discuss the distinct algorithms used in various stages of facial expression recognition.

2.1 Methodology

In 2001, Paul Viola and Michael Jones[14] used AdaBoost algorithm to do real-time face detection. Five years after the publication, in 2006, Fujifilm unveiled the first digital camera that has a real-time face detector. It was a very rapid transfer from basic science research to real world application.

Face detection is the first step of facial expression recognition. Image processing approaches for face detection are usually considered in two categories, feature-based and image-based approaches[15]. There are so many features in the human face, which can be used to distinguish a face from other objects. There are various feature-based approaches, which locate faces by extracting structural features like eyes, nose, mouth etc. and then uses them for face detection in real-time, such as adaptive skin color[15], Haar classifier[16], AdaBoost[17] and contour points[18]. Image-based face recognition has been an active area of biometrics research in recent years[19], such as linear subspace methods, neural networks[20] and statistical analysis methods[21].

Face detection methods that have been used mostly assume frontal upright face view in images or image sequences being analyzed. Viola & Jones[22] provided competitive face detection in real time, uses the AdaBoost algorithm to exhaustively pass

a search window over the image at various scales for rapid detections. Essa & Pentland [23] performed spatial and temporal filtering together with a threshold to identify the motion blobs from image sequences. Eigenface method[24] is also used to detect face. The PersonSpotter system[25] tracked the bounding box of the head in video using spatio-temporal filtering and disparity in pixels, thereby selecting the region of interests (ROIs) and then passing through skin detector and convex region detector to check for face.

The next step toward facial expression recognition is the extraction of features from face images. There are mainly two approaches: appearance based and geometric based. Littleworth et al.[26] used a bank of 40 Gabor filters at a different scale and orientation to apply convolution on the image. Essa & Pentland[23] used the same face extraction image to find facial feature points using a set of sample images via FFT and local energy computation. Cohn et al.[27] firstly normalized the feature points in the first frame of the image sequence and then used optical flow to track them. The displacement vectors for each landmark between the initial and peak frame represent the extracted information. Other methods including Active Appearance Models (AAM), T.F.Cootes et al.[28] used a statistical model of shape and gray-level appearance of the object of interest which can generalize to almost any valid example. The facial action coding system (FACS) is also used for feature extraction combined with other techniques, such as Dynamic Bayesian Network(DBN)[29] and Local Binary Pattern(LBP)[30].

The third step involves classification of emotions based on a training model constructed on the properties related to feature points extracted from last step. There are many classification algorithms or classifier such as Artificial Neural Network(ANN)[31], Hidden Markov model (HMM)[32], Support Vector Machine(SVM)[33] and Bayesian Network[34]. Image-based face recognition has been an active area of biometrics

research in recent years[19]. Given a database of suitably labeled training images of numerous individuals, a supervised or unsupervised pattern-recognition technique aspires to recognize the expression.

Deep learning methods have been increasingly popular this decade in artificial intelligence and machine learning field[35]. Deep learning has gained more attention as the amount of available training data has increased. It has solved more and more complicated applications with increasing accuracy over time[36]. Lately, Convolutional Neural Network(CNN) has become a hot topic due to its discriminative power. There have been some studies that used CNN to solve the problem of facial expression recognition. Gudi et al.[37] proposed a CNN model to recognize gender, race, age and emotion from face images. Tang and his team[38] used CNN combined with SVM to achieve their goal, also exploited augmentation techniques to create more data for training.

2.2 Brief History of Neural Networks

Neural networks are not new. There are different kinds of neural networks, such as Artificial Neural Network, Deep Belief Network and Recurrent Neural Network that can be used to do image recognition. The main reasons why Convolutional neural networks have become an important tool for object recognition are:

1. **Parameter Sharing:** A feature detector (such as a vertical edge detector) that is useful in one part of the image is probably useful in another part of the image.
2. **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

As of now, the most popular convolutional neural networks for object detection and object category classification from images are AlexNets, GoogLeNet, and ResNet50.

These convolutional neural networks are all winners of ImageNet large scale visual recognition challenge in recent years.

Table 2.1: Important contributions in the history of neural networks

Contribution	Contributor, year
LeNet[39] , starting the era of CNN	LeCun, 1990
AlexNet(SuperVision)[40], breakthrough	Krizhevsky, Sutskever & Hinton, 2012
GoogLeNet[41]	Szegedy, 2014
VGG[42]	Simonyan, Zisserman, 2014
ResNet(MSRA)[43]	He, Zhang, Ren & Sun 2015

A series of major contributions related to CNN is presented in Table 2.1, including LeNet[39], created by LeCun, used on pattern recognition tasks, which leads up to today’s “era of Convolutional Neural Network”. The LeNet architecture is an excellent “first architecture” for Convolutional Neural Networks. One of the most substantial breakthroughs in convnets was in 2012, when Krizhevsky et al.[40] introduced the AlexNet, a seven layers Convolutional Neural Network, with great performance for image classification at that time. After that, with the presence of efficient regularization techniques (e.g., dropout[44], batch normalization[45], and data augmentation), and powerful frameworks like TensorFlow[46], theano[47], and caffe[48], the architecture of CNN models are getting deeper and deeper. VGG[42] has 19 layers and ResNet[43] has 152 layer. Theoretically, more layers CNN has, the better results you will have. But this will also increase computational cost and can easily run out your memory. Thus, a more powerful GPU is required for such deeper CNN models.

2.3 Wireless Applications

As the development of wireless network technology, mobility is a key point of all innovation high-tech products. To capture face image, the most accessible mobile machine with camera available to us are smartphones. There have been some researches conducted based on smartphones[2]. According[2], Song and his team developed a real-time facial expression recognition function on a smartphone using a trained deep convolutional neural network on a GPU to classify facial expressions. They just used the front camera on a smartphone to perform image acquisition and then display the analysis result through an App installed on it. The model and the testing process are performed in a remote server.

CHAPTER III: SYSTEM SETUP

3.1 Hardware

3.1.1 Raspberry Pi

The computational module in this project is Raspberry Pi 3 model B. It is a Linux based platform that uses Python as a programming language. In Linux, software development is quite simple as it is an open source code development environment. The main parts of a Raspberry Pi board include processor, memory, HDMI port, Ethernet port, USB ports and abundant global interfaces. It does not include a built-in hard disk or solid state drive, instead, relying on an SD card for operation system booting, and storage. We have a SanDisk 64GB micro SD card preloaded with the official Raspbian OS. Figure 3.1 displays the external view of Raspberry Pi board.

Figure 3.1: Raspberry Pi 3 Model B+



Following are lists of key features of Raspberry Pi 3 Model B single board computer:

- Broadcom BCM2837 SoC processor with 1.2GHz 64-bit ARM Cortex-A53;
- 1GB LPDDR2 memory
- VideoCore IV GPU supports up to 1920×200 resolution
- 8Mpix Camera module capable of full HD video @30fps;
- 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE);

- MicroSD card slot, 10/100Mbps Ethernet port, $4 \times$ USB
- 2.0 ports, HDMI, audio/video jack, GPIO header, microUSB power port, DSI and CSI ports.

3.1.2 Camera Module

The Camera Module used in this project is Pi Camera, which is a great accessory for the Raspberry Pi. The Raspberry Pi camera module V2 can be used to take high-definition videos, as well as photographs. It utilizes the IMX219PQ image sensor from Sony, which offers high-speed video imaging and high sensitivity. The Raspberry Pi Camera Module offers reduced image contamination such as fixed pattern noise and smearing. It also features automatic control functions such as exposure control, white balance and luminance detection. Figure 3.2 shows that the Pi camera is connected to the Raspberry Pi via a CSI port which allows a USB port free. The CSI bus is capable of extremely high data rates.

Figure 3.2: Pi camera connected to Raspberry Pi



3.1.3 Hardware Setup

During the initial setup, Raspberry Pi was configured as a miniature computer with camera module and an external monitor for viewing the captured images and videos. The Raspberry Pi runs Raspbian OS and the programming language which is Python. We have the step by step setup guide listed below:

- Step 1. After you get the Raspberry Pi board and Pi camera module, follow the official Raspberry Pi camera installation guide to install it.
- Step 2. You will need to start up the Pi board and ensure the software is enabled.
- Step 3. Before the next step, you can run a quick sanity check to ensure that Raspberry Pi camera is working properly. We can use command *\$ raspistill -o check.jpg* to do the checking. This command activates Raspberry Pi camera module, displays a preview of the image, then after a few seconds, snaps a picture, and saves it to the current working directory as *check.jpg*.
- Step 4. Install picamera module for python. Notice in command line we should install *picamera[array]*, and not just *picamera*, the command line using *pip* to install is:
\$ pip install "picamera[array]"

While the standard *picamera* module provides methods to interface with the camera, we need the *array* sub-module so that we can utilize OpenCV. Because when using Python bindings, OpenCV represents images as NumPy arrays, and the *array* submodule allows us to obtain NumPy arrays from the Raspberry Pi camera module. After above four steps are done, it's ready for us to access the Raspberry Pi camera using Python and OpenCV.

3.2 Software

3.2.1 Python and OpenCV

The software used for system implementation are Python and OpenCV (Open Source Computer Vision Library). OpenCV is an open source computer vision and machine learning software library. In addition to C++ (and C), there is also growing support for Python as a simpler scripting language through a Python interface on top of the C++ code base[49]. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects and extract 3D models of objects.

In this project, we use OpenCV pre-trained Haar-Cascade classifier to do face detection in real-time. The Haar-Cascade is not the “best” algorithm for face detection, but its low computational complexity is more suitable for Raspberry Pi. After tuning the parameters, the performance of Haar-Cascade classifier is good enough for our system.

3.2.2 Keras

Keras is a Python Deep Learning library, a high-level neural networks API, which is written in Python and is capable of running on top of many frameworks, like TensorFlow, CNTK, and Theano. Through its user friendliness, modularity and extensibility, Keras allows easy and fast prototyping. It is one of the most powerful and easy-to-use Python libraries for developing and evaluating deep learning models.

Moreover, Keras can run seamlessly on CPU and GPU. All these features allow us to use Keras to build and train our CNN.

There are two main types of models available in Keras: the Sequential model, and the Model class used with the functional API. Here in our project, we use the Sequential model. The Sequential model is a linear stack of layers. We need to notice that the first layer in a Sequential model needs to receive information about its input shape. In machine learning the shape of input image is a three dimensional tuple, formatted as (Height, Width, Channel). There are two ways to shape the input, one is channel first and the other is channel last.

During the training process, it's easy to adjust the current architecture, just simply add layers via the `.add()` method:

```
model = Sequential()  
model.add(Dense(32, input_dim=784))  
model.add(Activation('relu'))
```

3.2.3 TensorFlow

TensorFlow is an open source software library for high performance numerical computation, developed by researchers and engineers from the Google Brain team within Google's AI organization. Its flexible architecture allows easy deployment of computation across a variety of platforms, like CPUs, GPUs, and TPUs (A tensor processing unit is an AI accelerator application-specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning.), and from desktops to clusters of servers to mobile and edge devices. It comes with strong support for machine learning and deep learning. The flexible numerical computation core is used across many other scientific domains.

TensorFlow is a suite of software, an ecosystem for developing deep learning models. It contains all the tools right from building to deployment. TensorFlow has 3 main components: TensorFlow(API), TensorBoard and TensorFlow Serving.

TensorFlow(API). This component of TensorFlow contains the API's to define the models and train the models with data. The actual computation was written in C++ though it is accessed with python API's. These advantages are two fold: first, we get the user-friendly python interface to develop the models; second, we can run the models via fast and efficient compiled C++ code.

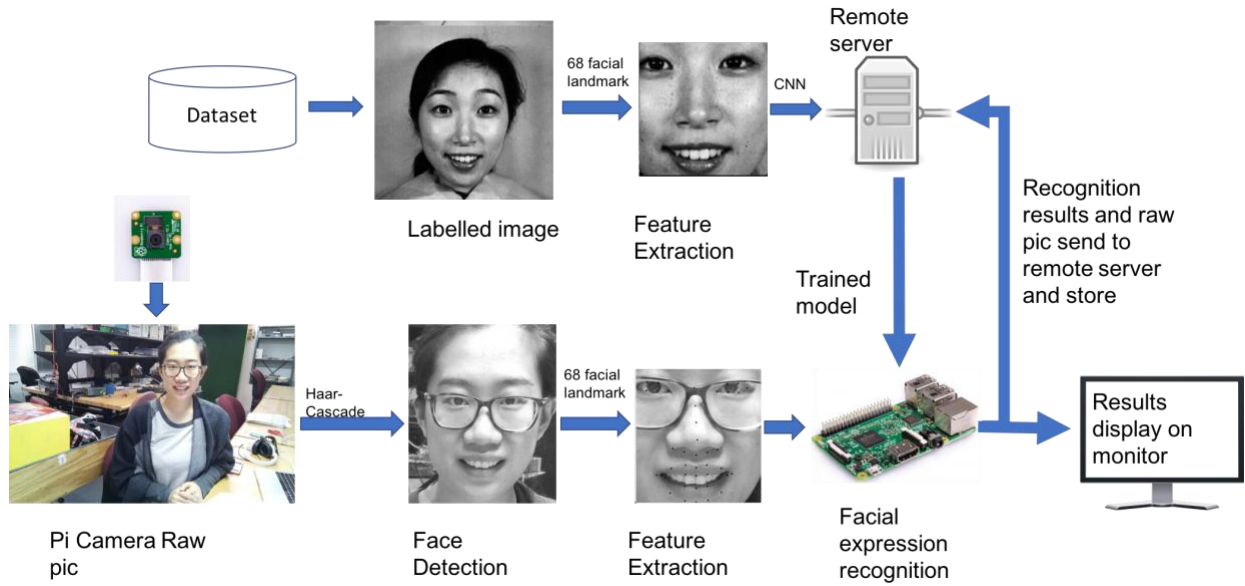
TensorFlow Serving. This component of TensorFlow helps to deploy the pre-trained models. TensorFlow serving is capable of switching from old models to new models without any downtime. This is the stand out feature in the ecosystem. This is also written in C++ and can be accessible with python interfaces.

TensorBoard. The third component of the ecosystem and the boon for engineers is the TensorBoard. It helps to analyze, visualize, and debug TensorFlow graphs.

3.3 System Diagram

Our proposed system follows the main steps similar to ordinary facial expression recognition systems such as face detection, feature extraction, and facial expression classification. However, there are other steps to generate features between neutral and expression frames. The system is composed by following parts: Pi camera to capture an image, and Raspberry Pi board to run image recognition programs. Compatible monitors also connected with this system during initial stages to preview the captured images and show the user. The system diagram is shown in Figure 3.3.

Figure 3.3: Facial expression recognition system overview.



The whole system contains two parts, model training and raw input image expression classification. In the model training part, we used all frontal face, head centered with blank background images from the datasets. There is no need to waste computation on face detection. The labeled image from dataset will go directly to feature extraction, where we use 68 facial landmarks as prototype to number the common facial specific features we will use to feed to the following CNN architecture. After feature extraction, we have a clean face image with facial parts that will contribute to an expression. This face image then is resized to (64×64) . Before starting to train CNN classifier, we further pre-process our input data by scaling the data points from $[0, 255]$ to the range $[0, 1]$, also convert the label to the image to vectors using one-hot encoding (One-hot encoding transforms categorical features to a format that works better with classification and regression algorithms.). We then perform a training/testing split on the data using 75% of the images for training and 25% for testing. Subsequently, we

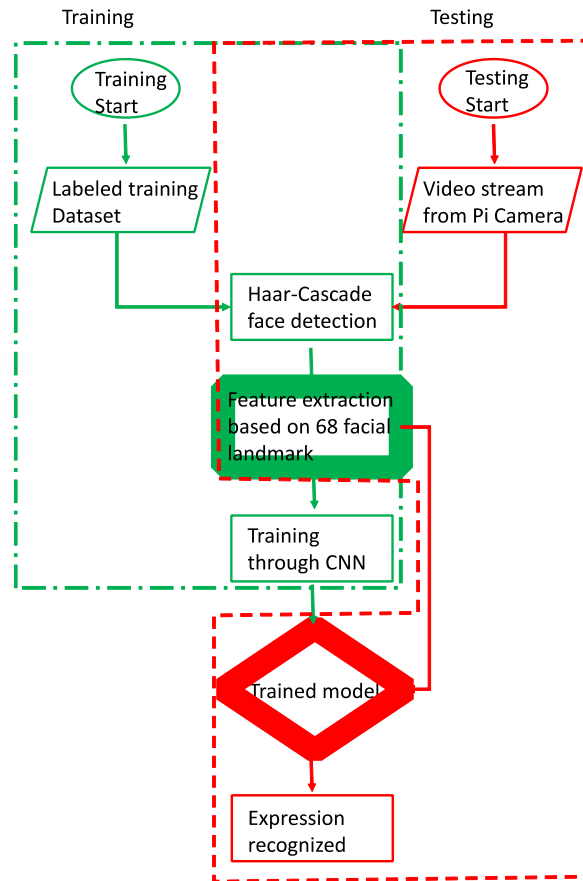
will perform some data augmentation, enabling us to generate “additional” training data by randomly transforming the input images using the parameters, 30 degree range for random rotations, random horizontal or vertical shifts, shear intensity (Shear angle in counter-clockwise direction as radians), small range for random zoom, and randomly flip inputs horizontally. This allows us to use a smaller dataset and still achieve good results. The next step is training the CNN. The trained CNN model will be loaded to Raspberry Pi board. The detailed CNN architecture will be illustrated specifically in Section IV.

Facial expression classification in real-world applications uses the same methodology described in model training previously. While the Pi Camera video stream is on, if a motion is detected, a sequence of pictures will be taken. Unlike the dataset used for model training, the position of human face may not be in the center of the natural picture, also the face is probably posted at a weird angle than a frontal face. Use Haar-Cascade classifier, we can quickly detect if there is a face in the taken picture. If it is true, the next step will crop the face region out for future processing. If not, this picture will be discarded. Similar to the training step, in feature extraction, it further reduces the face region to extract the useful facial parts and resize to the same dimension as the input size required for our CNN model. Then the same data pre-processing techniques as previous are deployed. Instead of training, in testing, we already have the trained model loaded on Raspberry Pi board. All we need to do is just invoke the model to do classification. The output will be a classified expression result with its classification confidence value.

CHAPTER IV: METHODS

There are several steps to implement the real-time facial expression recognition system: face detection, feature extraction, and classification of facial expression. In this section, we will discuss the approaches in detail toward real-time facial expression recognition, in terms of the algorithms and techniques used at various steps. Our proposed system uses gray-scale frontal face images to classify six basic emotions: happy, sad, disgust, fear, surprise, anger plus neutral face. An overview of the system flowchart is illustrated in Figure 4.1.

Figure 4.1: Flow Chart



4.1 Face Detection

Face detection is the first important step of facial expression recognition analysis. Whether it is natural images taken by Pi Camera in real-time or face images from database, they all contain useless information from surrounding environment. After the Pi board has been initialized, the Pi Camera will be turned on and generate video streams constantly. The demand for storage will exponentially increase if we record all the video frames. In order to avoid large datasets, we program the camera model to take photo periodically only when it detects motions. Motion detection works on the basis of frame difference, that means comparing how pixels (usual blobs) change location after each frame. The method looks for an object change in the image. We take the weighted mean of previous frames along with the current frame, so our algorithm can dynamically adjust to the background, even as the time of day changes as well as the lighting conditions.

Now we have an image, the image needed to be segmented into two parts: face region and non-face region. We will use Haar Cascade technique to fulfill this. Haar feature-based cascade classifiers were proposed by Paul Viola and Michael Jones in[22]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. One of the great things about OpenCV is that it already contains many pre-trained classifiers for face, eyes, smile etc. We can directly use the pre-trained face classifier to crop the face in an image. There is a cropped face image in Figure 4.2 for example from our process.

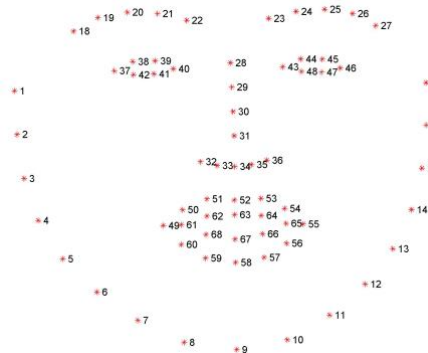
Figure 4.2: Cropped Face from Face Detection Stage



4.2 Feature Extraction

Facial feature extraction can be defined as a process to locate specific features in a facial image such as points or contours. The features can be physical such as eyes, eyebrows, mouth, nose, etc. As we already have face located at the center of the cropped image, the computational complexity of the facial feature extraction can be significantly reduced. From Figure 4.2 we can see that there is still background on the cropped face image. The background information will do nothing for the expression recognition. In fact, it will reduce the recognition accuracy by introducing misleading features. We only care about the features which can contribute to an expression, e.g. big smile with open mouth meaning happy; eyes wide open with round mouth meaning surprise; transverse nasal lines will appear when disgusting etc..

Figure 4.3: Visualization of the 68 facial landmarks

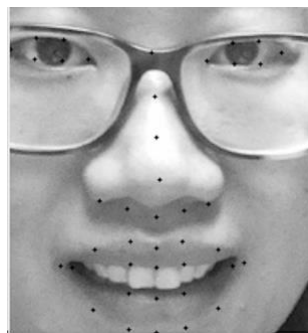


[50] provides the 68 and 51 distinguish points facial landmark localization. Our feature extraction method is based on the 68 points landmark, which is shown in Figure 4.3. Through examining the 68 points landmark, we can find that different facial regions can be accessed via Python indexing (zero- indexing with Python Vs one-indexing in Figure 4.3):

- The jaw denoted by [0, 17];
- The right eyebrow through points [17, 22];
- The left eyebrow through points [22, 27];
- The nose using [27, 35];
- The right eye using [36, 42];
- The left eye with [42, 48];
- The mouth going through points [48, 68].

The features we can use to distinguish expressions are mostly around nose, eyes and mouth areas, so we just examine the points range [27, 68] and crop out the feature image. The extracted feature face image is shown in Figure 4.4, we can see that the noisy background information has been cropped out.

Figure 4.4: Feature Extracted based on 68 Facial Landmark.



4.3 Facial Expression Recognition

There are various approaches for facial expression recognition in real-time applications. Two phases appear in most of these approaches. The first phase is training phase in which the input is a set of instances with its feature values and class label as one of the expressions. The training data includes examples of every expression with sufficient data for training. The second phase is classification, which categorizes the unknown emotion into seven different classes (i.e. happy, sad, fear, disgust, anger etc.). Deep learning with Convolutional Neural Network has been used to establish solutions for image classification. When human beings look at a picture, they can classify it subconsciously as if the picture has identifiable features. In a similar way, the computer is able to perform image classification by looking for lower level features such as edges and curves, and then builds up more abstract concepts through a series of convolutional layers.

4.3.1 Convolutional Neural Network

Before moving to the model training, we will discuss the detailed architecture of CNN. When designing the CNN, we need to know how many layers, what kind of layers needed, and what's the size of the filter for each layer, etc.. We want to find the balance between the complexity of the architecture and the computation time to train a model that does not overfit. Training the CNN actually is selecting the best weights for all of the neuron connections. The weights are learned using an algorithm called back propagation. Of course we don't have to do this manually, there are several open source libraries that make training neural networks fairly straightforward, like Caffe in C/C++, Keras which is used in this project, Lasagne and Theano in Python.

Neural networks are usually composed of several layers of interconnected neurons. A CNN comprises several types of neural layers sequentially and each of these layers plays a different role. We use three main types of layers to build CNN architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. We also use two other types of layers, Batch Normalization layer, and Dropout Layer to accelerate training time and reduce overfitting in this project. We will stack these layers to form a full ConvNet architecture. Except for the layers before, activation functions also play an important role for non-linearity, which help neural networks to learn and make sense of complicated features and Non-linear complex functional mappings between the inputs and response variable. We use a non-linear function ReLU after each convolutional layer. Before output layer, we use a softmax function to classify our seven categories of facial expressions.

Convolutional Layer. In convolutional layers, the input is convolved with a filter map, containing the shared weights to produce a feature map. Let ω be the weight or filter with a filter size $m \times n$ applied to the input v . One convolutional layer result can be calculated as written:

$$c_{x,y}^k = b^k + \sum_{i=0}^{m/2} \sum_{j=0}^{n/2} w_{ij}^k v_{x+i,y+j} \quad (1)$$

Where $c_{x,y}$ is the convoluted results in the position (x, y) ; b^k is the bias at the k -th feature map.

Rectified Linear Unit(ReLU). ReLU layer applies an elementwise activation function. The result c is passed through a non-linear activation function:

$$\text{ReLU}(c) = \max(0, c) \quad (2)$$

Using this function is more efficient than sigmoid and more suitable for deep neural network architectures[51]. This helps to create a sparse feature representation. Moreover, the computational complexity is low.

Batch Normalization Layer. Batch Normalization normalizes input features to mean 0 and variance 1 that can speed up learning process. It makes neural network much more robust to the choice of hyperparameters in much bigger range, for example, batch normalization can use larger learning rate. More importantly, it makes training process faster even for very deep networks.

$$\mu = \frac{1}{m} \sum_i z^{(i)} \quad (3)$$

$$\sigma^2 = \frac{1}{m} \sum_i (z^{(i)} - \mu)^2 \quad (4)$$

$$z_{norm}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}} \quad (5)$$

where μ is the mean value of input features z , σ^2 is the standard deviation of input features z . $z_{norm}^{(i)}$ is the output of layer i after Batch Normalization. The use of ε is to avoid the division by zero.

Max Pooling Layer. Max pooling can lead to faster convergence, select superior invariant features, and improve generalization[52]. In the max-pooling layer, a region of the previous layer is connected to a unit in the current layer. The max-pooling reduces the dimensions (width \times height) of input volume by applying the maximum function for the next layer. The pooling layer does not affect the depth dimension of the input. They simply subsample some $(k \times k)$ region and output a single value, which is the maximum in that region. For instance, if their input layer is an $(M \times M)$ dataset, they will then output results, as each $(M \times M)$ region $\frac{M}{k} \times \frac{M}{k}$ is filtered to just a single value via the max function.

Dropout Layer. Dropout is a technique used to improve overfitting on neural networks, by randomly setting a fraction rate of input units to 0 at each update during

training time. Dropout layers can be used on the fully connected layers, as well as after the max-pooling layers. It can create some bit of image noise augmentation. Dropout is only used during the training of a model and is not used when evaluating the performance of the model. Generally, use a small dropout value of 20%-50% of neurons. A probability value too low has minimal effect and a value too high results in underfitting by the network.

Dense/Fully Connected Layer. After multiple convolutional and max-pooling layers, the high-level reasoning in the neural network is done via fully connected layers. The output layer has one neuron per class in the classification task.

Softmax. Softmax classifier is a generalization of the binary Logistic Regression classifier to multiple classes, which uses cross-entropy loss function. It gives a slightly more intuitive output (normalized class probabilities). The mapping function f is defined such that it takes an input set of data x and maps them to the output class labels via a simple (linear) dot product of the data x and weight matrix W :

$$f(x_i; W) = Wx_i \quad (6)$$

The cross-entropy loss that has the form:

$$L = -f_{y_i} \sum_j^n e^{f_j} \quad (7)$$

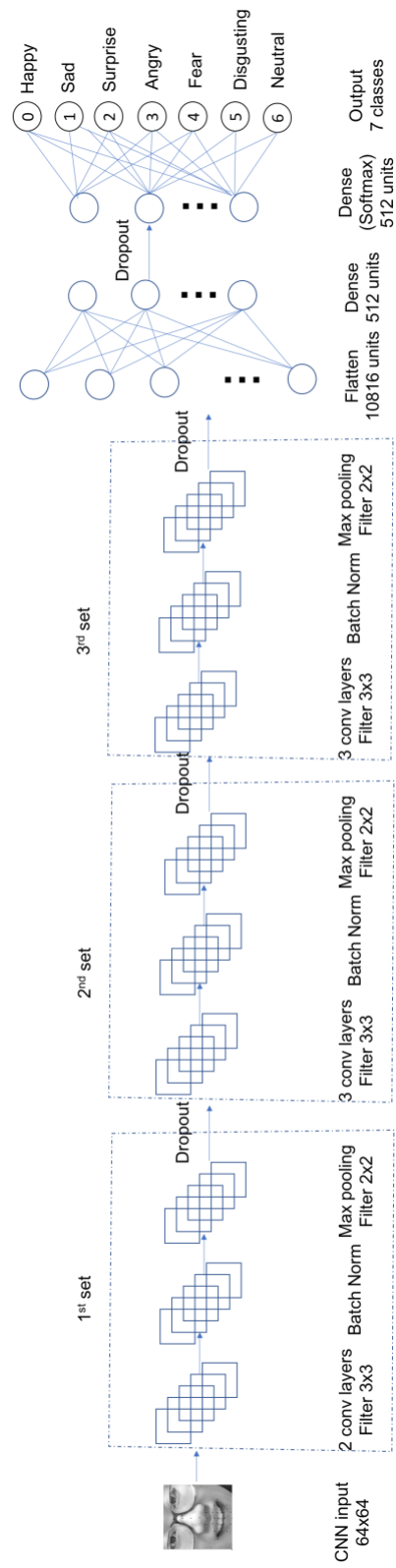
where the notation f_y means the j -th element of the class vector which scores f .

4.3.2 CNN Architecture

The architecture of CNN is shown in Figure 4.5. The first layer is called the input layer, each neuron corresponds to an input feature (in our case, a pixel). The size (height \times width) of input greyscale image in our project is 64×64 , that means the input layer has 4096 units. Next we apply three sets of convolutional, batch normalization, and ReLU layers. There are one Max pooling layer and one Dropout layer followed by each set. The

first set of convolutional layer has 32 filters with size 3×3 , while the second set has 64 filters with size 3×3 and the third set has 128 filters with the same size on each convolutional layer in each set. The more filters there are, deeper the training will be. The Max pooling size is 2×2 that can use down sampling to half the size of its input. In case of overfitting, we add dropout layers. Except for the three dropout layers behind each max pooling layer, there is another one between the two dense layers.

Figure 4.5: Architecture of the Convolution Neural Network

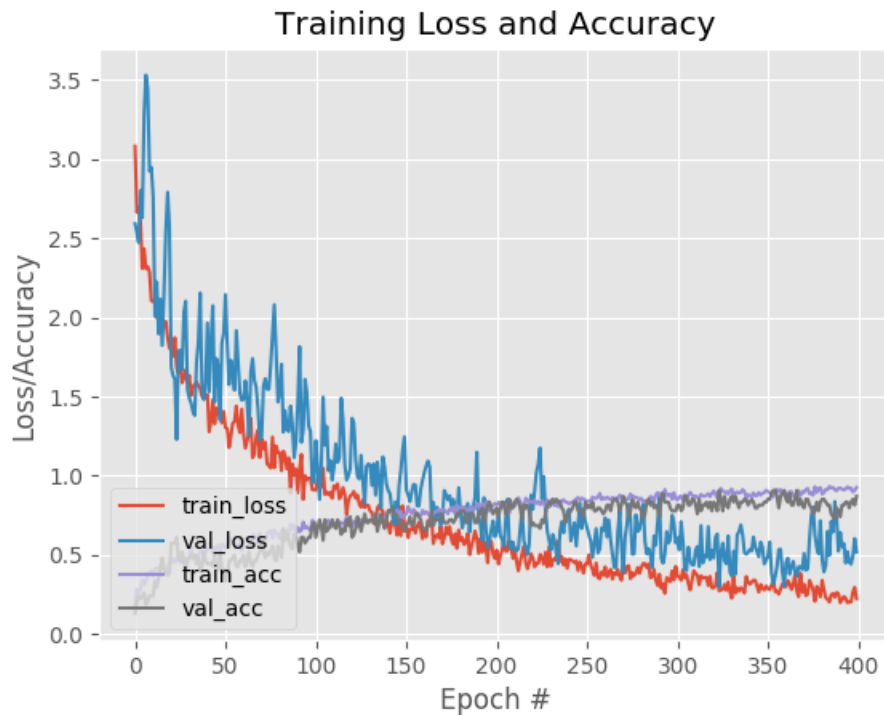


The CNN comprises eight convolutional layers, ten batch normalization layers and three max pooling layers, two dense layers and four dropout layers. The detailed architecture information also showed in Table 5.1 in Chapter V.

4.3.3 CNN Training Process

The history of training and testing loss and accuracy of our convolutional neural network model are shown in Figure 4.6. The red curve and purple curve represent training loss and training accuracy. The blue curve and black curve represent testing loss and testing accuracy distinctively. Initially, both training and testing accuracy were noted very low but gradually improved at each training epoch and almost reached one in the end as shown in Figure 4.6. Similarly, the training and testing loss was very high at the beginning but almost reached zero in the end as shown in Figure 4.6. This clearly shows the convergence of the proposed well-tuned deep convolutional neural network. The difference between training accuracy and testing accuracy is very small which means our trained CNN model is robust and has no overfitting.

Figure 4.6: Loss and Accuracy



After calculation, the training loss is 0.07 and training accuracy is 0.98; testing loss is 0.52 and testing accuracy is 0.87. The results match with the figure above.

4.3.4 Confusion Matrix

After training, we need to do some assessment on our model. A confusion matrix is a table that is used to describe the performance of a classification model on a set of test data for which the true labels are known. We print out confusion matrix for our trained model to check the performance, shown in Table 4.1. There are three terminologies we should know before interpreting the confusion matrix. They are Precision, Recall and F1-score. Precision is of all the classes how many we predicted correctly, it should be high as possible; Recall is of all the positive classes, how many we predicted correctly, It should be high as possible; It is difficult to compare two models with low precision and high

recall or vice versa. So to make them comparable, we use F1-Score. F1-score helps to measure Recall and Precision at the same time.

Table 4.1: Confusion Matrix

	Precision	Recall	F1-score
Happy	0.90	1.00	0.95
Sad	0.87	0.93	0.90
Surprise	1.00	0.92	0.96
Angry	0.80	0.50	0.62
Fear	0.67	1.00	0.80
Disgusting	1.00	0.67	0.80
Neutral	0.75	1.00	0.86
avg	0.88	0.87	0.86

From the table we can see that each row of the matrix represents the measurements for the corresponding class at that row. There are always trade-off between Precision and Recall, so we use F1-score to present the robustness of our model. Except Angry, all other six classes have great performance.

CHAPTER V: VISUALIZATION

Neural network models are associated with a mystery word – “Black Box”. In order to interpret and understand our trained model, we present how to visualize a convolutional neural network. It is absolutely crucial that we know what our trained model is doing, and how it’s making decisions on its predictions. There are several reasons why we need to do visualization:

1. Understanding how the trained model works
2. Assisting in Hyperparameter tuning
3. Finding out the failures in the model and determining why they fail

5.1 Methods of Visualizing a CNN model

5.1.1 Preliminary Methods

The simplest way to plot the model is using Keras function `model.summary()`. From Table 4, we can get information like the output shapes of individual layers of neural network and the numbers of parameters in each layer. Input and output layers are not shown in Table 5.1. We already know that the size of input image is $64 \times 64 \times 1$ (height \times width \times channel) as greyscale image has one channel. From Figure 4.5 in Chapter IV, we can see the architecture of the trained CNN has three sets of Conv \rightarrow ReLU \rightarrow Batch Norm \rightarrow Max Pooling \rightarrow Dropout; followed by two sets of fully connected layers; at last use softmax to categorize seven classes facial expressions. In Table 5.1, except all that architecture structures, we also can know the output shape() of each layer, how many parameters there are in this particular layer. The number of the parameters can be used to calculate the memory needed for computation.

Table 5.1: Plotting model architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	320
activation_1 (Activation)	(None, 64, 64, 32)	0
batch_normalization_1 (Batch Normalization)	(None, 64, 64, 32)	128
conv2d_2 (Conv2D)	(None, 64, 64, 32)	9248
activation_2 (Activation)	(None, 64, 64, 32)	0
batch_normalization_2 (Batch Normalization)	(None, 64, 64, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 32)	0
dropout_1 (Dropout)	(None, 32, 32, 32)	0
conv2d_3 (Conv2D)	(None, 32, 32, 64)	18496
activation_3 (Activation)	(None, 32, 32, 64)	0
batch_normalization_3 (Batch Normalization)	(None, 32, 32, 64)	256
conv2d_4 (Conv2D)	(None, 32, 32, 64)	36928
activation_4 (Activation)	(None, 32, 32, 64)	0
batch_normalization_4 (Batch Normalization)	(None, 32, 32, 64)	256
conv2d_5 (Conv2D)	(None, 32, 32, 64)	36928
activation_5 (Activation)	(None, 32, 32, 64)	0
batch_normalization_5 (Batch Normalization)	(None, 32, 32, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 64)	0
dropout_2 (Dropout)	(None, 16, 16, 64)	0

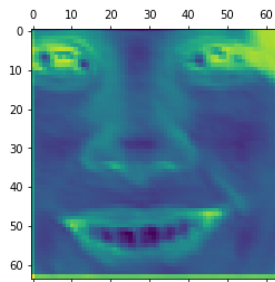
Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 16, 16, 128)	73856
activation_6 (Activation)	(None, 16, 16, 128)	0
batch_normalization_6 (Batch Normalization)	(None, 16, 16, 128)	512
conv2d_7 (Conv2D)	(None, 16, 16, 128)	147584
activation_7 (Activation)	(None, 16, 16, 128)	0
batch_normalization_7 (Batch Normalization)	(None, 16, 16, 128)	512
conv2d_8 (Conv2D)	(None, 16, 16, 128)	147584
activation_8 (Activation)	(None, 16, 16, 128)	0
batch_normalization_8 (Batch Normalization)	(None, 16, 16, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_3 (Dropout)	(None, 8, 8, 128)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 1024)	8389632
activation_9 (Activation)	(None, 1024)	0
batch_normalization_9 (Batch Normalization)	(None, 1024)	4096
dense_2 (Dense)	(None, 1024)	1049600
activation_10 (Activation)	(None, 1024)	0
batch_normalization_10 (Batch Normalization)	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175

Layer (type)	Output Shape	Param #
activation_11 (Activation)	(None, 7)	0
Total params: 9,928,103		
Trainable params: 9,922,727		
Non-trainable params: 5,376		

5.1.2 Activation Maps

Activation Map is the output of a particular convolutional layer. In order to visualize what the neural network is doing, we can apply the filters over an input image and then plot the output. This allows us to understand what kind of information or what sort of input patterns activate a particular filter. For example, we randomly picked the 30th filter from the first convolutional layer, the output after the filter is showed in Figure 5.1.

Figure 5.1: Output of 30th filter of the first convolutional layer

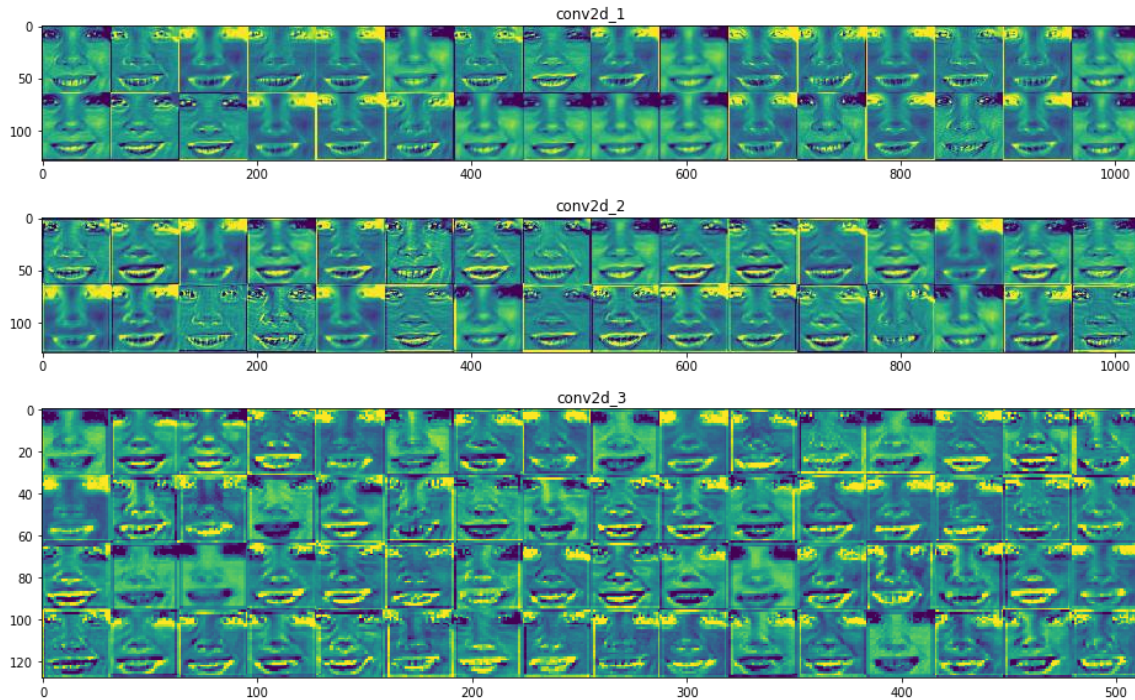


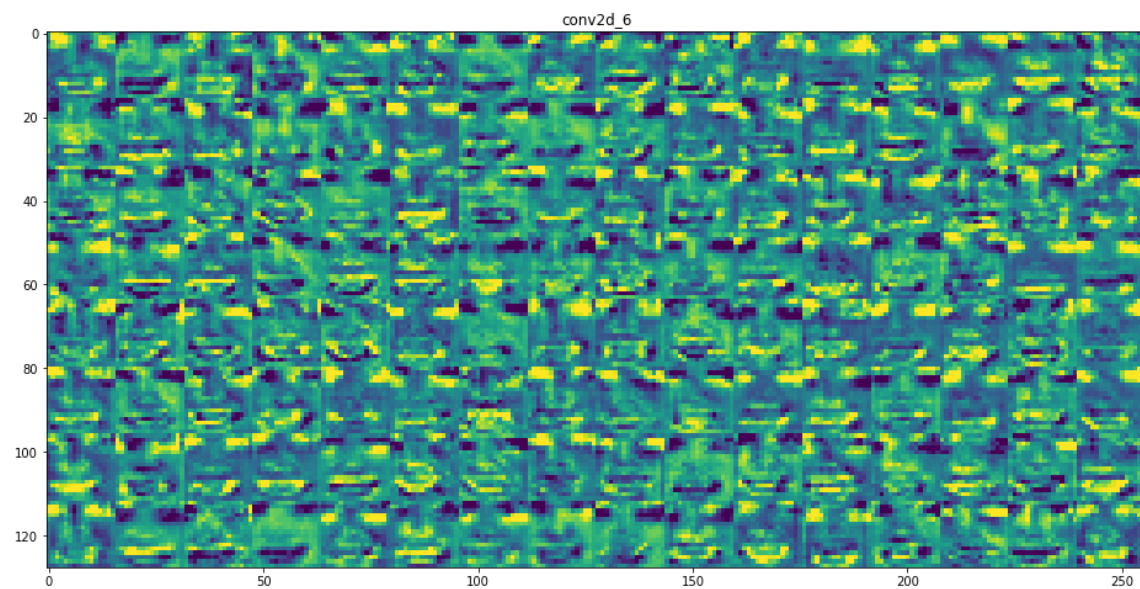
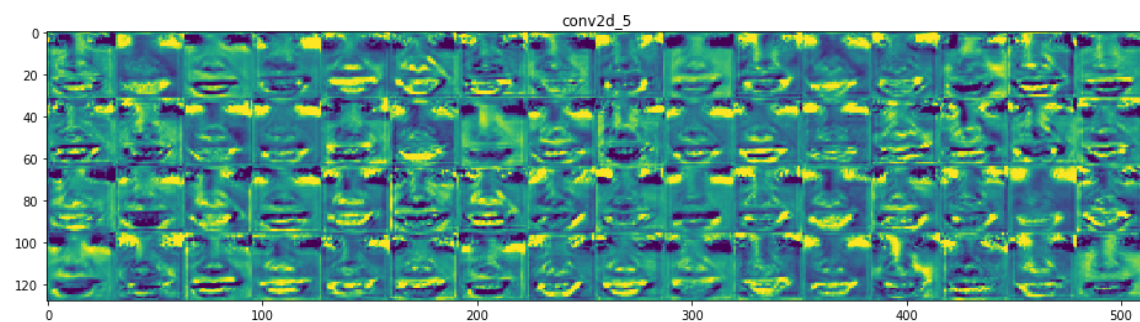
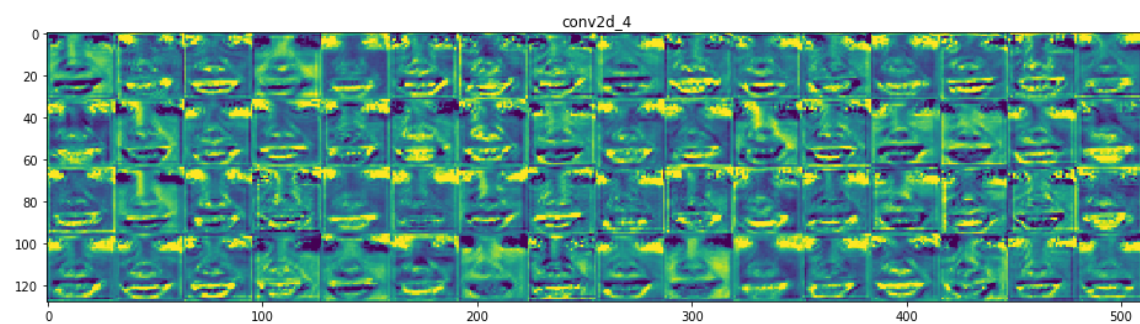
From the Figure 5.1, we can see that the output is like a X-ray image, it highlighted the contour of the facial organs. To have a close look at the activation map, all the outputs filtered in the eight convolutional layers are pulled out and are shown in Figure 5.2. Each layer are displayed in a different block. Every box shows an activation map corresponding to some filter. We can see the first convolutional layer (conv2d_1)

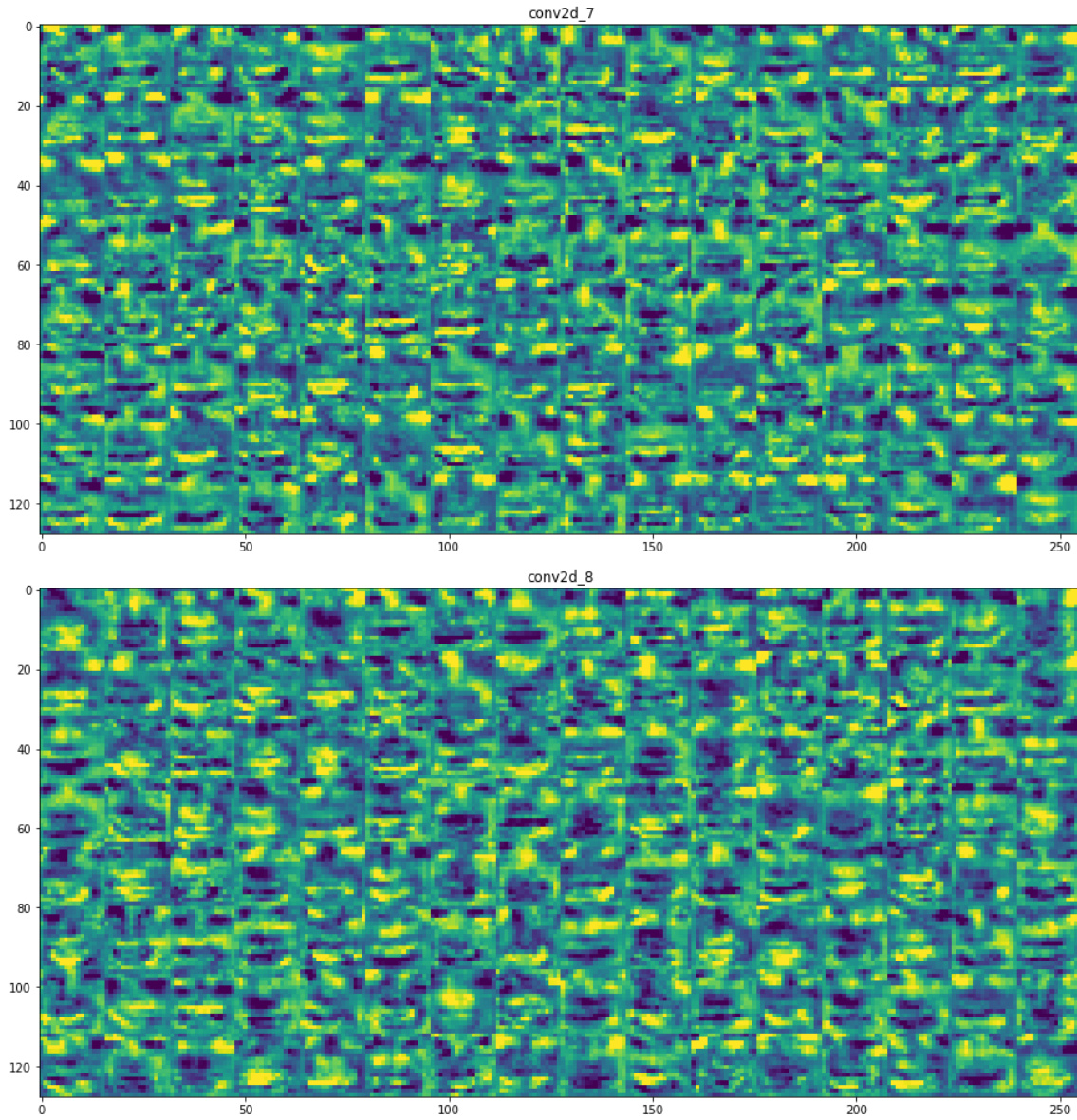
learns to detect edges from raw pixels, then used the edges to detect simple shapes in the second layer, and then uses these shapes to detect higher-level features.

From the first three convolutional layers, conv2d_1 to conv2d_3, we can still see the whole facial parts, eyes, nose and mouth. That means the filters on the first three layers are looking for edges or some low level features that can form the whole face. The following two convolutional layers, conv2d_4 and conv2d_5, the nose area become blurred. And the last three convolutional layers, conv2d_6 to conv2d_8, all the images become abstract pieces. I assume in these layers, filters are looking for high level features that can be used key points to determine which facial expression this is.

Figure 5.2: Learned features from all eight convolutional layers





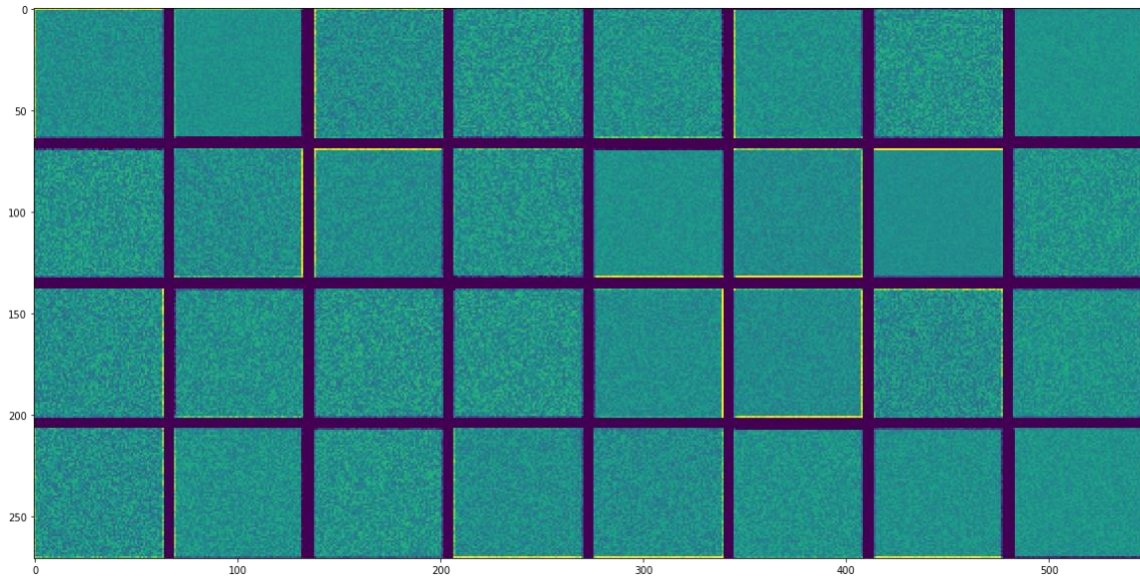


5.2 Layer Activation

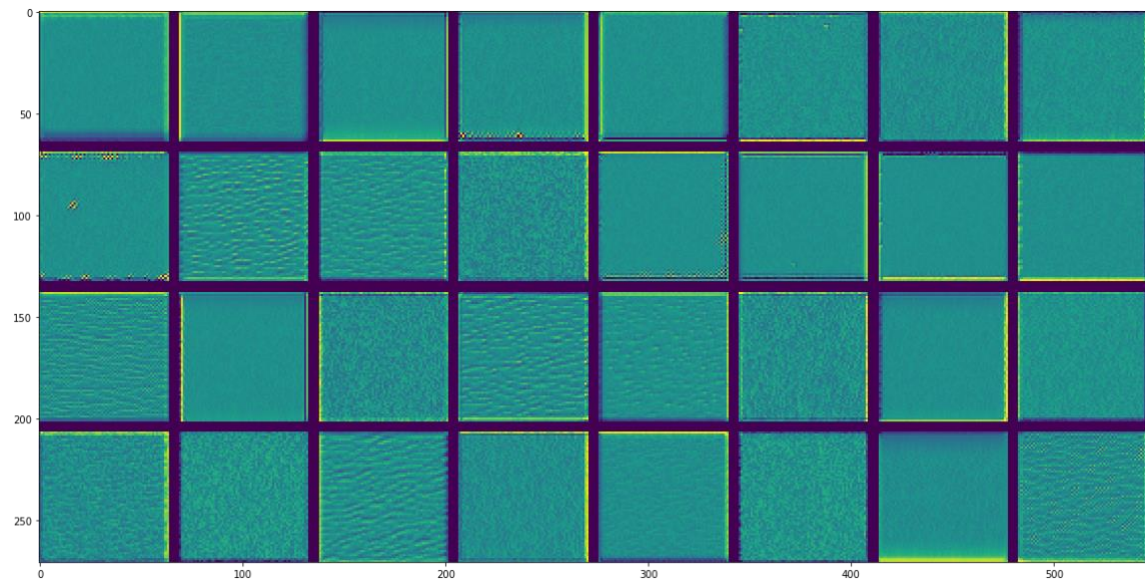
Using the model described in Chapter IV, we now visualize the feature activations. Figure 5.3 shows feature visualizations from our model. For a given feature map, we show the 8 convolutional layers' activations, each projected separately down to

pixel space, revealing the different structures that excite that map and show its invariance to input deformations.

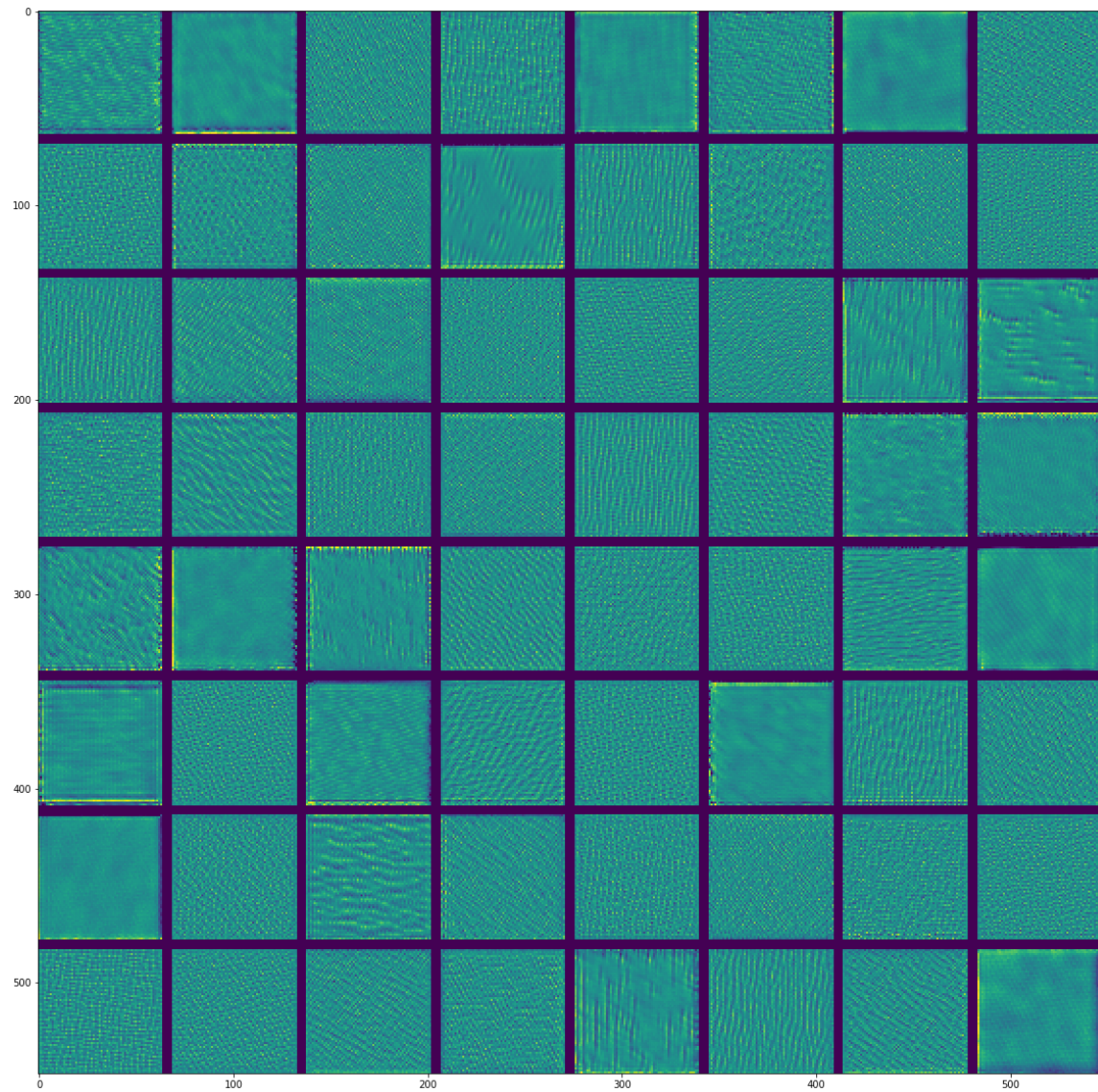
Figure 5.3: Visualization of features in a fully trained model



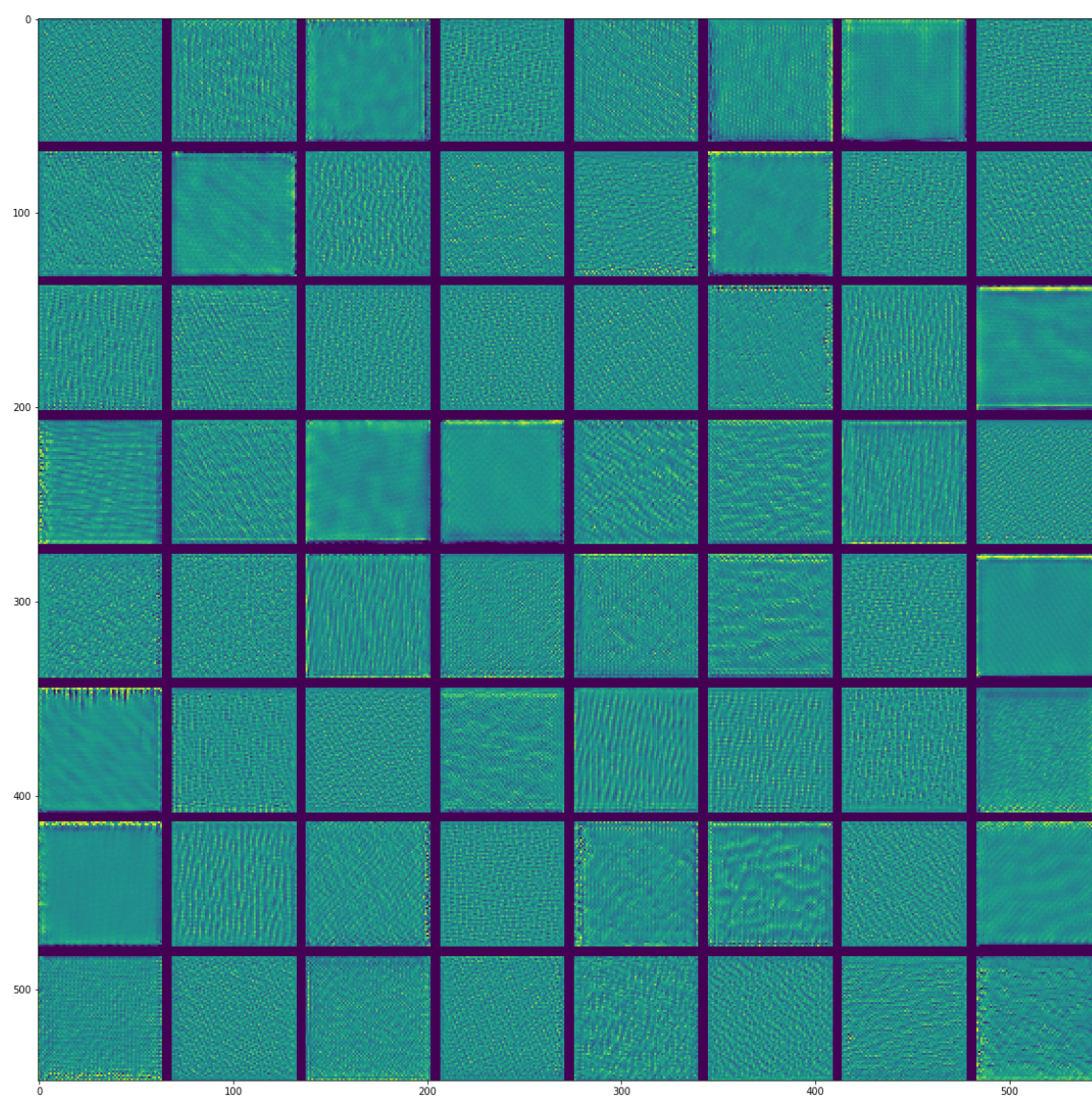
(a) Conv2d_1, 32 filters with size 3×3 : Conv2d_1 is the first convolutional layer of our trained model, from above figure, it seems there is no clear evidence of what the filters are looking for.



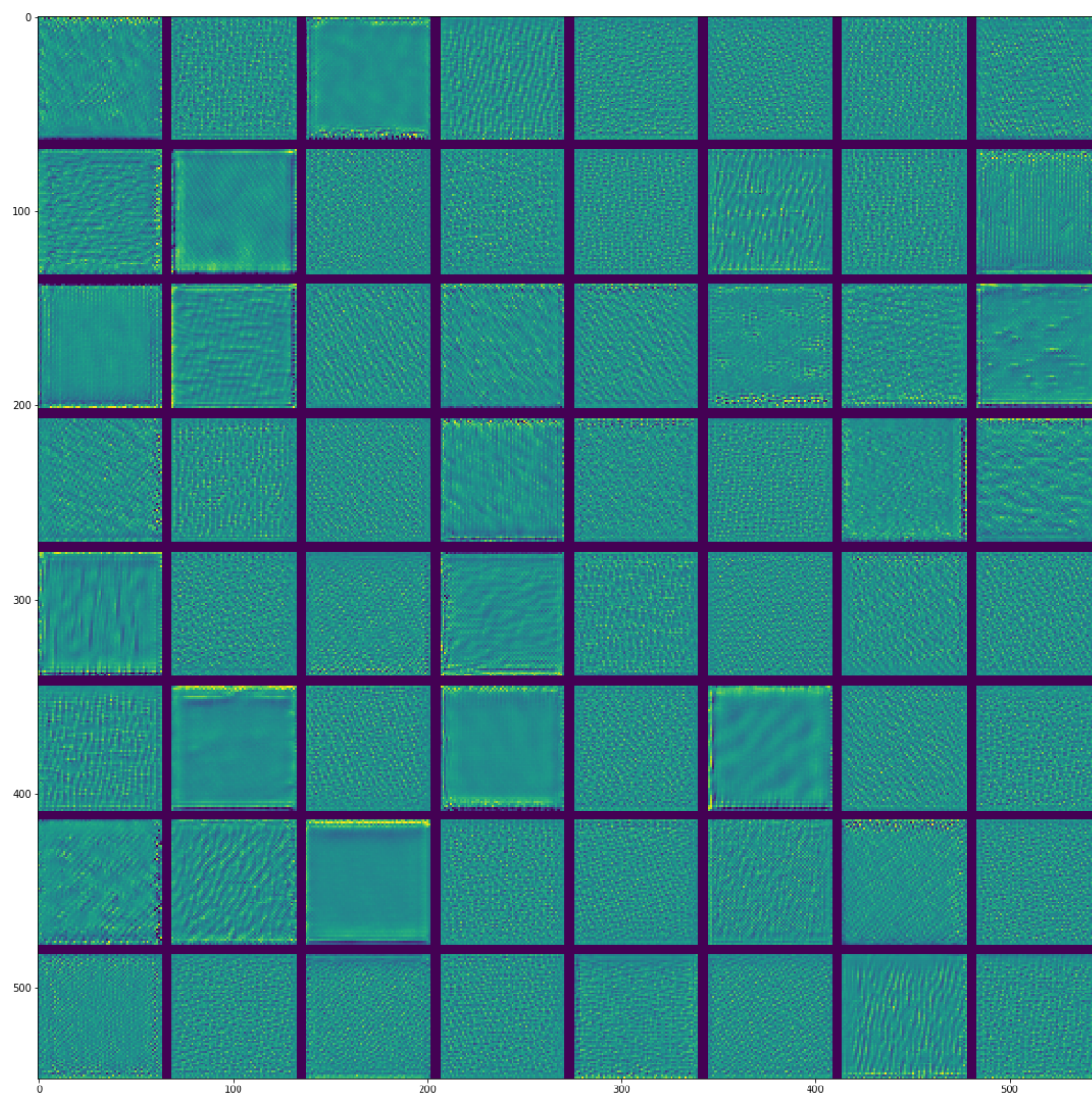
(b) Conv2d_2, 32 filters with size 3×3 : From the second convolutional layer, there seems to be some features filtered. For example, in the filter in second row, second column, the filter has captured some horizontal textures.



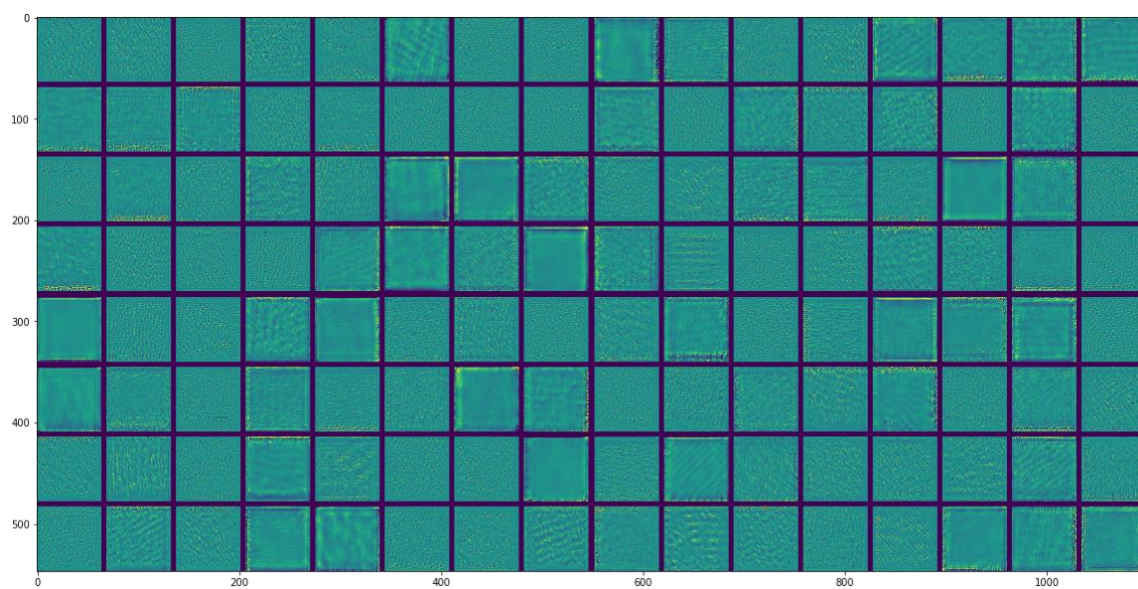
(c) Conv2d_3, 64 filters with size 3×3 : There are more features from the third convolutional layer. We can see horizontal, vertical, diagonal and other features have been filtered. That's corresponding to section 5.1.2, filters are looking for edges of contour of face or facial parts. Same results of the following two convolutional layers, conv2d_4 and conv2d_5.



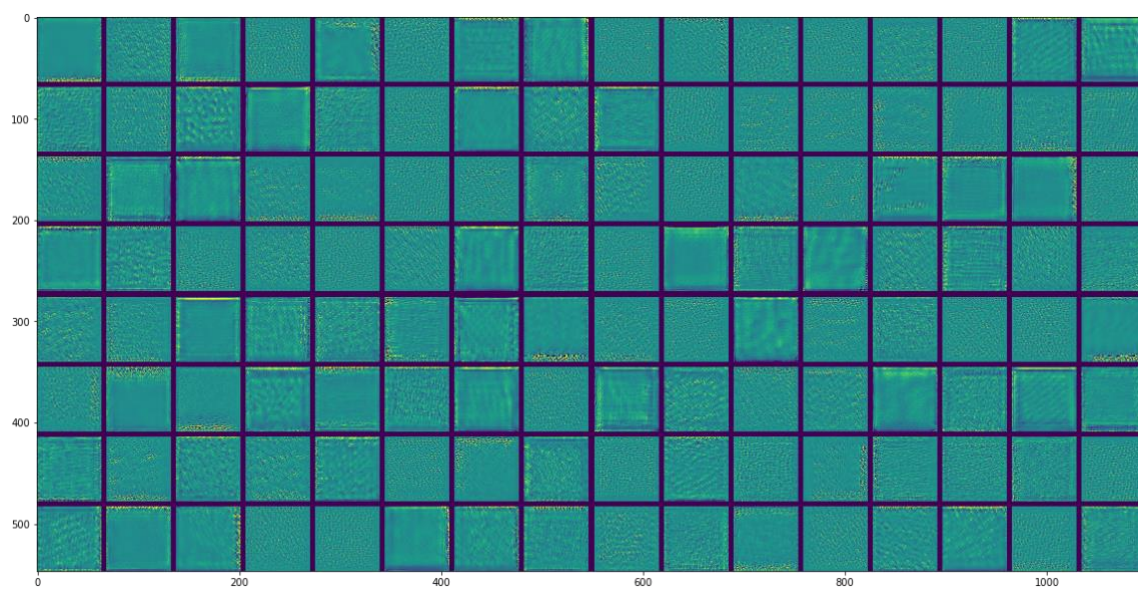
(d) Conv2d_4, 64 filters with size 3×3



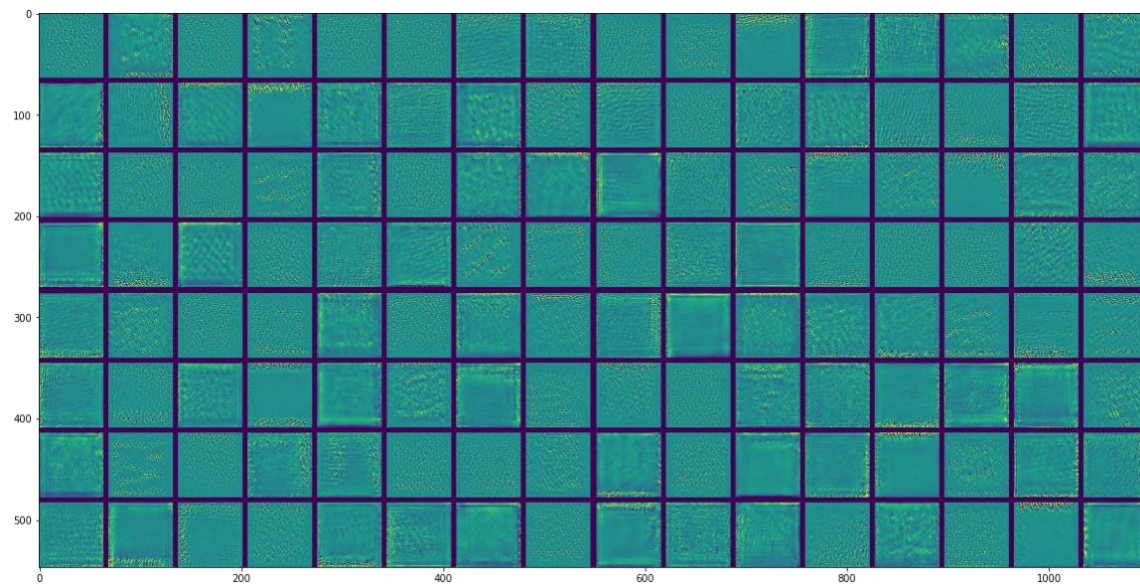
(e) Conv2d_5, 64 filters with size 3×3



(f) Conv2d_6, 128 filters size 3×3



(g) Conv2d_7, 128 filters with size 3×3



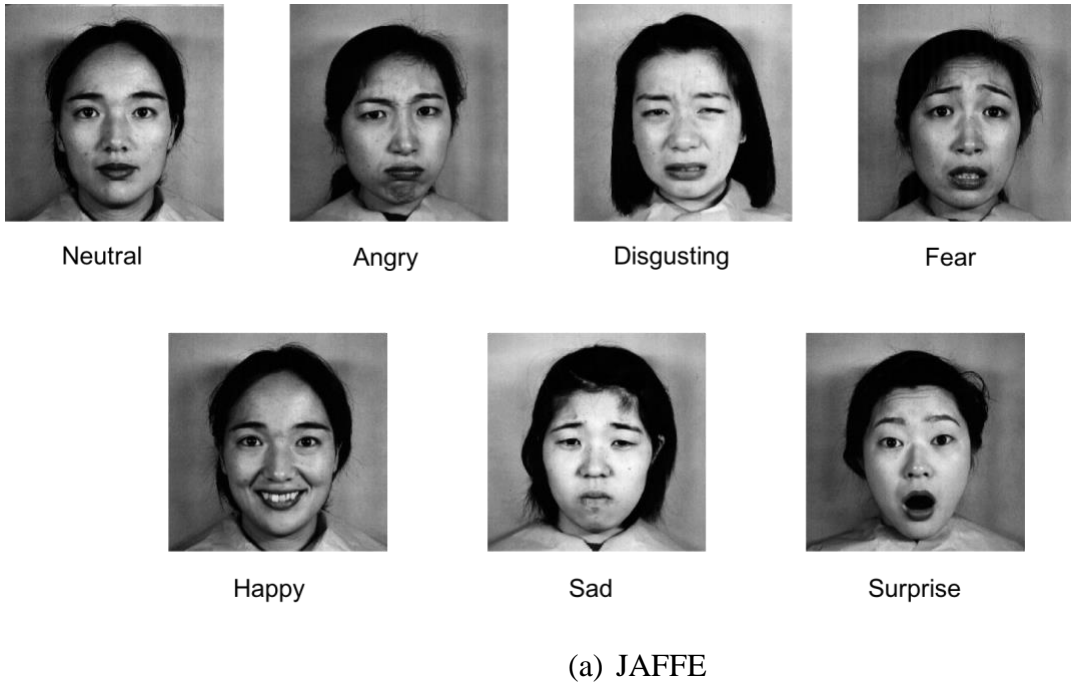
(h) Conv2d_8, 128 filters with size 3×3 : The last three convolutional layers seem to look for some high level features. From the above figures we can see clusters of dots and other shapes. That's also corresponding to section 5.1.2, the final convolutional layers are looking for features that can determine what expression this is.

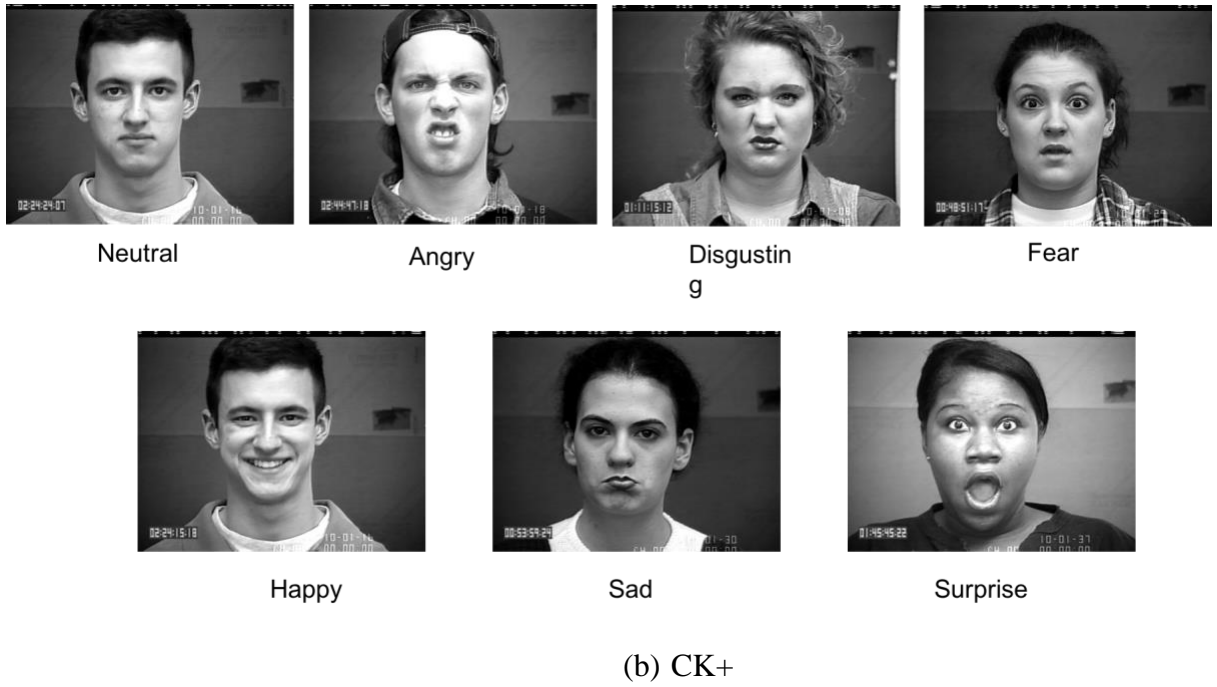
CHAPTER VI: RESULTS

6.1 Simulation

The current public face image datasets we choose to use in the simulation are JAFFE and CK+. We will test models trained on these two datasets separately, as well as the model trained using the combination of these two datasets, by which we can have an enlarged mix dataset. A sample set of photos with different expressions from JAFFE and CK+ are showed in Figure 6.1(a) and Figure 6.1(b) respectively.

Figure 6.1: Samples from dataset





Japanese Female Facial Expression(JAFFE)[1] database contains 213 images of 7 facial expressions (6 basic facial expressions + 1 neutral) posed by 10 Japanese female models. All the 213 images are used in our project. Extended Cohn- Kanade(CK+) [55] dataset includes 327 video sequences and each sequence labeled one of the seven expressions: anger, disgust, fear, happy, sad, surprise and contempt. In CK+, facial behavior of 210 adults was recorded. Participants were 18 to 50 years of age, 69% female, 81% Euro-American, 13% Afro-American, and 6% other groups. During examination of this dataset we notice that the distribution of different expressions is not even. Some participants only have one or two expressions recorded, while some may have ten expressions recorded. The first frame of each video sequence is neutral face, after a gradual change, last frame is the expression we can use for recognition. We only need six of these expressions plus neutral face to do classification. To construct our own dataset, we discard contempt expression and use the first frame of happy expression to

make a neutral face class. The total number of images we can use from CK+ dataset is 537.

Table 6.1: Confusion matrix for seven expressions on the JAFFE dataset

	Happy	Sad	Surprise	Anger	Fear	Disgust	Neutral
Happy	87.1%	5.6%	0.0%	4.0%	0.0%	0.0%	3.2%
Sad	2.4%	53.8%	0.0%	36.8%	0.0%	0.0%	8%
Surprise	0.0%	0.0%	87.5%	0.0%	3.6%	0.0%	4.5%
Anger	0.0%	3.4%	0.0%	96.6%	0.0%	0.0%	0.0%
Fear	0.0%	3.0%	0.0%	0.0%	77.6%	3.7%	15.7%
Disgust	0.0%	0.0%	0.0%	26.8%	0.0%	73.2%	0.0%
Neutral	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100%

75% of the two datasets will be trained by CNN. The proposed architecture is discussed in previous section. Then, 25% of the two datasets will be used for validation. Each dataset implements 5 runs. Table 6.1 shows the confusion matrix of JAFFE dataset. It can be seen that Happy, Surprise, Angry and Neutral have the recognition accuracy of 87.1% or higher. The expression, Fear and Disgust, has a lower recognition accuracy at 77.6% and 73.2% respectively. However, for expression Sad, the recognition accuracy is low at 53.8%. Many sad faces are recognized as angry. It may be because we knit our brows when we feel sad or angry.

Table 6.2: Confusion matrix for seven expressions on the CK+ dataset

	Happy	Sad	Surprise	Anger	Fear	Disgust	Neutral
Happy	96.3%	0.0%	0.0%	0.0%	0.0%	0.0%	3.7%
Sad	0.0%	100%	0.0%	0.0%	0.0%	0.0%	0.0%
Surprise	0.0%	0.0%	97.8%	0.0%	2.2%	0.0%	0.0%
Anger	0.0%	14%	1.2%	84.2%	0.0%	0.0%	0.6%
Fear	8.9%	0.0%	0.0%	0.0%	82.3%	3.8%	5.1%
Disgust	11.8%	0.0%	0.0%	23.5%	11.8%	52.9%	0.0%
Neutral	0.0%	1.5%	0.0%	5.6%	0.0%	0.0%	92.8%

Table 6.2 shows the confusion matrix of seven expressions for the CK+ dataset. It can be seen that all the expressions have recognition accuracy of 82.3% or higher, except for the disgust expression, which only has 52.9%. The expressions with recognition accuracy lower than 90% are those have smaller data under the specific categories. At the beginning of this section, we already declared that we reconstruct the original CK+ dataset by our own needs. It turns out that we can only use 32 disgusting expressions. It is less than one-tenth of the whole dataset, that is, smaller dataset results in lower accuracy. Another reason we think is, like Zavaschi et al.[53] proposed, the images of same subject are not in the training and validation groups at same time, this will result in a lower recognition accuracy compared with the result when the images of same subject are in both in the training and testing groups at same time. For CK+ dataset, it is not like JAFFE, we cannot guarantee that each subject has all seven expressions. Furthermore, we separate the dataset to training and testing parts randomly, so that it may be possible that one subject only appears in training or testing group.

Table 6.3: Confusion matrix for seven expressions on the combined dataset of CK+ and JAFFE

	Happy	Sad	Surprise	Anger	Fear	Disgust	Neutral
Happy	97.0%	0.0%	0.0%	0.0%	0.0%	0.0%	3.0%
Sad	2.6%	52.8%	0.0%	19.7%	1.3%	1.3%	22.3%
Surprise	0.8%	0.0%	93.4%	0.0%	0.0%	0.0%	5.8%
Anger	1.3%	5.7%	1.3%	86.3%	0.0%	0.0%	5.3%
Fear	22.9%	2.1%	0.7%	6.4%	51.4%	2.1%	12.9%
Disgust	15.3%	6.3%	0.0%	18.0%	6.3%	50.5%	0.9%
Neutral	1.5%	0.7%	0.0%	3.0%	0.4%	0.0%	94.4%

In Table 6.3, we combine the two datasets into one in order to have larger dataset and wider representation of different races. The recognition results for Sad, Fear, Disgust are just above 50%. It indicates that for a different person, the Sad, Fear and Disgust can act differently. Note that there is no standard expression for every person. The person labeled the dataset sometimes has to face the difficulty that a Sad face looks very similar to a Disgusted face. In this situation, a label Sad or Disgust has to be given without high confidence (i.e for example 100 persons label a face. 50 persons label Sad. 50 persons label Disgust). Thus, it gives the computer a very hard time to distinguish Sadness, Fear, and Disgust.

Table 6.4: Confusion matrix for five expressions on the CK+ dataset





	Happy	Sad	Surprise	Anger	Neutral
Happy	92.9%	0.0%	0.0%	0.0%	7.1%
Sad	0.0%	86.4%	0.0%	2.4%	11.2%
Surprise	0.0%	0.0%	100%	0.0%	0.0%
Anger	0.0%	8.0%	0.0%	74.1%	17.8%
Neutral	0.9%	0.0%	0.9%	0.0%	98.2%

In order to minimize the uncertainty above, we merge some Fear and Disgust expressions to angry with high confidence in CK+ only and remove the rest from Fear and Disgust. Table 6.4 shows the recognition results with the 5 expressions. It shows that the Angry has 74.1% accuracy and others have accuracy 86.4% or higher.

6.2 Implementation

In this section, we use the trained CNN model from dataset of CK+ and choose 5 expressions like in Table 6.5. We use our eCamera in our lab. Four subjects in lab are tested. We have collected data within 92 days. Each day, some of the five face expressions are captured from the four subjects. The experiment results are given in Table 6.5. It can be seen that for each subject the accuracies of different expressions are various. For all four subjects, Happy, Surprise, Neutral have higher recognition accuracy than Sad and Anger. The average expression recognition accuracy for four subjects is 93.2%.

Table 6.5: Experimental Results

	Person1	Person2	Person3	Person4
Happy	95.1%	96.3%	97.5%	92.8%
Sad	91.2%	95.2%	87.3%	85.4%
Surprise	98.3%	99.0%	97.6%	97.5%
Anger	89.1%	91.2%	85.1%	84.5%
Neutral	96.7%	93.3%	93.6%	97.7%
Test				

CHAPTER VII: CONCLUSION

In this paper, we proposed a Raspberry Pi and Pi camera based facial expression recognition system - eCamera by using a fully connected convolutional neural network and other state-of-the-art machine learning techniques. The experiment was performed with deep learning technique, trained with moderate public datasets. The deep CNN architecture was composed of 40 layers (exclude input and output layers). The simulation results are conducted on the two popular datasets, JAFFE and CK+. The real-world experiment is also provided. 4 subjects are tested within 92 days. The results showed that our eCamera system can have high face recognition accuracy as shown in Chapter VI. In the future, we will implement our eCamera with kids toys and to analyze the psychological effects of kids based on facial expression.

During our image pre-processing, Haar-Cascade classifier has been used for face detection. The limitation of Haar-Cascade classifier we used is that it can only detect frontal faces. Future work needs to be focused on non-frontal facial images with different illumination conditions as in real-time processing these global conditions are not uniform.

REFERENCES

1. Lyons, M., et al. Coding facial expressions with gabor wavelets. in Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on. 1998. IEEE.
2. Song, I., H.-J. Kim, and P.B. Jeon. Deep learning for real-time robust facial expression recognition on a smartphone. in Consumer Electronics (ICCE), 2014 IEEE International Conference on. 2014. IEEE.
3. Wu, L., et al. Robot-assisted intelligent emergency system for individual elderly independent living. in Global Humanitarian Technology Conference (GHTC), 2016. 2016. IEEE.
4. Ekman, P. and E.L. Rosenberg, What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS). 1997: Oxford University Press, USA.
5. Voulodimos, A., et al., Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, 2018. **2018**: p. 13.
6. Chen, M., et al., A 5G cognitive system for healthcare. Big Data and Cognitive Computing, 2017. **1**(1): p. 2.
7. Crockett, K., A. Latham, and N. Whitton, On predicting learning styles in conversational intelligent tutoring systems using fuzzy decision trees. International Journal of Human-Computer Studies, 2017. **97**: p. 98-115.
8. Hu, S. and G. Zheng, Driver drowsiness detection with eyelid related parameters by Support Vector Machine. Expert Systems with Applications, 2009. **36**(4): p. 7651-7658.

9. Gilda, S., et al. Smart music player integrating facial emotion recognition and music mood recommendation. in *Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2017 International Conference on. 2017. IEEE.
10. Marinchak, C.L.M., E. Forrest, and B. Hoanca, The Impact of Artificial Intelligence and Virtual Personal Assistants on Marketing, in *Encyclopedia of Information Science and Technology*, Fourth Edition. 2018, IGI Global. p. 5748-5756.
11. Saini, V. and R. Saini, Driver drowsiness detection system and techniques: a review. *International Journal of Computer Science and Information Technologies*, 2014. **5**(3): p. 4245-4249.
12. Jack, R.E., R. Caldara, and P.G. Schyns, Internal representations reveal cultural diversity in expectations of facial expressions of emotion. *Journal of Experimental Psychology: General*, 2012. **141**(1): p. 19-25.
13. Martinez, A., C. Benitez-Quiroz, and R. Srinivasan, Facial Color Is an Efficient Mechanism to Visually Transmit Emotion. *Journal of Vision*, 2018. **18**(10): p. 192-192.
14. Viola, P. and M. Jones. Rapid object detection using a boosted cascade of simple features. in *Computer Vision and Pattern Recognition*, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. 2001. IEEE.
15. Zhao-Yi, P., Z. Yan-Hui, and Z. Yu. Real-time facial expression recognition based on adaptive canny operator edge detection. in *Multimedia and Information Technology (MMIT)*, 2010 Second International Conference on. 2010. IEEE.
16. Wilson, P.I. and J. Fernandez, Facial feature detection using Haar classifiers. *Journal of Computing Sciences in Colleges*, 2006. **21**(4): p. 127-133.

17. Ghimire, D. and J. Lee, Geometric feature-based facial expression recognition in image sequences using multi-class adaboost and support vector machines. *Sensors*, 2013. **13**(6): p. 7714-7734.
18. Geetha, A., et al., Facial expression recognition—A real time approach. *Expert Systems with Applications*, 2009. **36**(1): p. 303-308.
19. Chellappa, R., C.L. Wilson, and S. Sirohey, Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 1995. **83**(5): p. 705-741.
20. Rowley, H.A., S. Baluja, and T. Kanade, Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 1998. **20**(1): p. 23-38.
21. Colmenarez, A.J. and T.S. Huang. Face detection with information-based maximum discrimination. in *Computer Vision and Pattern Recognition*, 1997. *Proceedings.*, 1997 IEEE Computer Society Conference on. 1997. IEEE.
22. Viola, P. and M.J. Jones, Robust real-time face detection. *International journal of computer vision*, 2004. **57**(2): p. 137-154.
23. Essa, I.A. and A.P. Pentland, Coding, analysis, interpretation, and recognition of facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 1997. **19**(7): p. 757-763.
24. Turk, M. and A. Pentland, Eigenfaces for recognition. *Journal of cognitive neuroscience*, 1991. **3**(1): p. 71-86.
25. Steffens, J., E. Elagin, and H. Neven. Personspotter-fast and robust system for human detection, tracking and recognition. in *Automatic Face and Gesture Recognition*, 1998. *Proceedings. Third IEEE International Conference on*. 1998. IEEE.
26. Littlewort, G., et al., Fully automatic coding of basic expressions from video. University of California, San Diego, San Diego, CA, 2002. **92093**.

27. Cohn, J.F., et al. Feature-point tracking by optical flow discriminates subtle differences in facial expression. in *fg*. 1998. IEEE.
28. Cootes, T.F., G.J. Edwards, and C.J. Taylor, Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2001(6): p. 681-685.
29. Ko, K.-E. and K.-B. Sim. Development of a Facial Emotion Recognition Method based on combining AAM with DBN. in *Cyberworlds (CW), 2010 International Conference on*. 2010. IEEE.
30. Wang, L., et al., Feature representation for facial expression recognition based on FACS and LBP. *International Journal of Automation and Computing*, 2014. **11**(5): p. 459-468.
31. Haykin, S.S., et al., *Neural networks and learning machines*. Vol. 3. 2009: Pearson Upper Saddle River, NJ, USA:.
32. Uddin, M.Z., J. Lee, and T.-S. Kim, An enhanced independent component-based human facial expression recognition from video. *IEEE Transactions on Consumer Electronics*, 2009. **55**(4).
33. Cortes, C. and V. Vapnik, Support-vector networks. *Machine learning*, 1995. **20**(3): p. 273-297.
34. Oliver, N.M., B. Rosario, and A.P. Pentland, A Bayesian computer vision system for modeling human interactions. *IEEE transactions on pattern analysis and machine intelligence*, 2000. **22**(8): p. 831-843.
35. Mollahosseini, A., D. Chan, and M.H. Mahoor. Going deeper in facial expression recognition using deep neural networks. in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. 2016. IEEE.
36. Goodfellow, I., et al., *Deep learning*. Vol. 1. 2016: MIT press Cambridge.

37. Gudi, A., Recognizing semantic features in faces using deep learning. arXiv preprint arXiv:1512.00743, 2015.
38. Tang, Y., Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.
39. LeCun, Y., et al. Handwritten digit recognition with a back-propagation network. in Advances in neural information processing systems. 1990.
40. Krizhevsky, A., I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. in Advances in neural information processing systems. 2012.
41. Szegedy, C., et al. Going deeper with convolutions. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
42. Simonyan, K. and A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
43. He, K., et al. Deep residual learning for image recognition. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
44. Srivastava, N., et al., Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014. **15**(1): p. 1929-1958.
45. Ioffe, S. and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.
46. Abadi, M., et al. Tensorflow: a system for large-scale machine learning. in OSDI. 2016.
47. Bastien, F., et al., Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590, 2012.
48. Jia, Y., et al. Caffe: Convolutional architecture for fast feature embedding. in Proceedings of the 22nd ACM international conference on Multimedia. 2014. ACM.

49. Solem, J.E., Programming Computer Vision with Python: Tools and algorithms for analyzing images. 2012: " O'Reilly Media, Inc."
50. Sagonas, C., et al. 300 faces in-the-wild challenge: The first facial landmark localization challenge. in Proceedings of the IEEE International Conference on Computer Vision Workshops. 2013.
51. Liu, P., et al. Facial expression recognition via a boosted deep belief network. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.
52. Scherer, D., A. Müller, and S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in Artificial Neural Networks–ICANN 2010. 2010, Springer. p. 92-101.
53. Zavaschi, T.H., et al., Fusion of feature sets and classifiers for facial expression recognition. Expert Systems with Applications, 2013. **40**(2): p. 646-655.
54. Wang, H., et al. eCamera: A Real-time Facial Expression Recognition System. in Proceedings of the 2018 International Conference on Artificial Intelligence. 2018.