



Real-Time Emotion Detection

Project No.: SCE17-0603

Name	:	Koh Nguan Theng, Melvyn
Matriculation No.	:	U1521083K
Supervisor	:	Dr Althea Liang Qianhui
Examiner	:	Assoc Prof Deepu Rajan

A Final Year Project in the Partial Fulfilment of the Requirements for the
Degree of Bachelor of Computer Engineering

School of Computer Science and Engineering

Academic Year 2018/2019

Abstract

Deep learning dominates the field of computer vision in recent years and in every few weeks a new deep learning technology takes over the other. Herein, convolutional neural network (CNN) is applied in this project.

Detecting facial expressions have been a very fast-growing topic in the field of computer vision as facial expressions are seen as a significant role in human communication and behavioural analysis. Ever since Paul Ekman devised the Facial Action Coding System (FACS) to detect a human facial feature and model the facial behaviours, many scientists are inspired to conduct psychological research on detecting real emotions of a person. Therefore, this has in turn inspired computer scientists to conduct tremendous active research in this field – finding the most accurate and fast models to detecting the true emotion of a person with a camera. This involves using Extended Cohn-Kanade (CK+) and FER2013 datasets.

This project aims to build a Real-Time Emotion Detection application that detects seven emotions namely – Anger, Disgust, Fear, Happy, Sad, Surprise and Neutral.

The software application is written in Python programming language with OpenCV for processing images and videos. CNN-based approach is done with Google's Tensorflow machine-learning library to construct the trained model. Lastly, Keras is used as the high-level neural networks API (application programming interface) that runs on top of Tensorflow. The model is trained and evaluated on the FER2013 and CK+ datasets.

Test subjects watching a series of videos is an experiment to evaluate on the performance of the software application. Each image frame is recorded and saved into a file in real-time as the subject is watching the video. The filename of each image is labelled with the emotion captured and determined by the application as well as the time and date when the image frame was taken.

Acknowledgements

Firstly, I would like to show my greatest appreciation to my supervisor, Professor Althea Liang Qianhui, for her thoughtful guidance and professional advice throughout the development of my Final Year Project. Her guidance and advice are not limited to the topic of my project but also the enhancement of my soft skills as well as presentation techniques. The amount of time spent on clearing my doubts and carefully observe my progress in this project. It was undoubtedly a pleasing and enriching journey and I am grateful to have had this experience.

Secondly, I would like to thank all the participants who took their time off amidst their busy schedules to take part in the experiments held by me. The outcome of this project would not be possible without the sincere help.

Lastly, I would like to thank my family members mates for their moral support and encouragement that they have been providing all these while and for not giving up their support on me despite so many failures. Without their sincere support, this project would not be possibly completed.

Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CK+ Database	Cohn-Kanade+ Database
CNN	Convolutional Neural Network
FER2013	Image dataset of 35,888 samples provided by Kaggle
FPS	Frames Per Second
GUI	Graphical User Interface
LBP	Local Binary Patterns
ReLU	Rectified Linear Unit
TPR	True Positive Rate
FPR	False Positive Rate

Table of Contents

Abstract	1
Acknowledgements	2
Abbreviations	3
List of Figures	6
Chapter 1: Introduction.....	7
1.1. Background.....	7
1.2. Objective.....	7
1.3. Scope.....	8
1.4. Report Organization.....	8
Chapter 2: Literature Review	9
2.1. Tensorflow.....	9
2.1.1. Introduction to Tensorflow.....	9
2.1.2. Tensors	9
2.1.3. Convolutional Neural Networks (CNNs).....	10
2.1.4. Batch Size and Epoch.....	11
2.1.5. Activation Functions	12
2.1.6. Gradient Descent	12
2.1.7. MobileNet.....	13
2.2. Research on Face Detection.....	14
2.2.1. HAAR Classifier	14
2.2.2. LBP Cascade Classifier	15
2.3. Keras with Tensorflow as Backend	16
Chapter 3: Software Tools Used	17
3.1. Programming Language and IDE	17
3.2. GUI Tools	17
Chapter 4: Preparing Datasets.....	18
4.1. Using Google Images.....	18
4.1.1. JavaScript snippets on the console in Google Chrome browser	18
4.1.2. Download the Google Images with a Python script.....	18
4.1.3. Check if the files are OpenCV-friendly	18
4.1.4. Remove unwanted images from our training datasets	19
4.2. FER2013 Dataset	19
4.2.1. Introduction	19
4.2.2. Dataset Specifications	19
4.2.3. Preprocess the Dataset.....	21

4.3. Conclusion	21
Chapter 5: Implementation	22
5.1. Deep Learning Procedures.....	22
5.1.1. Constructing CNN with Keras	22
5.1.2. Training the Network	24
5.1.3. Validations.....	24
5.2. Emotion Detection	26
5.2.1. Video Stream/ Real-Time Detection	27
5.2.3. Upload Static Image for Detection.....	28
Chapter 6: Experimentations	29
6.1. Experiment Setup.....	29
6.2. Storing of image frames captured in real-time	29
6.3. Results from Experiments.....	31
6.3.1. Video Stream/ Real-Time.....	31
6.3.2. Static Images	32
Chapter 7: Conclusion and Future Works.....	33
7.1. Conclusion	33
7.2. Future Works	34
References.....	35
Appendix A: Preprocessing of Dataset	37
Appendix B: Construction of CNN Structure.....	38
Appendix C: Main Script for Emotion Detection.....	39
Real-Time Video Capture	39
Upload Static Image	40
Appendix D: Results from Experiments	41
Angry.....	41
Disgust.....	42
Fear.....	42
Happy	43
Sad	43
Surprise.....	44
Neutral	44

List of Figures

Figure 1 – From source: [14], Visual Representation of levels of abstraction.....	11
Figure 2 - Example of Max Pooling	11
Figure 3 – From source: [11], ReLU Test Accuracy against Number of Iterations.....	12
Figure 4 – Gradient Descent Example.....	13
Figure 5 – From source: [6], HAAR Classifier	14
Figure 6 – Face Detection (Frontal)	15
Figure 7 – Face Detection (Either Side)	15
Figure 8 – From source: [9], LBP Cascade Classifier.....	15
Figure 9 – From source: [14], Overview of CNN Structure Workflow	16
Figure 10 – GUI of the Emotion Detection Program	17
Figure 11 – Some Sample Images from FER2013.....	20
Figure 12 – Last 5 Rows of FER2013	21
Figure 13 – High-level Workflow of Emotion Detection	22
Figure 14 – Summary of Steps Taken	23
Figure 15 - First Convolutional Layer.....	23
Figure 16 – From source: [14], A (2, 2) Stride.....	23
Figure 17 – 2nd and 3rd Convolutional Layer	24
Figure 18 – Confusion Matrix of the Validations	25
Figure 19 – True Positive Rate Equation	25
Figure 20 – Workflow of the Python Program.....	26
Figure 21 – Folder Name Created	27
Figure 22 – Example of Filename	27
Figure 23 – Bounding Boxes and Labels	28
Figure 24 – Origin and Dimension of Bounding Box	28
Figure 25 - Experimental Setup.....	29
Figure 26 – An Example of 4366 Frames Captured.....	30
Figure 27 - Emotion Detection (Real-Time)	31
Figure 28 – 4 Faces from Static Image.....	32
Figure 29 – 20 Faces from Static Image.....	32

Chapter 1: Introduction

1.1. Background

The study of facial expression dates back to the 5th century when humans were using physiognomy as a practice to determine a person's personality based on their facial appearance [1]. By the 20th century, it has advanced progressively and researchers like Paul Ekman had leveraged on the advancement of modern technologies that made a great impact in the development of emotion detection systems on a human [2]. His works defined facial cues to detect even subtle expressions like micro-expressions. Furthermore, his most famous work was to detect if an expression is real or fake based on the study of facial muscles, expression asymmetries and several other factors that determine a real expression.

His works inspired many researchers to build camera systems that detect emotions of a person in real-time that is extremely fast and highly accurate. The advancement of technology improves our daily lives and these systems are useful in real-life applications such as airport security checks, lie detector during job interviews and human computer interactions.

In this project, the focus is building a software application that detects the emotions of a person in real-time or from static images uploaded by the user of the application.

1.2. Objective

The objective of this project is to study some of the existing works that leveraged on open-source machine-learning techniques on facial expression recognition and implement the best method of real-time emotion detection in a software application.

1.3. Scope

To achieve a satisfactory result within the stipulated time-frame, the scope of this project is to classify six categories of emotions – anger, fear, happy, sad, surprise and neutral. The system classifies each image frame from the video into discrete emotion categories.

Test subjects are to watch videos that evoke certain emotions of Anger, Disgust, Fear, Happy, Sad, Surprise and Neutral. The findings from the experiments conducted would help to evaluate the accuracy of the software application that detects the aforementioned emotions in real-time.

1.4. Report Organization

Chapter 2: Literature Review

Related works and studies of the machine-library as well as the face detection techniques are discussed in Chapter 2.

Chapter 3: Software Tools Used

The software tools used in this project like the machine-learning libraries are specified in this chapter.

Chapter 4: Preparing Datasets

This chapter discusses how the datasets are preprocessed before passing them to the model for training and testing.

Chapter 5: Implementation

The implementations of the techniques and methods taken in this project.

Chapter 6: Experiments

The procedures taken during the experiments for the program evaluation.

Chapter 7: Conclusions and Future Works

The overall learning experience and lessons are wrapped up in this chapter and the possible future works following the end of this project will also be mentioned.

Chapter 2: Literature Review

2.1. Tensorflow

2.1.1. Introduction to Tensorflow

TensorFlow is a software framework for building and deploying large-scale machine learning models [7]. Machine learning is a different approach than traditional programming. With traditional programming, we write the program that tells the computer exactly what to do to complete the task. With machine learning, we don't explicitly tell the computer how to do something. Instead we show a training data and the machine learning algorithm uses the training data that come up with its own rules to complete the task.

Developed by Google, TensorFlow was designed to be a common platform for building machine learning applications internally. It is licensed under the Apache open source software license which means that anyone can use it, and even redistribute modified version of it. Furthermore, there is no licensing fees to use TensorFlow and it provides the basic building blocks that we need to design, train, and deploy machine learning models.

It is widely popular amongst several types of machine learning algorithm to build deep neural networks for areas like image recognition, speech recognition and language translation.

2.1.2. Tensors

Tensors are basically n-d (multi-dimensional) arrays and the name TensorFlow comes from the design of the system where it works with large data sets made up of many different individual attributes [7]. Any data that you want to process with TensorFlow has to be stored in the multi-dimensional array. To run operations on the data

set, we construct a computational graph similar to a flow chart that determines how data flows from one operation to the next.

For instance, if we pass in two tensors, adding those numbers, squaring them and then output of the result is another tensor. TensorFlow is designed to be very generic and open-ended. We can define the graph of operations that does any calculation that you want. While TensorFlow is most often used to build deep neural networks, it can be used to build nearly anything that involves processing data by running a series of mathematical operations.

Tensorflow executes a program in two phases:

- 1) Building the computational graph
- 2) Evaluating the computational graph (by creating a session `sess = tf.Session()`)

A computational graph is a series of Tensorflow operations arranged into a graph. The nodes of the graph represent Tensorflow operations and the edges represent values (tensors) that follow through the graph.

2.1.3. Convolutional Neural Networks (CNNs)

Different Neural Networks have their own specializations. For example, Recurrent Neural Networks (RNNs) are mainly used in natural language processing like speech recognition in YouTube videos. Whereas for Convolutional Neural Networks (CNNs), they are typically used in computer vision tasks like facial recognition and emotion detection. The initial layers of CNN are meant to extract distinct features from the image input.

A CNN is a form of an ANN (artificial neural network), but what differentiates it from just a multi-layer perceptron is that it is one of the most useful ANN [5]

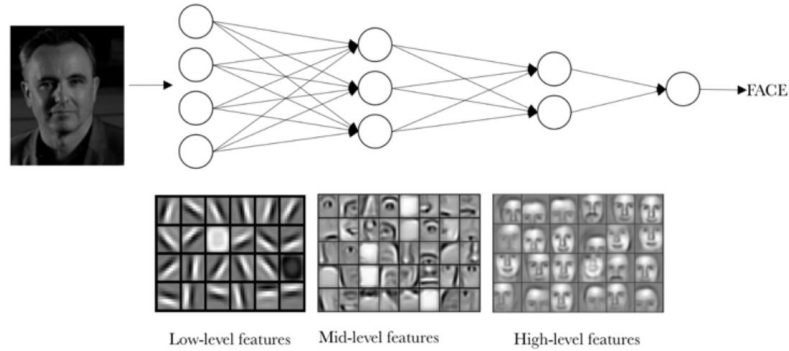


Figure 1 – From source: [14], Visual Representation of levels of abstraction

The idea of the figure above is that each layer in the CNN is learning different levels of abstraction. Different layers detect features or visual features in the images such as lines and edges.

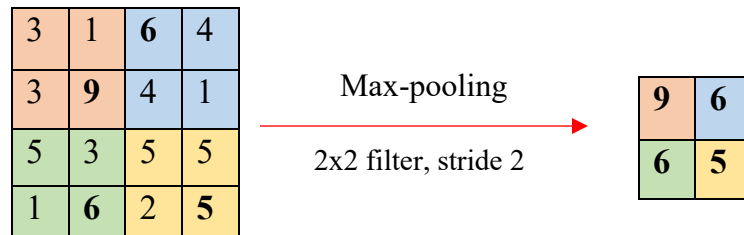


Figure 2 - Example of Max Pooling

In most cases, there will be a max pooling every time after each convolutional layer [10]. This layer simply reduces the number of pixels in the image. Using the above figure as an example, the 2x2 sized filter (respective colored square) replaces with only the highest value on that same colored square. Stride refers to the number of units the filter shifts each time, and if the stride is 2, the filter shifts 2 units at a time.

2.1.4. Batch Size and Epoch

As most datasets that are used in practice today are very large (>2000), it is not practical to feed the artificial neural network the entire dataset at once due to the hardware limitations of computers. A dataset is divided into a number of batches, which can also be seen as 'sets' [15].

The number of epochs is the number of times the dataset is passed forward and backward through the network. If the number of epochs is 1, it means that the entire dataset is fed to the network only once. As mentioned in the previous paragraph, we cannot pass all the data at once, so the number of epochs depends on how many times the dataset is divided into batches.

2.1.5. Activation Functions

There are many different types of activation functions available. However, there are only a few that are widely used – ReLU, tanH and sigmoid.

Table 1 – From source: [11], Test Accuracies for MNIST Dataset

Activation Function	Testing Accuracy
ReLu	0.9843
TanH	0.7591
eLu	0.9687
Sigmoid	0.7851
SoftPlus	0.9765
SoftSign	0.7773

Using the MNIST test set, the test accuracies for each activation function can be seen in Table 1. ReLU has the best performance with 98.43% classification accuracy in comparison to the other activation functions [11].

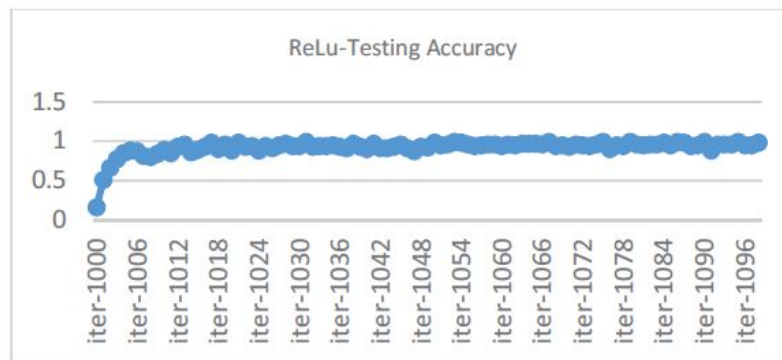


Figure 3 – From source: [11], ReLU Test Accuracy against Number of Iterations

The visualization graph in Figure 2 shows the plot of ReLU's test accuracies against the number of iterations. Iterations is the number of batches needed to complete one epoch.

2.1.6. Gradient Descent

Gradient descent is an iterative optimization algorithm to obtain the best result based on the minima of the curve, be it local or global minima. It is iterative as the algorithm runs in a loop until the most optimal result is obtained.

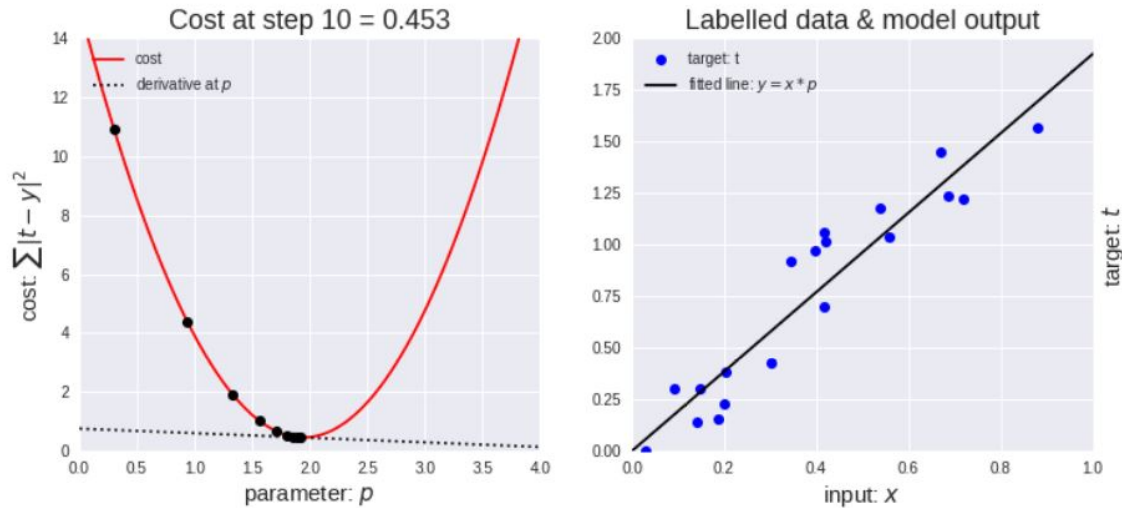


Figure 4 – From source: [15], Gradient Descent Example

In Figure 3, the graph on the left illustrates how the algorithm finds the minima of the curve until the rate of inclination equals to 0. This gives the best-fit line for the scatterplot on the right.

2.1.7. MobileNet

MobileNet is an architecture of convolutional neural network designed by researchers at Google. They are coined “mobile-first” in that they’re architected from the ground up to be resource-friendly and run quickly, right on your phone.

Advantages:

- 1) Very small and lightweight
- 2) Extremely fast
- 3) Remarkably accurate
- 4) Easy to use and many resources and documentations online

As one of the main goals of this project is to export it into a smartphone application, the deep learning tasks can be performed in the cloud (TensorFlow Cloud). To classify an image taken from our smartphone camera, that image is sent to the web service and it’s classified on a remote server before the result is sent back to the smartphone.

2.2. Research on Face Detection

Before I embark on the study of real-time emotion detection, the fundamentals of face detection should not be overlooked. There are various techniques on face detection that is widely available as open-source. In this project, I used the HAAR Cascade Default Classifier as the face classifier to detect any faces found in the given frame.

OpenCV with Python was chosen to be used in this project as it has the largest open-source machine learning library in Computer Vision. There are two main machine learning based approach, mainly *HAAR Classifier* and *LBP Cascade Classifier* [9].

2.2.1. HAAR Classifier

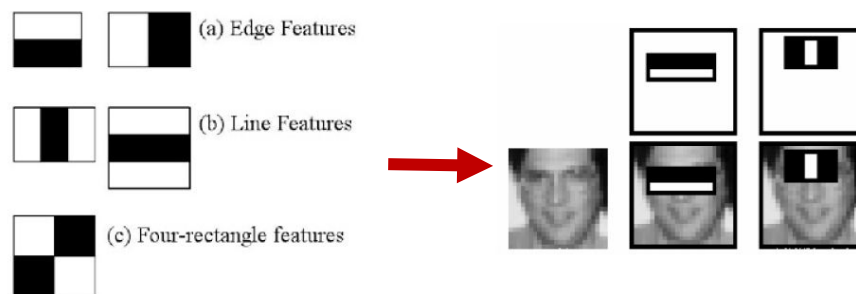


Figure 5 – From source: [6], HAAR Classifier

HAAR Classifier is an object detection method as proposed by Paul Viola and Michael Jones [4]. The steps taken by the classifier are systematic as follows:

1. Convert image to grayscale
2. Each window is placed over the image to calculate a single value (feature value):
 $feature_value = SOP_underBLACK - SOP_underWHITE$ SOP: sum of pixels
3. The areas that are neither darker or lighter than other regions are discarded
4. This process is called Adaboost, where features are selective to improve the classification accuracy

The figures in the following page show the implementation of the HAAR cascade face classifiers and the efficiency of the classifier is tested – frontal and side views of the subject's face. Even in side views, it still achieves an accuracy rate of more than 99%.

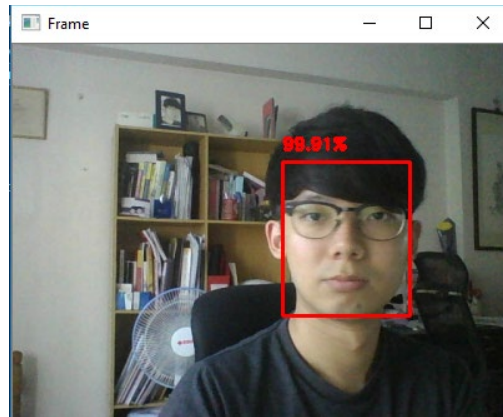


Figure 6 – Face Detection (Frontal)

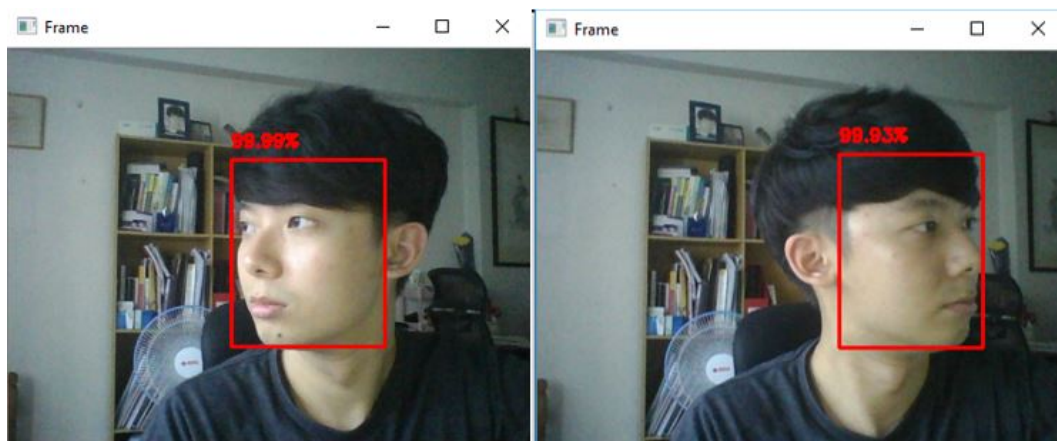


Figure 7 – Face Detection (Either Side)

2.2.2. LBP Cascade Classifier

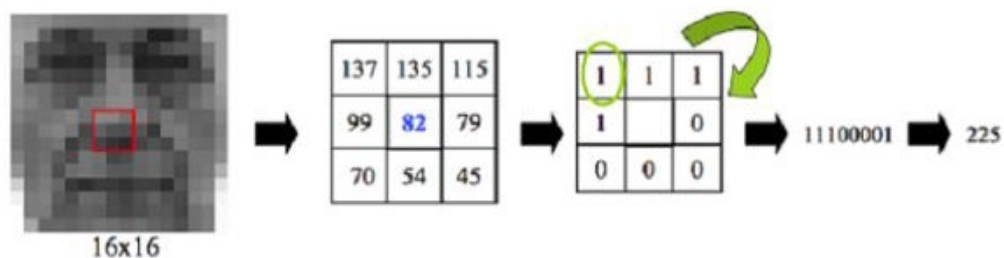


Figure 8 – From source: [9], LBP Cascade Classifier

Each image is divided into blocks of 3x3 pixels each. The central pixel compares with every neighboring pixel value. With each pixel value greater than or equal to the central pixel value, '1' is set to that pixel, and '0' otherwise. In each block, the binary values are then converted to decimal number as a representation of that block. Finally, it

concatenates these block histograms to form a one feature vector for one image, which contains all the features we are interested in.

2.3. Keras with Tensorflow as Backend

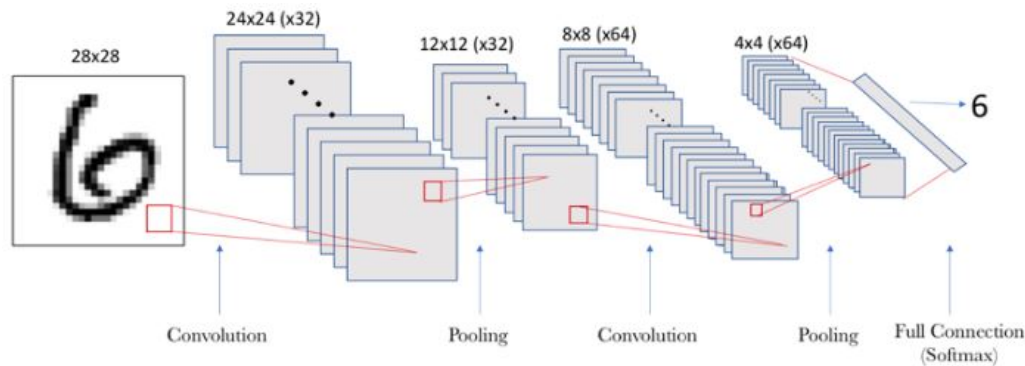


Figure 9 – From source: [14], Overview of CNN Structure Workflow

Keras serves as the high-level API for Tensorflow and it is seen as rapid prototyping a Tensorflow model with minimal lines of code. Even with Keras, a very complex neural network can be built within a few minutes by tweaking the parameters and the hyperparameters required in the Model and Sequential APIs.

Advantages of using Tensorflow with Keras:

1. Flexibility
2. Functionality
3. Threading and Queues

It has the flexibility of Tensorflow and it is as good as a low-level library, where the cost function, metric and the number of layers can be specified. Even though Tensorflow provides a wider range of mathematical operations, Keras does provide a lot of operations that is within the scope of this project, hence Keras as a machine-learning library is sufficient. Lastly, sessions are leveraged in Tensorflow while Keras leverages on threading and queues. They are both very similar as queues are powerful enough to compute the tensors asynchronously in a computational graph.

Chapter 3: Software Tools Used

3.1. Programming Language and IDE

Python is used as the programming language in this project. The deep learning artificial neural network was developed in Python using Visual Studio Code as the Integrate4d Development Environment. Visual Studio Code is fast and lightweight for Python machine learning. In this project, Keras is used as the high-level API to build and train the deep learning model for emotion detection. It uses Tensorflow as the backend as it runs on top of it. It is more user-friendly and easier to use as compared to Tensorflow.

3.2. GUI Tools

The Tk graphical user interface toolkit, or in short, Tkinter, is used in this project to create the GUI for human-computer interaction. Tk is a lightweight object-oriented layer around Tcl Developer Xchange. It is the standard GUI library for many dynamic programming languages that produce rich. The main advantage is that the native applications built are executable across many operating systems including Windows, Mac OS X and Linux. In this project, the Tk GUI is also coded in Python.

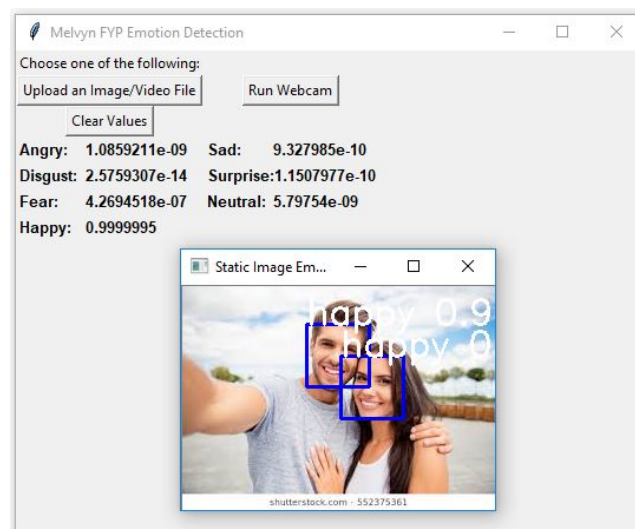


Figure 10 – GUI of the Emotion Detection Program

Chapter 4: Preparing Datasets

4.1. Using Google Images

4.1.1. JavaScript snippets on the console in Google Chrome browser

I experimented on using Google images as my training data for my machine learning model. Downloading the Google images for each expression one by one is too time consuming as the process involves a lot of overheads. Furthermore, for a strong machine learning model, it must have a strong training dataset. On the Google Image page, I search for the expression that I'm looking for and proceeded to JavaScript console and run JavaScript snippets that pulls down the jQuery JavaScript library. Next, the CSS selector is used to obtain the URLs of the images that have been loaded on that page:

```
var urls = $(' .rg_di .rg_meta').map(function() { return JSON.parse($(this).text()).ou; });
```

Lastly, write the URLs of the images to a text file.

4.1.2. Download the Google Images with a Python script

Using the 'requests' and 'cv2' libraries make this process a lot easier. Given the URLs of the images, the package 'requests' is responsible for downloading the images. Running the Python script requires 2 arguments: the path to the text file that contains all the URLs, and the path to the output images that are successfully downloaded from Google Images.

4.1.3. Check if the files are OpenCV-friendly

I made sure that the downloaded file can be opened by OpenCV. Basically, I looped through the downloaded files and try to open with OpenCV using the Try and Except in each iteration. If OpenCV fails to open, that image file will be removed. One of the many reasons why some images fail to be read by OpenCV is that some images are corrupted or an image file type that is not supported by OpenCV, for instance, the gif and svg format.

4.1.4. Remove unwanted images from our training datasets

Not every image from Google Image is what I was looking for. In fact, there are many irrelevant ones like emoticons, dogs and posters. The manual deletion of the unwanted images took out a huge amount of time.

4.2. FER2013 Dataset

4.2.1. Introduction

FER2013 is a dataset which was published on International Conference on Machine Learning (ICML) in the year 2013. It was also used in a data science competition hosted by Kaggle, the world's largest online community of data scientists and machine learners. It is a CSV file with each row containing the emotion label of each image followed by all the pixel values of that image, and lastly the usage type is either 'Training' or 'Testing'.

4.2.2. Dataset Specifications

In the FER2013 dataset, it consists of 48x48-pixel (2304 pixels values per sample) grayscale images of faces with a specific emotion that can be categorized as one of the seven emotions – Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral. There is a total of 35,888 different images, 28,709 samples are training set and the test set consists of 3,589 samples. The figure below shows some of the example images in the FER2013 dataset all in grayscale.

The emotion labels and the number of images corresponding to each emotion are as follows:

Table 2 – Number of Images for Each Emotion Label

Label	Emotion	No. of Images
0	Angry	4593
1	Disgust	547
2	Fear	5121
3	Happy	8989
4	Sad	6077
5	Surprise	4002
6	Neutral	6198



Figure 11 – Some Sample Images from FER2013

Figure 6 reveals some of the images that are contained in the dataset csv file. All the images are in grayscale, which implies that color channel depth of 1.

4.2.3. Preprocess the Dataset

	emotion		pixels	Usage
35882	6	50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...	PrivateTest	
35883	3	178 174 172 173 181 188 191 194 196 199 200 20...	PrivateTest	
35884	0	17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...	PrivateTest	
35885	3	30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...	PrivateTest	
35886	2	19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...	PrivateTest	

Figure 12 – Last 5 Rows of FER2013

Using the Pandas library for processing of data, the above figure shows below are the last 5 rows of the dataset.

Firstly, the elements in the ‘pixels’ column are converted into a list for each row by splitting the long string of pixel values by white space as a list.

Secondly, to save computational cost for building the machine-learning model, each image is normalized and resized.

Lastly, the dataset is split into 3 sets – training, testing and validation, which will be explained in the next chapter – implementation.

4.3. Conclusion

There are numerous challenges and difficulties when using stock images from Google search, and they are:

1. It is very time-consuming especially when a huge dataset (>3000) is needed for training and testing, and filtering those irrelevant ones like emoticons take up a lot of time.
2. Inconsistent image dimensions.
3. Different file formats.

Since FER2013 is open-source and it was used for data science competitions like in Kaggle, I decided to use FER2013 to feed my ANN and ultimately build a model.

Chapter 5: Implementation

5.1. Deep Learning Procedures

5.1.1. Constructing CNN with Keras

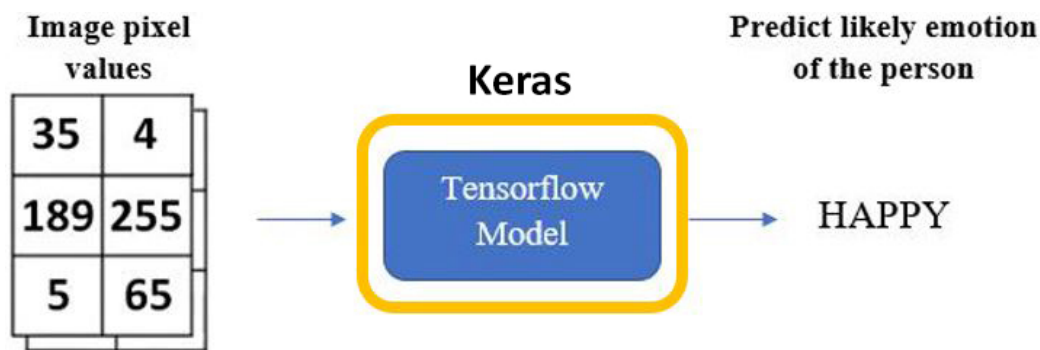


Figure 13 – High-level Workflow of Emotion Detection

Tensorflow is used as backend with Keras as the high-level API to build the CNN. The train set and test set are stored in their respective x_train , y_train , x_test , y_test arrays based on the categorical value in 'Usage' column in the FER2013.csv file. The 4 types of 'Usage' in FER2013 are – Training, Public Test and Private Test. x_train is the input while y_train is the target based on the corresponding x_train .

In this project, a 3-layer convolution network is created with rectified linear unit (ReLU) as the activation function for each hidden layer while SoftMax classifier is used at the final output layer. The reason for using SoftMax is that it normalizes outputs in the final layer in the scale of $[0, 1]$ as the sum of outputs will always be equal to 1 (largest probability equals 1). This is applied on the very last stage by converting the output of the convolution part of the CNN into a 1-dimension feature vector, and this operation is called flattening, which I will be implementing later on in this chapter.

The steps can be summarized in the figure below.

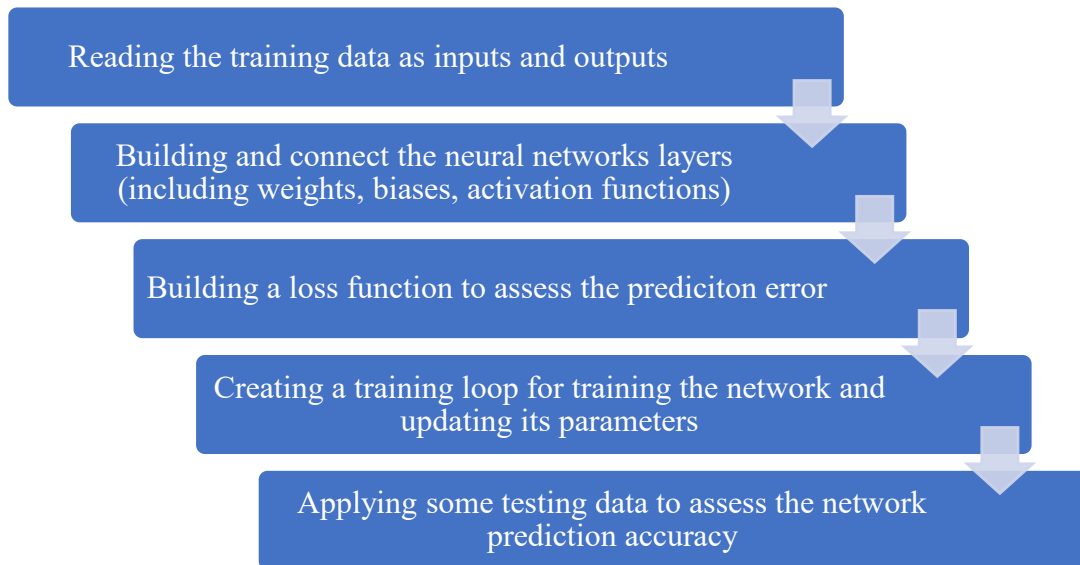


Figure 14 – Summary of Steps Taken

```
#1st convolution layer
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48,48,1))) #64 filters using a 5x5 window
model.add(MaxPooling2D(pool_size=(5,5), strides=(2, 2))) #5x5 window for pooling layer
```

Figure 15 - First Convolutional Layer

Since each image in the FER2013 dataset has a dimension of 48x48 pixels, the `input_shape` is set as (48,48,1) with '1' as the argument for the color channel depth of 1 as the images are all in grayscale. There are 64 filters using a 5x5 window for the convolutional layer and a 5x5 window size for the max pooling layer. The strides of (2,2) refers to the amount by which the 5x5 window shifts at a time, which can be illustrated in the figure below.

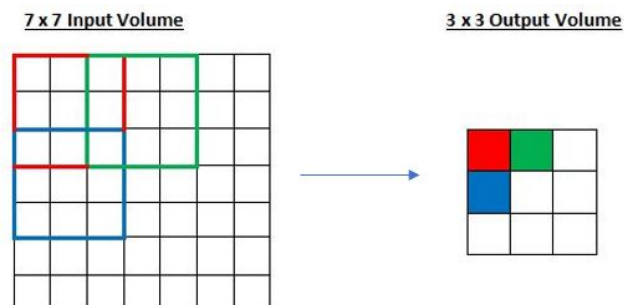


Figure 16 – From source: [14], A (2, 2) Stride


```

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

model.add(Flatten()) #flatten the 3D tensor to 1D tensor before applying SoftMax

```

Figure 17 – 2nd and 3rd Convolutional Layer

In the 2nd and 3rd layer, the window size for the filters is decreased to 3x3 and the Average Pooling is used instead of Max Pooling. Because of the loss in pixels after each pooling, the number of filters is increased, and in this case to 128.

Lastly, after the construction of the CNN structure, the model must be flattened from a 3-D to a 1-D tensor before the SoftMax classifier is applied at the output layer. A single long 1-D feature vector is used by the dense layer for the final classification of the emotion label.

5.1.2. Training the Network

Using Keras's out-of-the-box *fit_generator* function, training samples in the training set are randomly selected to implement the training of the network. As categorizing of emotions is a form of multi-class classification, SoftMax cross-entropy is used as loss function in the output layer. The cross-entropy function correlates between output probabilities and one hot encoded label.

5.1.3. Validations

Validation is the approach to using the entire dataset available to select the model and estimate the error rate. The validation uses a part of the dataset, which is the test set, to select the model, which is known as the validation test.

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	214	9	53	30	67	8	86
Disgust	10	24	9	2	6	0	5
Fear	45	2	208	29	89	45	78
Happy	24	0	40	696	37	18	80
Sad	65	3	83	56	285	10	151
Surprise	7	1	42	27	9	303	26
Neutral	45	2	68	65	88	8	331

Figure 18 – Confusion Matrix of the Validations

The figure shows the confusion matrix of the model that is tested on the x_{test} test set. The diagonal values across the table, which are highlighted in yellow, are the true positives while the rest are false positives. An ideal result is a 100% true positive rate where only the diagonal values are non-zero and the rest are zero.

To calculate the True Positive Rate (TPR) based on the confusion matrix,

$$TPR = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

Figure 19 – True Positive Rate Equation

Table 3 – Emotion Label with its TPR and Overall Average TPR

Emotion	True Positive Rate
Angry	214/410 = 52.1%
Disgust	24/41 = 58.5%
Fear	208/503 = 41.3%
Happy	696/905 = 76.9%
Sad	285/581 = 49.1%
Surprise	303/392 = 77.3%
Neutral	331/757 = 43.7%
Average = 57%	

Based on the result of each emotion label, the average is calculated by adding all the true positive rates and dividing by the total number of emotion labels, which is 7. The average accuracy rate is around 57%.

5.2. Emotion Detection

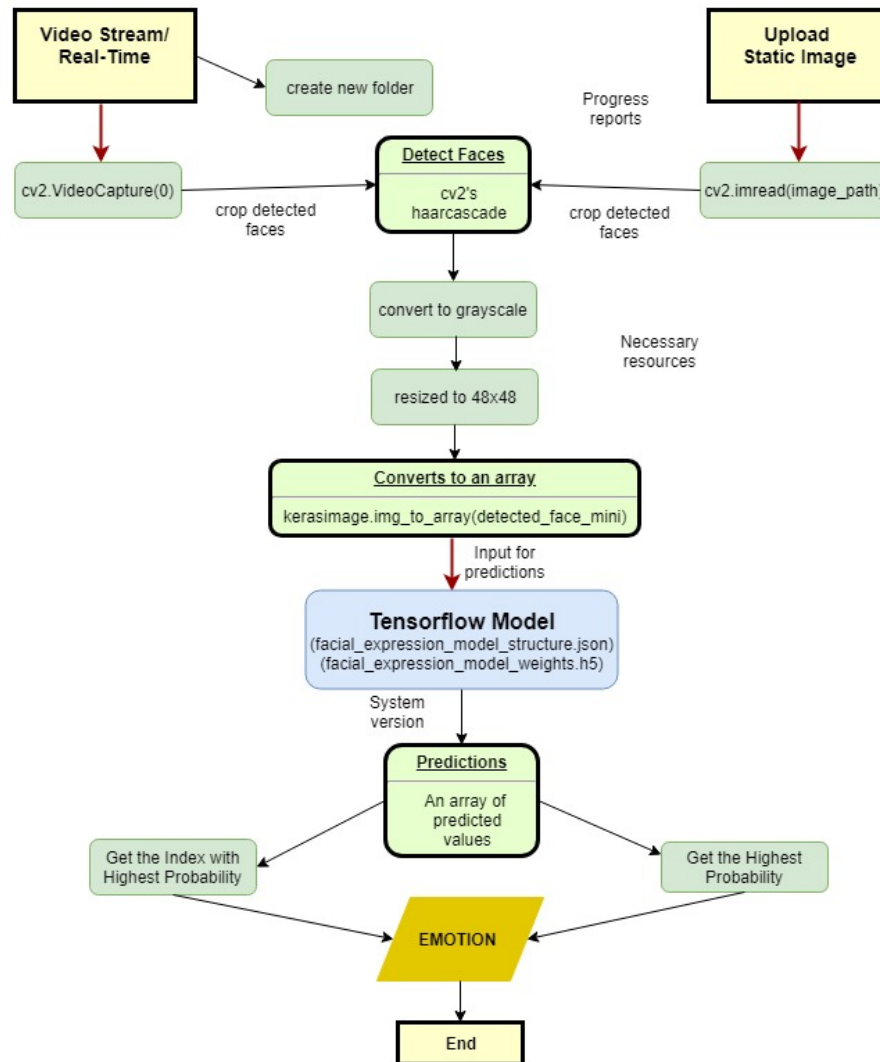


Figure 20 – Workflow of the Python Program

There are basically 2 types of detection that I have implemented in the GUI with Tkinter. The first detection is to have a real-time detection where the test subject must be present in front of the camera (in our case it is the webcam). The second method of detection is to upload an image file of JPEG format for the emotion detection.

5.2.1. Video Stream/ Real-Time Detection

Using OpenCV library's *VideoCapture()* function and with '0' as the argument, it runs the laptop's built-in webcam by default and a cv2 window will pop-up for live stream. A new folder is created, and the folder name is based on the current date and time.

frames_captured_20180917-174000	17/9/2018 5:40 PM	File folder
frames_captured_20180917-135429	17/9/2018 1:54 PM	File folder
frames_captured_20180927-150901	27/9/2018 3:09 PM	File folder

Figure 21 – Folder Name Created

For instance, in the figure above, “*frames_captured_20180917-135429*” is the folder name created that contains all the frames captured on 9 September 2018 at 1354 hours.

After the folder is created, the cv2 window tries to detect any faces using the face cascade classifier with *haarcascade_frontalface_default* xml file. When a face is detected, cv2 crops from the frame that contains the face and transform it to grayscale. After the transformation, it is resized to a 48x48 image dimension. Lastly, using Keras, convert all the pixel values as a single array of size 2304 ($48 \times 48 = 2304$), and then feed this array as an input data to the Tensorflow model that I have created earlier on.

The model then predicts the emotion based on the face image pixel values. The emotion label with the highest probability is determined as the emotion predicted. At the same time, the image file is saved inside the folder that was created earlier on, with the frame number, emotion label and probability of that emotion saved as the file name.



Figure 22 – Example of Filename

Figure 4 shows an example of how each image file is stored with the aforementioned details saved as the file name.

5.2.3. Upload Static Image for Detection

For this method of detection, it has a simpler approach as compared to real-time detection. It is similar to real-time detection except that it uses OpenCV's *imread()* function to read in the image file. The other difference is that it does not have to go through a loop to continuously capturing frames by frames in real-time as it only has one frame for detection.

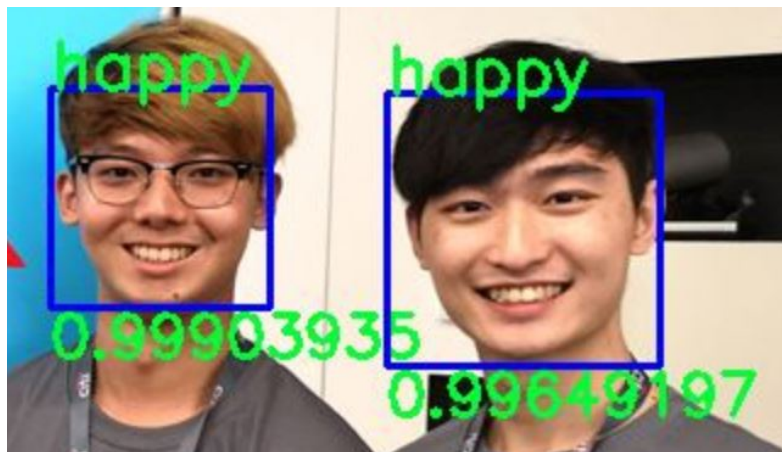


Figure 23 – Bounding Boxes and Labels

Figure 22 illustrates how the bounding boxes are drawn on every face that is detected by the HAAR Cascade classifier. The emotion labels are printed on top of the bounding box while the probability or the confidence level of the emotion predicted is printed below the bounding box. The location of the printing has to be in reference with the (x,y) origin of the bounding box where the top left corner is the origin (e.g. (0,0)).

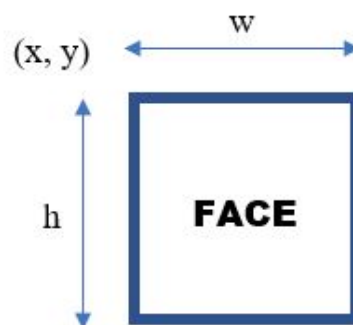


Figure 24 – Origin and Dimension of Bounding Box

Chapter 6: Experimentations

6.1. Experiment Setup

Test subjects are required to watch a series of videos which are aimed to evoke certain emotions to evaluate the performance of this application. A camera is placed at 70 centimetres from the face of the subjects and the results are recorded. The experimental setup is as shown in Figure 1.

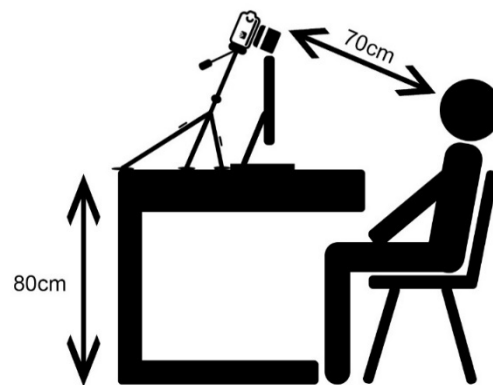


Figure 25 - Experimental Setup

Each subject received the same instruction and their facial expressions were recorded on camera. The recorded files were analysed in detail and interpreted in relation to the emotionally neutral face for each subject.

6.2. Storing of image frames captured in real-time

The image frames captured by the video camera are analysed by the model and stored in a specific folder in real-time.

The emotion label with its probability value determined by the CNN model will be stored as the file name of each corresponding image frame. Additionally, the frame number and time and date will also be recorded as the name of the folder. The figure below shows an example of image frames captured and stored as images files in real-time.

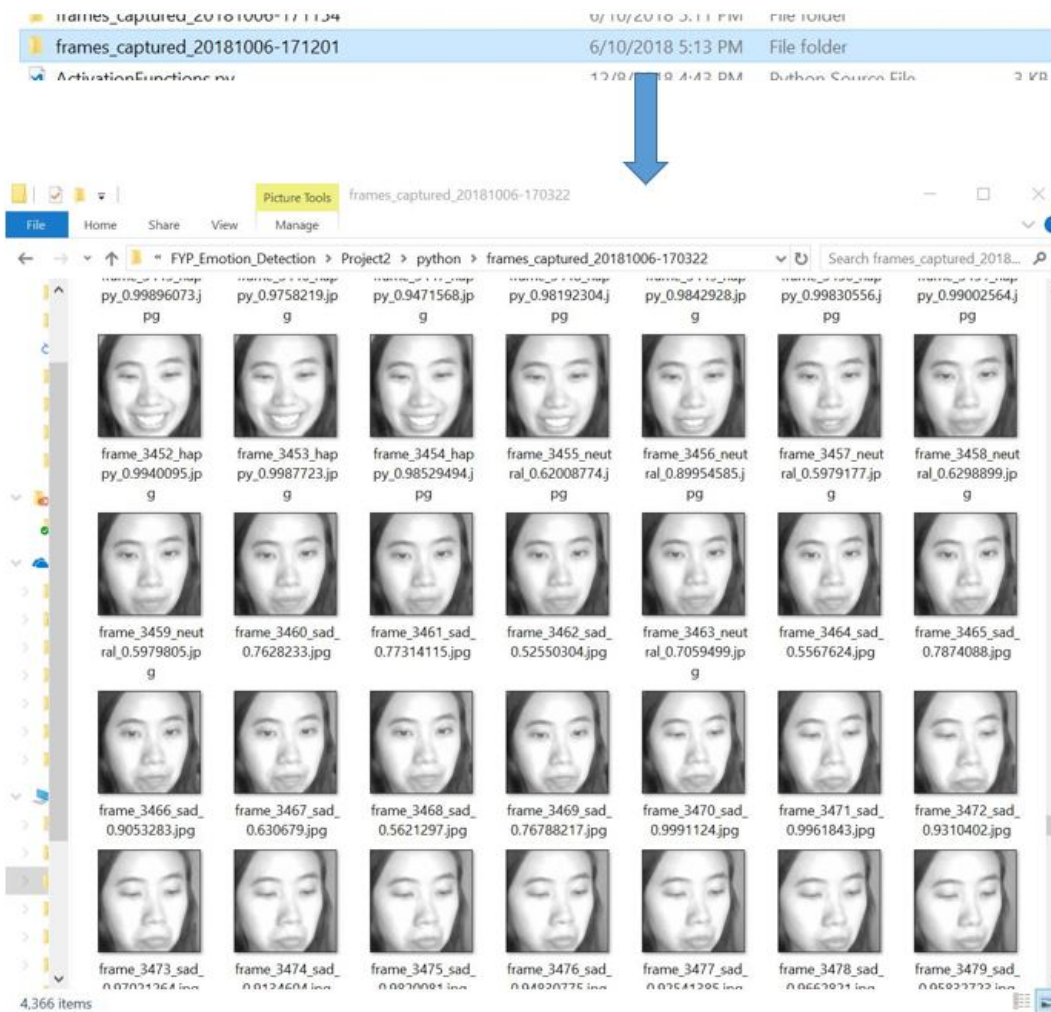


Figure 26 – An Example of 4366 Frames Captured

Figure 25 shows the frames captured in real-time and a total of 4366 image frames are saved in this folder with the date and time specified as the folder name. This process involves a lot of overheads as every face detected and classified with an emotion is captured and cropped in the frame, and lastly, save that cropped frame with the frame number, emotion label and the probability. However, this can be done effectively if the RAM is sufficient enough (at least 8GB).

6.3. Results from Experiments

6.3.1. Video Stream/ Real-Time

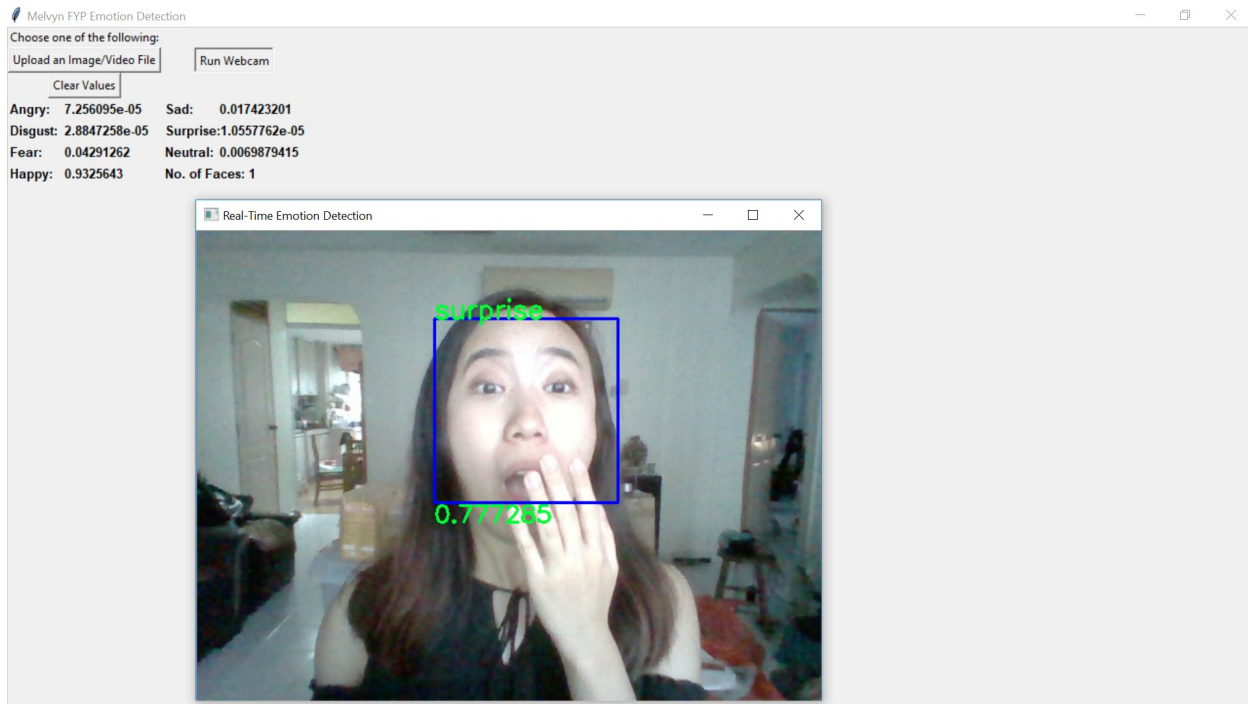


Figure 27 - Emotion Detection (Real-Time)

The 7 categories of emotions were detected: Anger, Disgust, Fear, Happy, Sad, Surprise and Neutral. After trying out numerous times of experimentations on getting the best output from the results, a few factors were figured out. To better detect the expressions, the conditions below must be met [8]:

1. Spectacles must be removed for better accuracy
2. Avoid sudden movement
3. Must have appropriately bright lighting condition
4. Eyebrows must be seen

During the experiment for real-time emotion detection, the test subject watches a video that would evoke certain emotions. And based on the image frames captured in the new folder, the accuracy of the predictions from emotion detection will be evaluated.

The results were accurate when it comes to emotions like 'Happy', 'Surprise' and 'Neutral'. There were many times where the program was confused with 'Disgust' and 'Fear'.

6.3.2. Static Images

In this part of the experiment, various kinds of images are uploaded and fed to the model to get emotion labels on every face as the output. The number of faces is also accounted for in the “No. of Faces” label in the GUI.

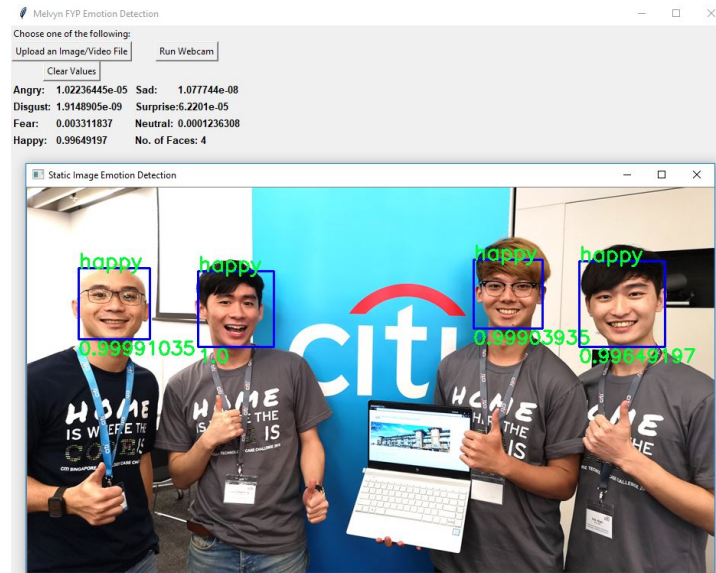


Figure 28 – 4 Faces from Static Image

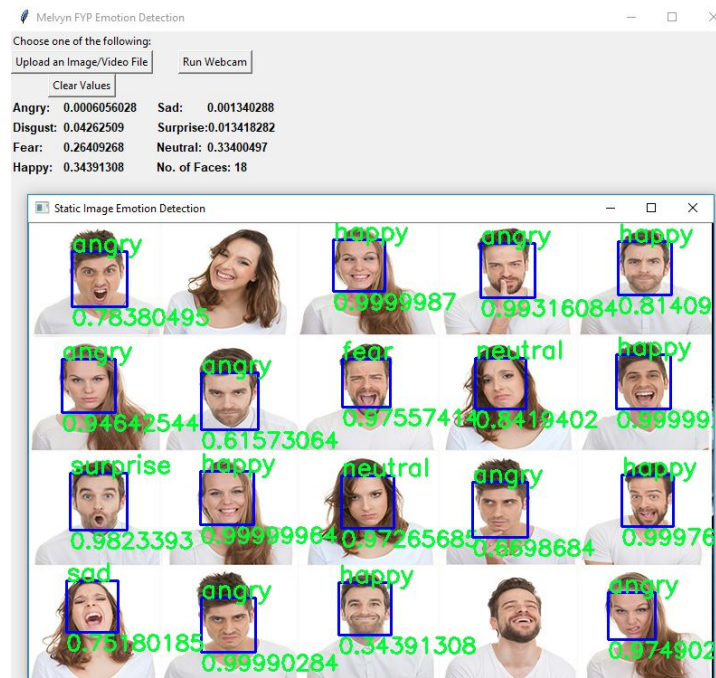


Figure 29 – 20 Faces from Static Image

In Figure 28, it proved that the system can detect more than 1 face at a time, with all the emotion label detected on each face with the corresponding probability. The experiment was pushed further by experimenting 24 faces, all in one image. The resulting post-processed image by the model in Figure 24 shows that it is able to detect all emotions except for 2 persons', and this is probably due to the fact that the faces are not detected by the face cascade classifier before passing it on to the model for emotion classification.

Chapter 7: Conclusion and Future Works

7.1. Conclusion

Based on the experimental results, the emotion detection in real-time system that was built achieved an average accuracy rate of 57%, which is significantly higher than that in the recent studies that achieved an accuracy rate of 48% [12]. A lesson learnt from this project is that the data set is the utmost important factor in building a good model. No matter how versatile the ANN is created, the outcome will never be of satisfactory result if the datasets are bad, and in this case, using the Google stock images as training set.

The performance of this system is poor when it is in a constrained environment, such as sudden change in illumination, camera angle, pose and so on which affects the accuracy of emotion detection.

The reason why the emotion 'surprise' has the highest true positive rate is due to the fact that it has the most distinct facial feature that can be easily distinguished from the rest of the emotions. For instance, a person opens his mouth widely and his eyes are unusually wider. However, for emotions with low true positive rates like 'fear' and 'sad' can be quite confusing at times as certain people express themselves differently. Some people's subtle sadness might be distinguished as 'fear', for instance.

7.2. Future Works

For future works, I will be working on upgrading the system from a real-time Emotion Detection to a real-time Micro-Expression Detection. Micro-expressions are instantaneous and involuntary reflections of human emotion. Because micro-expressions are short-lived, lasting only a few frames within a video sequence, they are difficult to perceive and interpret correctly, and they are highly challenging to identify and categorize automatically.

They usually last a fraction of a second, or to be exact, occurring within $1/25^{\text{th}}$ of a second. These expressions reveal the true emotions of the person. For example, a disappointed person will potentially reveal a flash of pain and fear that disappears off the face in a fraction of a second.

This idea was thought to be applicable in many real-world situations like airport security checks, job interviews and even visitors at home. Many big companies are reportedly missing out great talents as it is not humanly plausible to interview every candidate, resulting in a lottery process by the hiring team [13]. The deployment of real-time emotion and micro-expression detection system would make a great impact on the hiring process in many companies.

A micro-expression is detected if there is a sudden change in emotion classification within $1/25^{\text{th}}$ of a second. For example, if a nervous test subject displays a happy face, a 'happy' expression is detected and at some point, the subject subconsciously displays a 'fear' expression then back to 'happy' again $1/25^{\text{th}}$ second later. In this situation, a hidden emotion of a person is detected, and the system concludes that the person's current emotional state is fearful.

References

- [1] D. Borza, R. Itu and R. Danescu, "Real-time micro-expression detection from high speed cameras," 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, 2017, pp. 357-361.
- [2] Paul Ekman, "Emotions Revealed: Understanding Faces and Feelings". Phoenix, 2004
- [3] N. Fragopanagos and J. G. Taylor, "Emotion recognition in human-computer interaction," Neural Networks, vol. 18, no. 4, pp. 389–405, 2005
- [4] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," IEEE international conference on computer vision and pattern recognition, vol. 1, p. 511–518, 2001.
- [5] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85–117, Jan. 2015.
- [6] "Face Detection using Haar Cascades." [Online]. Available: https://docs.opencv.org/3.4.2/d7/d8b/tutorial_py_face_detection.html
- [7] Google Brain, "TensorFlow: A system for large-scale machine learning" [Online]. Available: <https://www.usenix.org/conference/osdi16/technicalsessions/presentation/abadi>
- [8] E. J. G. S. Appuhamy and B. G. D. A. Madhusanka, "Development of a GPU-Based Human Emotion Recognition Robot Eye for Service Robot by Using Convolutional Neural Network," 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, Singapore, 2018, pp. 433-438.
- [9] "Face Detection Using OpenCV and Python: A Beginner's Guide" [Online]. Available: <https://www.superdatascience.com/opencv-face-detection/>
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Adv. Neural Inf. Process. Syst., pp. 1–9, 2012.
- [11] F. Ertam and G. Aydın, "Data classification with deep learning using Tensorflow," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, 2017, pp. 755-758.
- [12] Arushi Raghuvarshi and Vivek Choksi (2016). Facial expression recognition with convolutional neural networks. CS231n Course Projects
- [13] H. Pettit (2018), "Canadian startup uses AI to help companies hire new talent", Daily Mail. [Online]. Available: <https://www.dailymail.co.uk/sciencetech/article-6131021/Canadian-startup-uses-AI-help-companies-hire-new-talent.html>

- [14] “Convolutional Neural Networks for Beginners: Practical Guide with Python and Keras” [Online]. Available: <https://towardsdatascience.com/convolutional-neural-networks-for-beginners-practical-guide-with-python-and-keras-dc688ea90dca>
- [15] “Epoch vs Batch Size vs Iterations” [Online]. Available: <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>

Appendix A: Preprocessing of Dataset

```
#using FER2013 datasets from Kaggle competition
#with open('/dataset/fer2013/fer2013.csv') as f:
with open('C:/Users/User/Documents/GitHub/FYP_Emotion_Detection/Project2/dataset/fer2013/fer2013.csv') as f:
    content = f.readlines()

lines = np.array(content)    #35888 lines

num_of_instances = lines.size
print("number of instances: ",num_of_instances)
print("instance length: ",len(lines[1].split(",")[1].split(" ")))

#-----
#initialize trainset and test set
x_train, y_train, x_test, y_test = [], [], [], []

#-----
#transfer train and test set data
for i in range(1,num_of_instances):
    try:
        emotion, img, usage = lines[i].split(",")

        val = img.split(" ")

        pixels = np.array(val, 'float32')

        emotion = keras.utils.to_categorical(emotion, num_classes)

        if 'Training' in usage:
            y_train.append(emotion)
            x_train.append(pixels)
        elif 'PublicTest' in usage:
            y_test.append(emotion)
            x_test.append(pixels)
    except:
        print("",end="")

#-----
#data transformation for train and test sets
x_train = np.array(x_train, 'float32')
y_train = np.array(y_train, 'float32')
x_test = np.array(x_test, 'float32')
y_test = np.array(y_test, 'float32')

x_train /= 255 #normalize inputs between [0, 1]
x_test /= 255

x_train = x_train.reshape(x_train.shape[0], 48, 48, 1)
x_train = x_train.astype('float32')
x_test = x_test.reshape(x_test.shape[0], 48, 48, 1)
x_test = x_test.astype('float32')

print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```


Appendix B: Construction of CNN Structure

```
#construct CNN structure
model = Sequential()

#1st convolution layer
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(5,5), strides=(2, 2)))

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

model.add(Flatten())

#fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax')) #last layer usually uses softmax
#-----
#batch process
gen = ImageDataGenerator()
train_generator = gen.flow(x_train, y_train, batch_size=batch_size)

#-----
#use cross entropy because categorising emotions is a form of multi-class classification
model.compile(loss='categorical_crossentropy'
              , optimizer=keras.optimizers.Adam()
              , metrics=['accuracy']
              )

#SAVING MY TRAINED MODEL
#-----
fit = True

if fit == True:
    #To complete the training in less time, I prefer to implement learning with randomly selected trainset instances.
    #That is the reason why train and fit generator used
    #model.fit_generator(x_train, y_train, epochs=epochs) #train for all trainset
    model.fit_generator(train_generator, steps_per_epoch=batch_size, epochs=epochs) #train for randomly selected one
else:
    model.load_weights('/data/facial_expression_model_weights.h5') #load weights
```

Appendix C: Main Script for Emotion Detection

Real-Time Video Capture

```
#For VideoStream/Live Stream or Emotion Detection
def select_vs():
    cap = cv2.VideoCapture(0) #real-time streaming using webcam
    cwd = os.getcwd()
    time_now = datetime.now().strftime("%Y%m%d-%H%M%S")
    new_folder = "frames_captured_{0}".format(time_now)
    new_folder_dir = "{0}\\{1}\\{2}".format(cwd, 'python', new_folder)
    createFolder(new_folder, new_folder_dir)

    #counter for filename later for faces captured in each frame
    j = 0
    while(True):
        ret, img = cap.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        #locations of detected faces

        for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2) #draw rectangle to main image
            detected_face = img[int(y):int(y+h), int(x):int(x+w)] #crop detected face
            detected_face = cv2.cvtColor(detected_face, cv2.COLOR_BGR2GRAY) #transform to gray scale
            detected_face_mini = cv2.resize(detected_face, (48, 48)) #resize to 48x48
            img_pixels = kerasimage.img_to_array(detected_face_mini)
            img_pixels = np.expand_dims(img_pixels, axis = 0)
            img_pixels /= 255 #pixels are in scale of [0, 255]. normalize all pixels in scale of [0, 1]

            #predictions is a 2-d array! predictions[0] contains probabilities of 7 emotions respectively
            predictions = model.predict(img_pixels) #store probabilities of 7 expressions

            #find max indexed array 0: angry, 1:disgust, 2:fear, 3:happy, 4:sad, 5:surprise, 6:neutral
            max_index = np.argmax(predictions[0]) #returns the index of the max value
            max_prob = np.max(predictions[0])

            #probabilities of each expression
            angry_prob = predictions[0][0]
            disgust_prob = predictions[0][1]
            fear_prob = predictions[0][2]
            happy_prob = predictions[0][3]
            sad_prob = predictions[0][4]
            surprise_prob = predictions[0][5]
            neutral_prob = predictions[0][6]
            i=0
            for i in range(7):
                emotions_probList[i] = predictions[0][i]

            set_label(emotions_probList)
            print(emotions_probList)
            emotion_captured = emotions[max_index] + ' ' + str(max_prob)
```



```

#write emotion text above rectangle
cv2.putText(img, emotion_captured, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

#saving of frames captured into frames_captured folder
j+=1
FaceFileName = "./python/{0}/frame_{1}_{2}.jpg".format(new_folder, j, emotion_captured)
cv2.imwrite(FaceFileName, detected_face)
#process on detected face end
#-----

cv2.imshow('Real-Time Emotion Detection',img)

if cv2.waitKey(1) & 0xFF == ord('q'): #press q to quit
    break

#kill open cv things
cap.release()
cv2.destroyAllWindows()

```

Upload Static Image

```

#For static Image for Emotion Detection
def select_image():
    #grab a reference to an image panel
    global panelA
    global angry_prob, disgust_prob, fear_prob, happy_prob, sad_prob, surprise_prob, neutral_prob
    panelA = None
    #open a file chooser dialog for user to select an input image
    path = filedialog.askopenfilename()

    #ensure a file path was selected
    if(len(path) > 0):
        image = cv2.imread(path)
        image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(image_gray, 1.3, 5)

        #locations of detected faces
        for (x,y,w,h) in faces:
            cv2.rectangle(image, (x,y), (x+w, y+h), (255,0,0),2) #draw rectangle to main image
            detected_face = image[int(y):int(y+h), int(x):int(x+w)] #crop detected face
            detected_face = cv2.cvtColor(detected_face, cv2.COLOR_BGR2GRAY) #transform to gray scale
            detected_face_mini = cv2.resize(detected_face, (48, 48)) #resize to 48x48
            img_pixels = kerasimage.img_to_array(detected_face_mini)
            img_pixels = np.expand_dims(img_pixels, axis = 0)
            img_pixels /= 255 #pixels are in scale of [0, 255]. normalize all pixels in scale of [0, 1]

            #store probabilities of 7 expressions
            predictions = model.predict(img_pixels)

            #####3
            #find max indexed array 0: angry, 1:disgust, 2:fear, 3:happy, 4:sad, 5:surprise, 6:neutral
            max_index = np.argmax(predictions[0]) #returns the index of the max value
            max_prob = np.max(predictions[0])

```

```

#probabilities of each expression
angry_prob = predictions[0][0]
disgust_prob = predictions[0][1]
fear_prob = predictions[0][2]
happy_prob = predictions[0][3]
sad_prob = predictions[0][4]
surprise_prob = predictions[0][5]
neutral_prob = predictions[0][6]
i=0
for i in range(7):
    emotions_probList[i] = predictions[0][i]
set_label(emotions_probList)

#this is the emotion distinguished by the model
emotion_captured = emotions[max_index] + ' ' + str(max_prob)

#write emotion text above rectangle
#####
WHITE TEXT (255,255,255)
final_image = cv2.putText(image, emotion_captured, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
#final_image = cv2.putText(image, emotion_captured, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
#####
cv2.imshow('Static Image Emotion Detection', final_image)

#In order to display our images in the Tkinter GUI, we first need to change the formatting.
#To start, OpenCV represents images in BGR order; however PIL/Pillow represents images in RGB order, so we need to reverse the ordering of the channels
#convert image to PIL format
final_image = Image.fromarray(final_image)
#then convert to ImageTk format
final_image = ImageTk.PhotoImage(final_image)

```

Appendix D: Results from Experiments

Angry

Melvyn FYP Emotion Detection

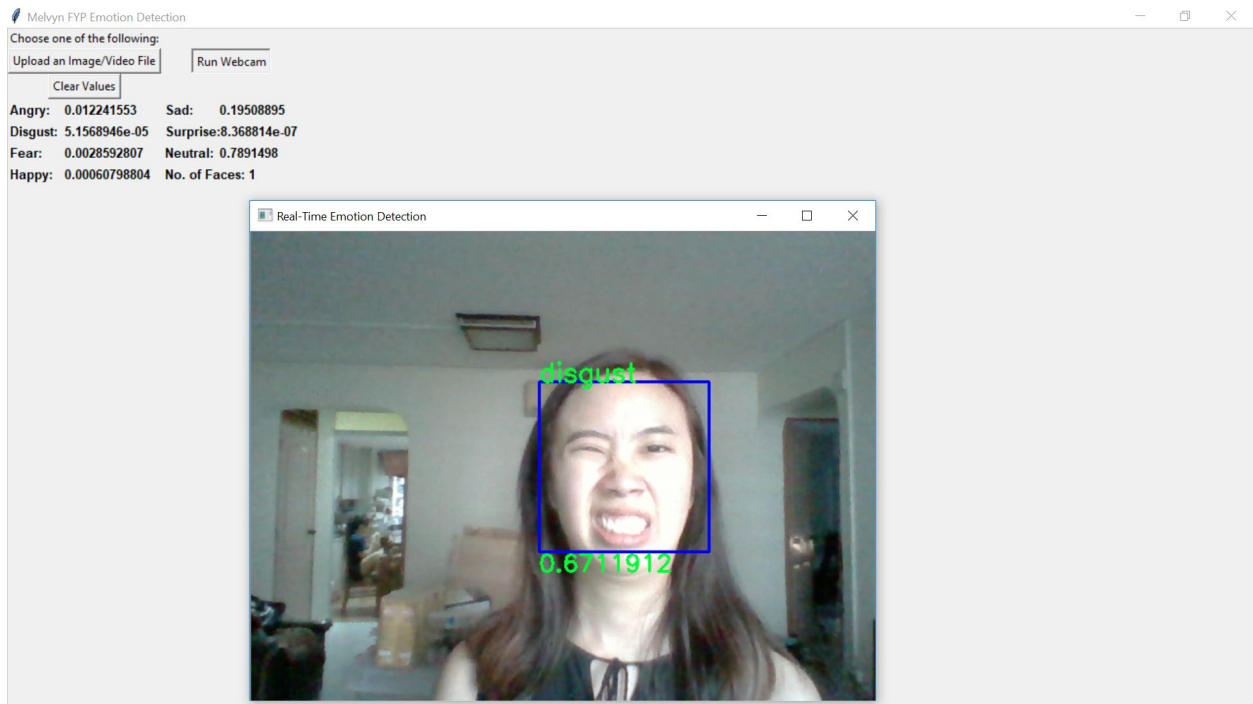
Choose one of the following:

Angry: 7.256095e-05	Sad: 0.017423201
Disgust: 2.8847258e-05	Surprise: 1.0557762e-05
Fear: 0.04291262	Neutral: 0.0069879415
Happy: 0.9325643	No. of Faces: 1

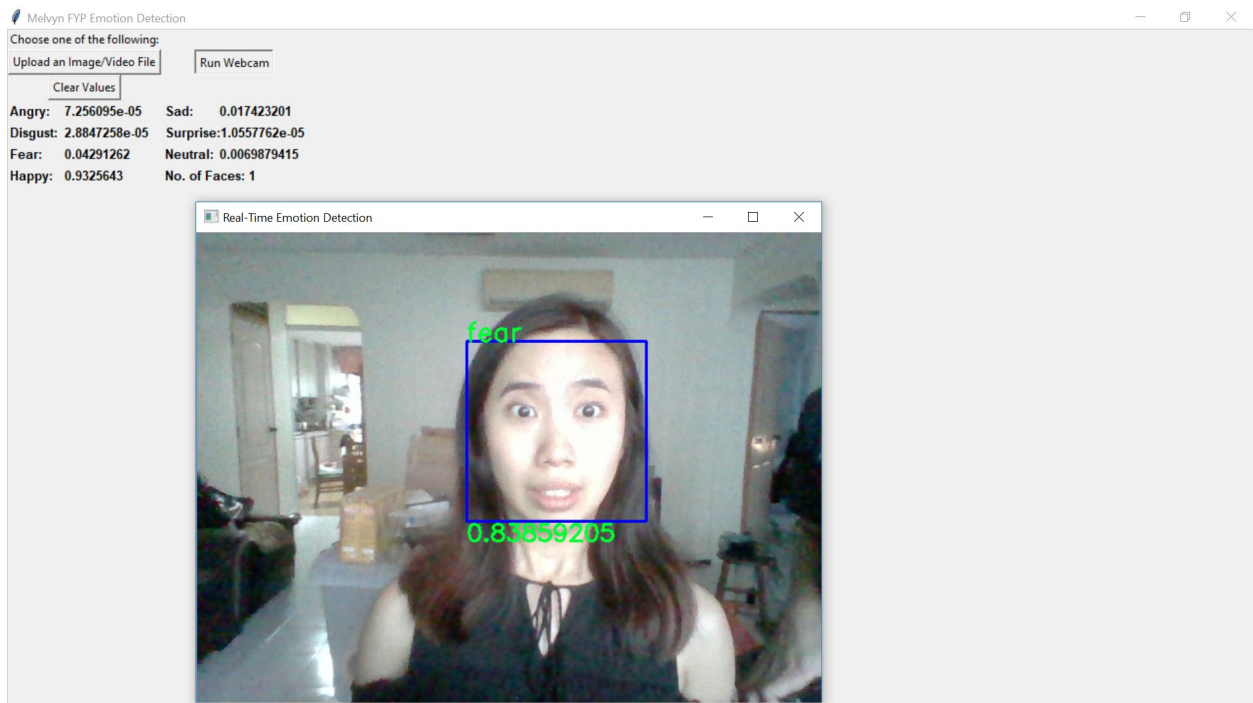
Real-Time Emotion Detection

The image shows a real-time video feed of a woman's face. A blue rectangular bounding box is drawn around her face. The word 'angry' is written in green text above the box, and the confidence score '0.48052475' is written in green text below the box.

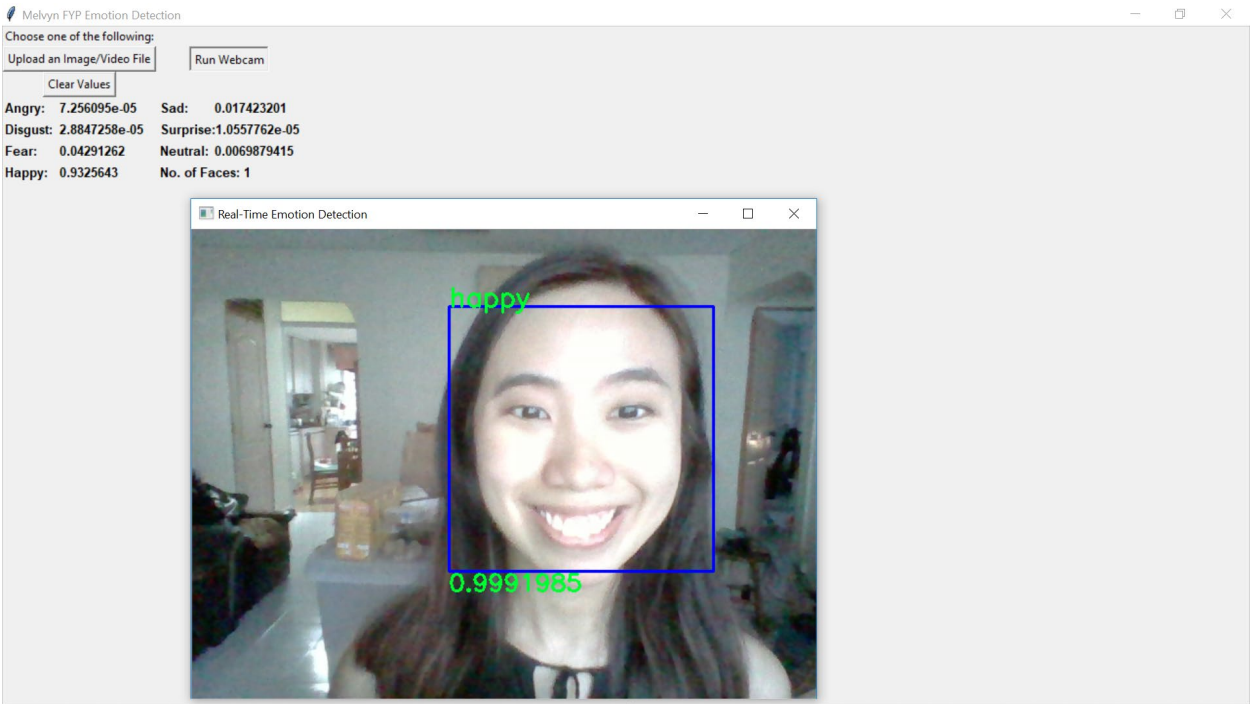
Disgust



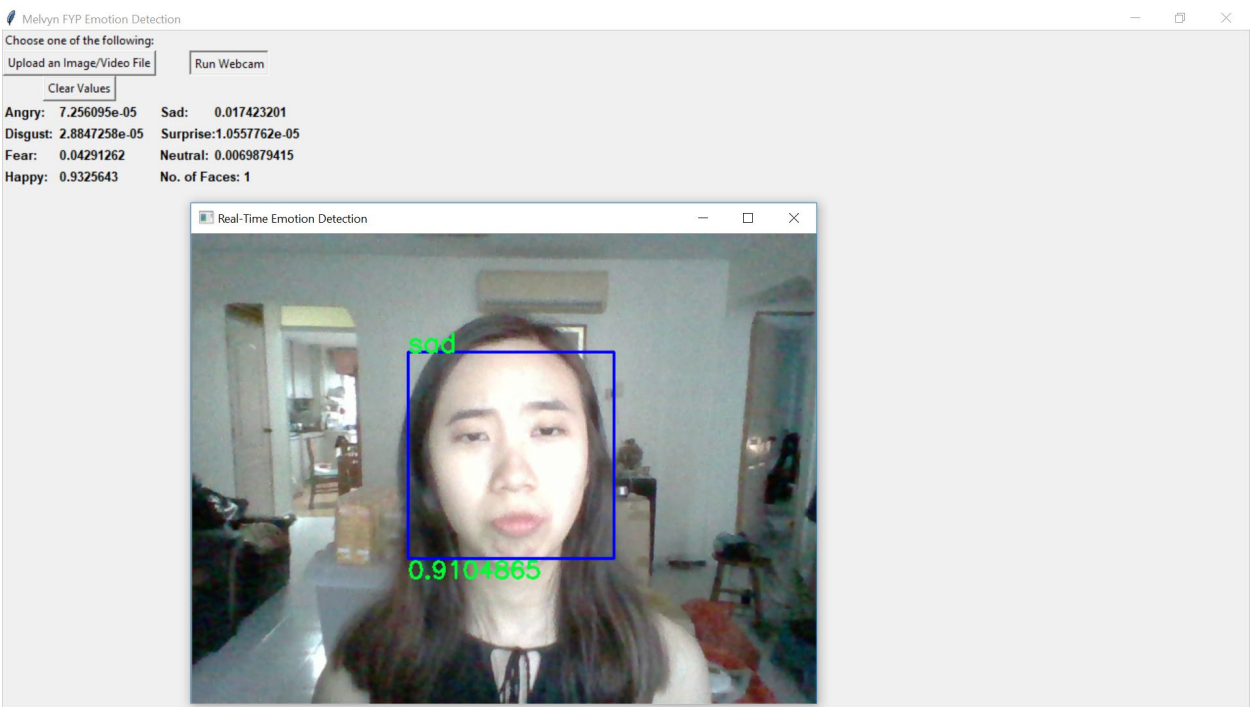
Fear



Happy



Sad



Surprise

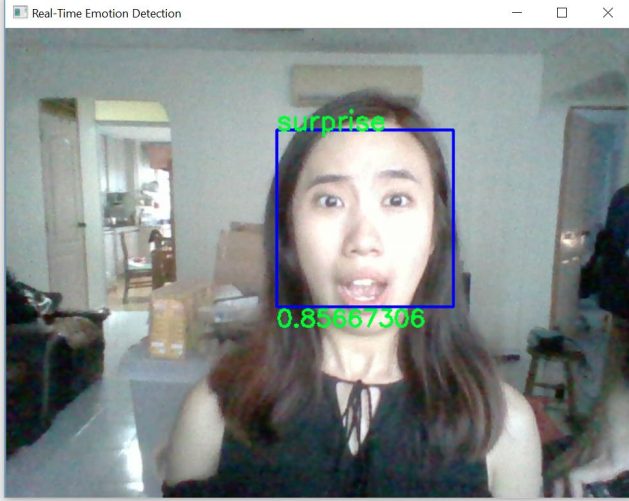
Melvyn FYP Emotion Detection

Choose one of the following:
Upload an Image/Video File Run Webcam

Clear Values

Angry: 7.256095e-05 Sad: 0.017423201
Disgust: 2.8847258e-05 Surprise: 1.0557762e-05
Fear: 0.04291262 Neutral: 0.0069879415
Happy: 0.9325643 No. of Faces: 1

Real-Time Emotion Detection



surprise

0.8566/306

Neutral

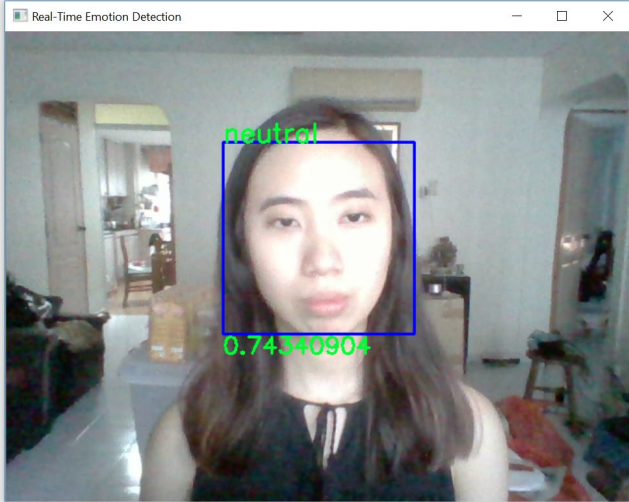
Melvyn FYP Emotion Detection

Choose one of the following:
Upload an Image/Video File Run Webcam

Clear Values

Angry: 7.256095e-05 Sad: 0.017423201
Disgust: 2.8847258e-05 Surprise: 1.0557762e-05
Fear: 0.04291262 Neutral: 0.0069879415
Happy: 0.9325643 No. of Faces: 1

Real-Time Emotion Detection



neutral

0.74540904