Project Report

on

[Emotion Detection using Raspberry Pi with Voice Command]

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

for part fulfilment of the requirements for the award of degree of

Bachelor of Computer Applications

Submitted By

Rudrani Ghosh (11600841)

Prerna Kumari (11605846)

Ankur Jha (11617192)

Dinesh Reddy (11600959)

Abhay Kumar (11601434)

Supervised By

Dr. Narinder Singh Rana

(Associate Professor)

LOVELY FACULTY OF TECHNOLOGY & SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

[April 2019]

DECLARATION

We hereby declare that the Capstone Project report **“Emotion Detection using Raspberry Pi with Voice Command”** is our project, and that we have accomplished it under the guidance of Dr. Narinder Singh Rana, Associate Professor, Department of Computer Applications, Lovely Professional University for the fulfillment of our degree Bachelor of Computer Applications(Hons.)
We have not submitted this project partially or totally for any fellowship or any similar titles or prizes.

Date:23rd April 2019

Place: Lovely Professional University, Phagwara

Signature:________________________(Rudrani Ghosh,11600841)

Signature:________________________(Prerna Kumari, 11605846)

Signature:________________________(Ankur P Jha, 11617192)

Signature:________________________(Dinesh Reddy, 11600959)

Signature:________________________(Abhay Kumar, 11601434)

Certificate

TO WHOMSOEVER IT MAY CONCERN

This is to certify that the project entitled ***"Emotion Detection Using Raspberry Pi with Voice Command"*** was presented by ***"Rudrani Ghosh(11600841), Prerna Kumari(11605846), Ankur Prakash Jha(11617192), Dinesh Reddy(11600959), Abhay Kumar(11601434)",*** student of BCA 6th Semester (Session 2016-2019), in partial fulfillment of the requirement for the award of the degree of Bachelors of Computer Applications(Hons), is a record of bonafide work carried out by them under my supervision.

In my knowledge, this work has not been submitted, either in part or in full, to any other university or institute for the award of degree or diploma.

(Dr. Narinder Singh Rana)
Associate Professor
School of Computer Application
Lovely Professional University

Acknowledgement

We thank the almighty for giving us the courage and perseverance in completing the Capstone Project on “Emotion Detection using Raspberry Pi”. This project itself is acknowledgement for all those people who gave us their heartfelt co-operation in making this project a grand success.
We extend our sincere thanks to Mr. Ashok Mittal, Chancellor of our University, for providing sufficient infrastructure and good environment in the University to complete our course and project.
We are thankful to our Head of School Mr. Ashwani Tewari for providing the necessary Infrastructure and labs, faculty support and guidance, and also permitting us to carry out this project.
We are thankful to our Head of Department Mr. Rishi Chopra for providing the constant encouragement, support and his valuable insights to complete this project.
With extreme jubilance and deepest gratitude, I extend special thanks to our Project coordinator, Dr. Narinder Rana, Associate Professor, School of Computer Applications, for his support and valuable suggestions regarding project work and for providing valuable guidance at every stage of this project.We are profoundly grateful towards the unmatched services rendered by him.
Our special thanks to all the faculty of School of Computer Applications and peers for their valuable advices at every stage of this work.
Last but not least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing this project.

Table of Contents

Chapter 1: Introduction

Emotions enable a form of communication among human beings. Complex social communication come into context with the understanding of Emotion. Emotion Detection can be done via voice, body gestures and other complex methods. There are many practical methods to examine facial emotions too.

There are seven types of human emotions that are universally recognized. The seven we are talking about includes happiness, sadness, fear, surprised, anger, disgust and neutral.

A service that detects emotion from facial emotions would be widely applicable, as such a service can bring advancement in various applications of gaming, marketing, consumer product satisfaction and entertainment.

Emotion Detection has attracted significant attention in the advancement of human behaviour and machine learning.

Various applications related to face and emotion detection include:

1. Personal Credentials and Access Control.
2. Video Phone and Teleconferencing.
3. Medical and Forensic Applications
4. Gaming and Applications
6. Analysing human behavior to ascertain work satisfaction

There are four categories of methods that are used to detect human face, namely:

1. Feature Method: Based on facial features like placement of eyes, nose, contour.
2. Knowledge Method: Pre-Trained Models as we instill our model with data sets.
3. Appearance Method: Based on Neural Networks approach
4. Template Method: Checks for the correlation between standard image and input image face pattern

These methods if used separately, cannot solve all the problems of face detection like orientation, pose and expression.

The difficulty one face with the emotion recognition is basically due to the following:

1. There is a moderate size database for training images.
2. Based on input image whether it is a static frame or it is a transition frame it becomes difficult to classify the emotion.

For real-time detection in which facial expressions differs dynamically, it is very difficult to detect emotions.

In the present emotion detection application examine static images of facial emotions. We will inspect the system that will do the emotion recognition in real time with a live video streaming. Computation of frame-by-frame classification is necessary for live detection. So we have developed a system for detecting emotions in real time. The result we achieved is an innovative system where an emotion-indicating text is displayed on screen.

Chapter 2: Objectives

The aim of the project is to come up with a solution to problem of emotion detection by dividing it into sub-problems. The scope of project does not only the two problems which will tell us whether a human face is detected or not, but also the multi-class problems that will take the user input, to start detecting the face and afterwards starts with a live emotion detection. For this different methodologies and techniques for dataset training, selection and classification, solutions to the problems as well as taking the computational complexity and timing issues has been considered. The major objective of the project is to implement emotion detection in terms of run time on the embedded system. Various Hardware Resources has been used to achieve the goal. Also various algorithms and methodologies are studied to attain the desired goals. Such type of emotion detection system can be useful for our day to day tasks. This system may would help the human life in near future.

The typical application for this system are listed here:

1. Business Meetings
2. Social Gatherings
3. Teaching Assistant
4. Education
5. Audio-Visual Speech Recognition
6. Gaming Experience
7. Multimedia

Chapter 3: System Analysis

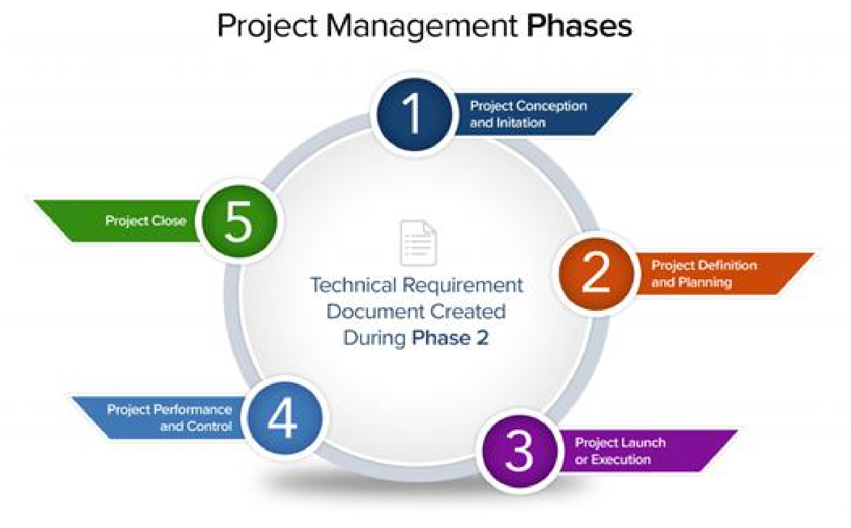
3.1 Identification of Need

Purpose: Emotion Detection is a very essential research topic for psychologists. Current advancements in image processing and emotion recognition has motivated research work on Automated Emotion Recognition. In past, to detect facial emotions from still images, lots of effort were dedicated. To fulfil this need of emotion detection, many techniques have been applied and neural networks, Gabor wavelets, and active appearance models are one of them. The only limitation that was caught with this strategy is that the still images only captures the apex of emotion from expression, that is the instant at which the indicators of emotion are most marked. People rarely show this apex of their Facial Emotions during normal communication. It can be captured for very precise cases and is short-term.

3.1.1 Technical requirement:

The aspects of our technical requirements are:

- Determining our budget
- Create our work breakdown schedule
- Develop the project DFDs
- Initiate a communication plan
- Define risk-management aspects


Project Management Phases
1
Project Conception and Initation
2
Project Definition and Planning
3
Project Launch or Execution
4
Project Performance and Control
5
Project Close
Technical Requirement Document Created During Phase 2

3.1.2. Economical Requirements:

The Economic requirement here deals with the Economic Assessment. It is a very important tool for achieving results like value for money and user satisfaction. Economical Assessment is a process for making alternative use of resources such as Raspberry Pi or PC. We are also focussing on analysis of customer needs, goal of project, options available, costing, various benefits and affordability.

1. Wide Scope of Economic Requirements.
2. Covers the cost and benefits for LPU.

3.1.3 Environmental Requirements:

1. Our project work is not creating any disastrous effect on environment.
2. We can say that our project is eco-friendly.

Chapter 4: Preliminary Investigation

4.1 Previous Work Done On Emotion Recognition

After going through various literature reviews it is found that Facial Emotion Recognition is a procedure that is implemented by both machines and human beings.

Computer needs to follow the following process that consist the following:

1. Finding or locating various faces in a live stream captured by camera. This is a step known as Face Detection.

2. From detected regions on the face, extracting various facial features like facial components, skin texture of face, etc. This step is called Facial Feature Extraction.

3. Analysing changes in appearance of facial features and categorizing these features into categories that can interpret Facial Expressions into various emotions like angry, happy, sad, etc. This step is called Facial Expression Interpretation.

On the basis of some systems that has already been designed in this arena, we found that this project can be implemented with four steps.
1. Identifying and Pre-processing
2. Registering Face
3. Extracting Facial Features
4. Emotion Detection

These processes are described here:

Identifying and Pre-Processing:

Operations with the image at the entry level of abstraction is known as Pre-processing. Steps involved in Pre-processing are as follows:

1. **Noise Reduction**
2. **Converting image to grayscale**
3. **Pixel Transformation**
4. **Geometric Transformation**

Registering Face

Identification of Human Faces as Digital Images make use of Face Registration. During the process of Face Registration, first we locate the image with the help of predefined set that contains landmark points. This is known as "Face Localization" or "Face Detection". Faces that are detected have been Geometrically Normalized so that it can match the dataset images also known as template image. This is called as Face Registration.

Extracting Facial Features

Process of locating specific facial regions, points, curves, landmarks and contours in a given 2D image or a 3D image is known as Facial Feature Extraction. For Face Extraction, Registered Image generate a numerical vector of facial features.

We can extract following features:

1. Eyes
2. Nose Tip
3. Lips
4. Eyebrows

Emotion Detection

At this step, algorithm classifies the face on the basis of emotions and data images. Emotion Detection can be done using various approaches.

1. Neural Network Approach: Artificial Neural Network Approach is based on the biological neural networks that constitute brains. In this approach neural networks are trained independently. Neural Network is a framework for various Machine Learning Algorithms to process complex data inputs. Learning of these system is based on examples, and they are not programmed for rules specific to the task. For example, they learn to identify images with happy faces by evaluating images that were labelled "happy" manually. They will automatically generate characteristics from the material they processed and it can help them with identification.

2. Gabor Filter: A linear filter, called Gabor Filter, has its uses in texture analysis. It will analyse the content of image for specific frequency that will be fixed for directions in a local region also known as region of analysis. According to vision scientists Human Visual System and Gabor Filters are similar. They are suitable for representing texture and discriminating textures. Some scientists also claim that Gabor Functions can model the cells in the cortex of mammal brain. Perception in human visual system is also on the same lines with Process of image analysis using Gabor Filter

3. Support Vector Machine: Support Vector Network (SVMs) are supervised learning models. They have associated learning algorithms. These algorithms analyse data for analysing regression and classification. If we provide SVM with set of training examples and they are marked which category they will either belong category one or to category two. SVMs will itself build a model. This model will assign various new examples to both categories. SVM perform linear as well non-linear classification.

If there is unavailability of Labelled data, supervised learning cannot be possible. Thus we need an unsupervised learning approach. Hava Siegelmann and Vladimir Vapnik created Support-Vector Clustering algorithm. Statistics of support vectors are applied by this algorithm. Support Vectors are developed in Support Vector Machines Algorithm. They are used to categorize unlabelled data.

4. **Training Database and Testing Database:** We can design algorithms in machine learning that learns from the data and make predictions on similar set of data. These algorithms follow data-driven approach by constructing a mathematical model from the data, to make predictions and decisions.

The final model of algorithm is based on data that comes from Multiple Datasets. For creating such algorithms following three datasets are used in different stages:

1. Training Dataset: Training dataset contains set of examples that help in accommodating various parameters of the algorithm. Using supervised learning we train the model with training dataset. Training Dataset consists of an input vector and the corresponding answer that can be scalar or vector and is indicated as target.

The present model is kept running with the training dataset and produces an outcome, which is then differentiated with the target, for each input vector in the training dataset. In view of the result of the evaluation and the particular learning algorithm that has been used, the parameters of the model are balanced. The model fitting can incorporate both variable choice and parameter estimation.

2. Validation Dataset: Progressively, the fitted model is utilized to foresee the reactions for the observations in a second dataset called the Validation dataset. The Validation Dataset gives an impartial evaluation of a model fit on the Training dataset while tuning the model's hyper parameters (for example the quantity of hidden units in a neural system).

Validation datasets can be utilized for regularization by early halting: quit training when the error on the Validation Dataset increases, as this is an indication of overfitting to the Training dataset. This straightforward method is complicated by the fact that the Validation Dataset error may change amid training, delivering numerous local minima. This inconvenience has prompted the production of some especially decided rules for choosing while overfitting has started.

3. Test Dataset: This dataset provides an impartial assessment of a final model fit on Training Dataset.

Chapter 5: Feasibility Study

5.1 Technical Study

Yes it is technically feasible to implement this project as here in LPU everyone is equipped with 24x7 internet connections which is must for making use of our project, as it deals in Google API for taking in voice commands. Also for training our data-set we are using high speed internet which enables rigorous downloading and uploading of images over a network.

The current work practises and procedures using Raspberry Pi is a bit time consuming when being trained as it only has a 1GB Ram support, which is the bare minimum to support the new system.

Our mode of operation will provide more throughput and response time when implemented on a PC with high a speed processor such as Intel i3 or higher and RAM support above 4GB.

5.2 Economical

Yes it is economically feasible to implement this project. For implementing this project, we have made an investment of INR 6800. ·

Our Budget is as follows:

Raspberry Pi	3000
Microphone	650
USB Camera	500
USB Mouse	400
USB Keyboard	250
Monitor	1500
HDMI Cable	300
Total:	6800

If we implement the overall setup using a desktop PC or laptop, we will not have to bear any of the above costs, as all the above peripherals will be pre-built in it.

5.3 Environmental

We are eliminating use of paper for feedbacks and providing a real time feedback via emotions. We are looking forward to sustain our project output to the elements of internet. So we are on eco-friendly stand in respect of our project.

5.4 Need

This project is made based on the information gathered through conducting a survey, whereby it highlights the real life situation to deal with emotions of employees, customers and service users to understand their level of satisfaction within an organization or with any product or with other services offered.

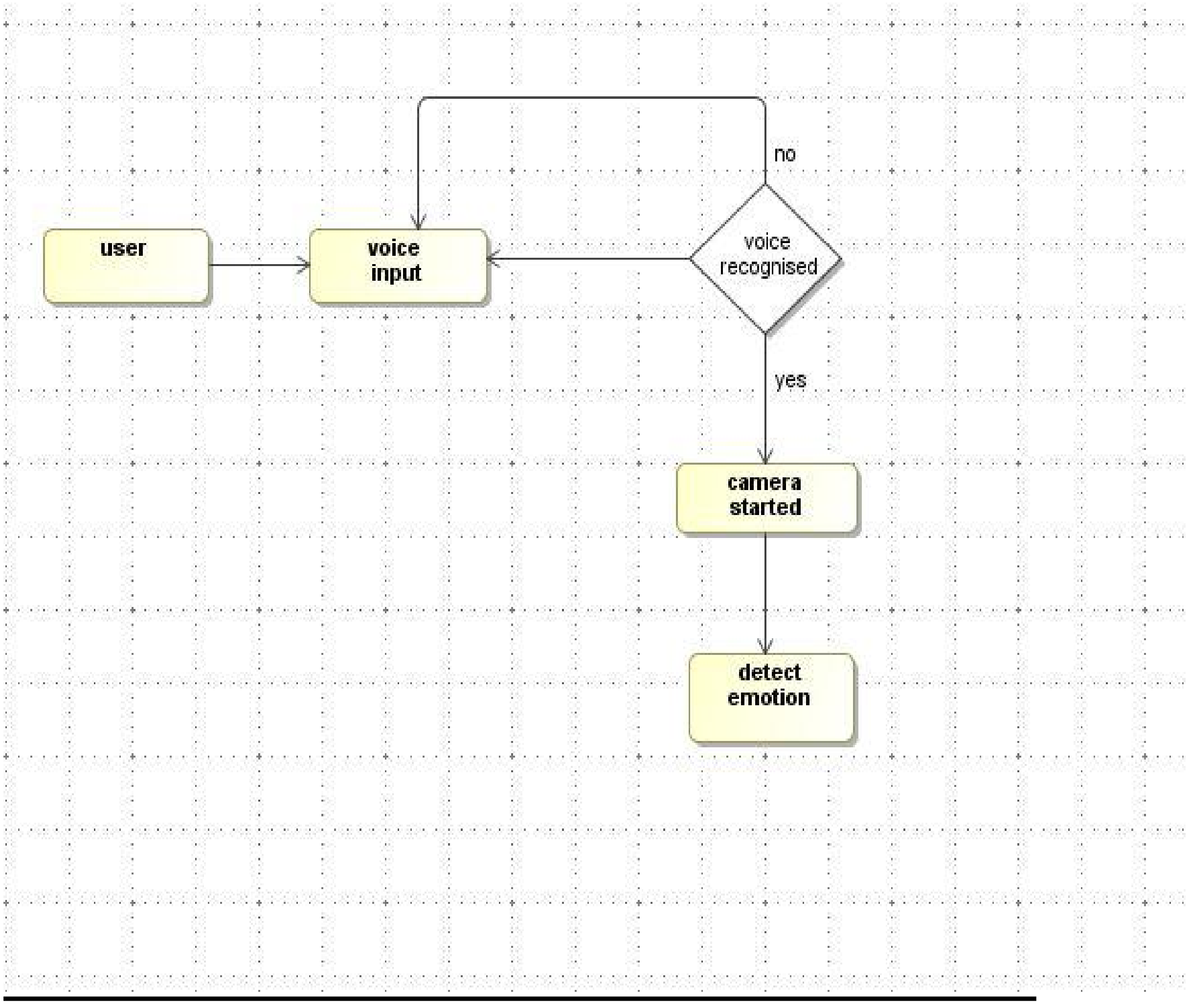
By keeping the record of the emotions, we can understand the user behaviour and user satisfaction and hence improve the services offered. In the long run, this will help in Machine Learning too as machines will be able to deal with human emotions.

5.5 Usability

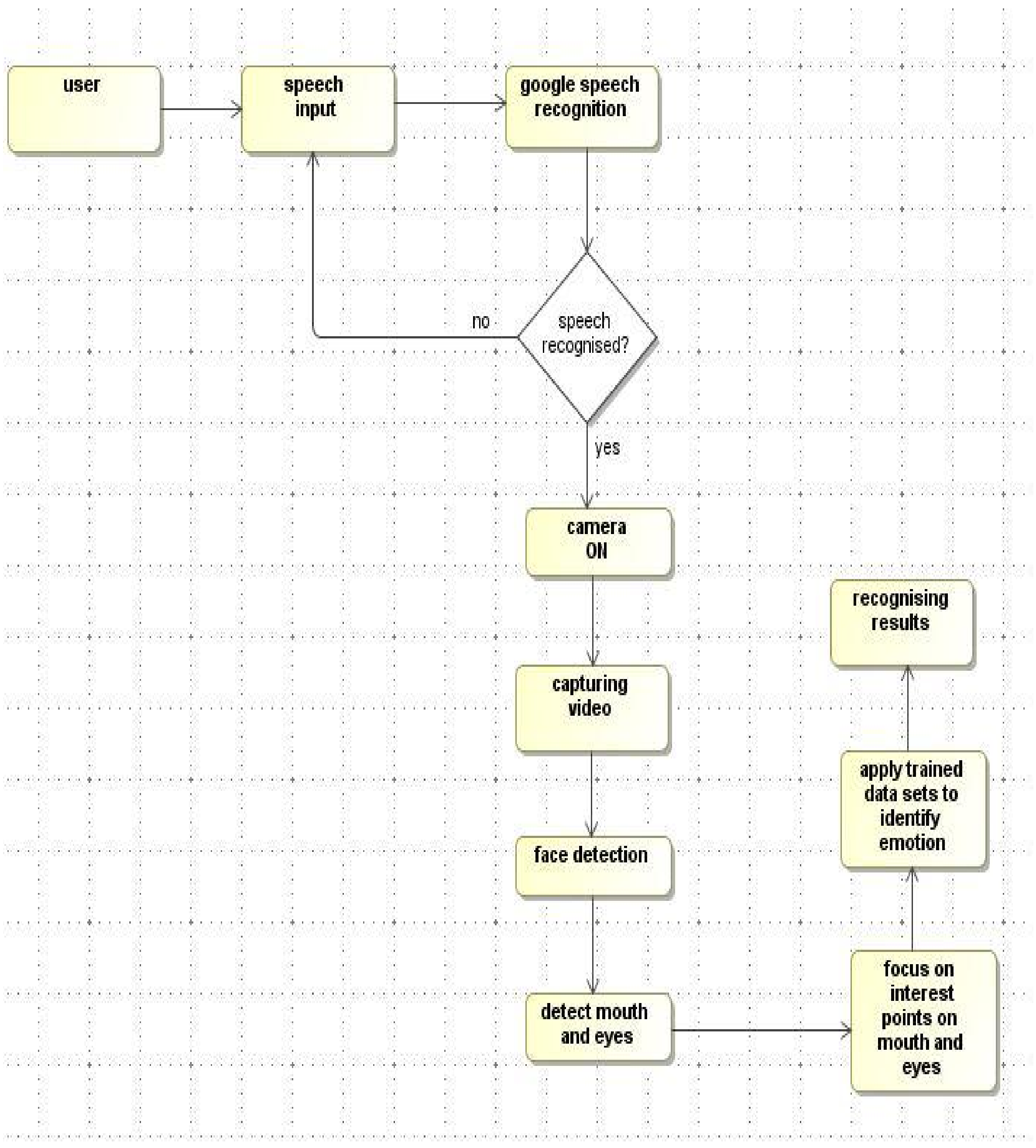
The interface will be developed in way that it would be really very easy to use model and information will be presented in concise and clear way, a Basic English language will be used on the action button of application so as to remove any ambiguity. The user will have to control the model with the voice command so as to avoid unethical use of this model.

Chapter 6: Data Flow Diagrams

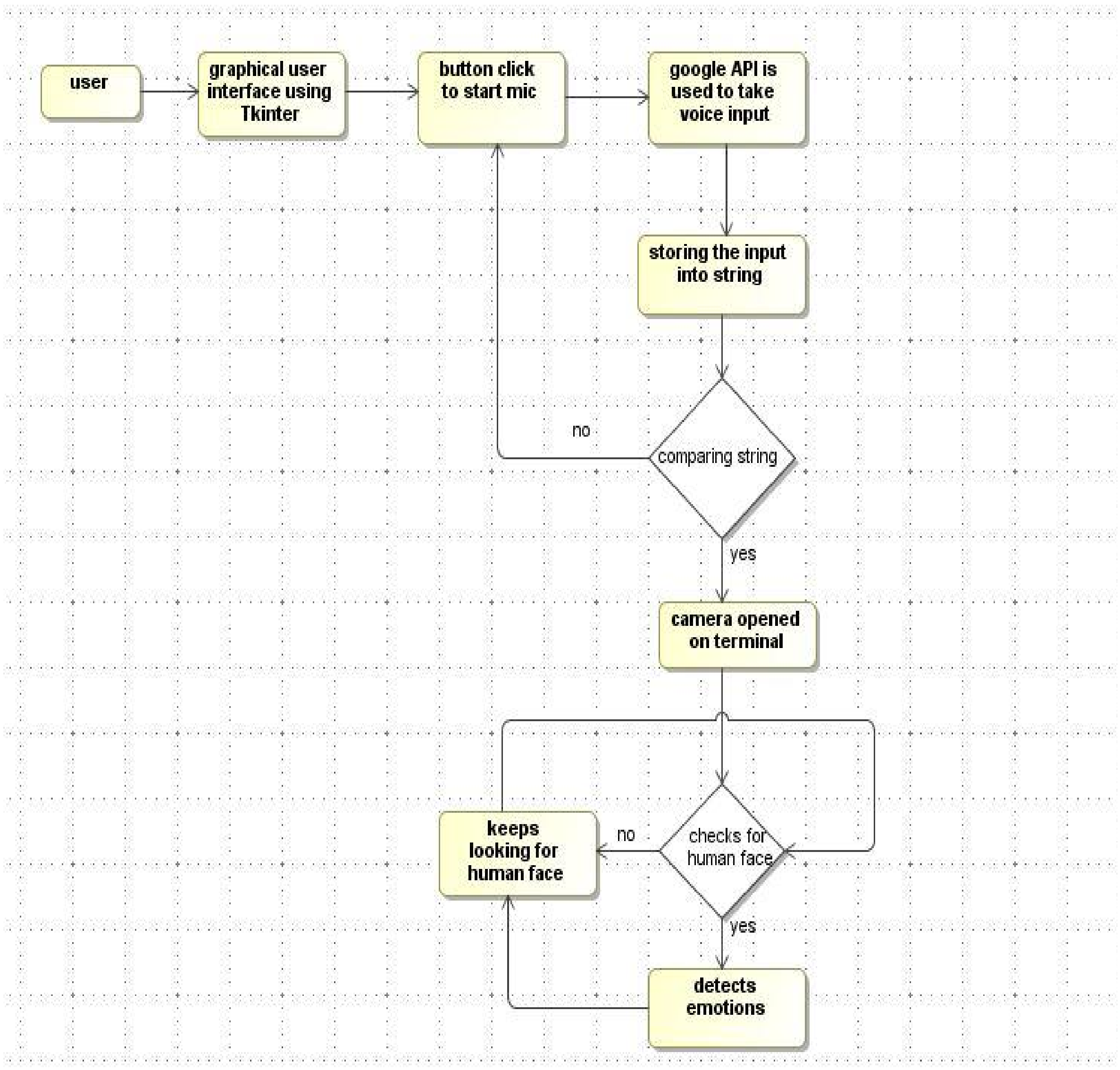
Level -0-DFD



Level-1-DFD


user
speech input
google speech recognition
speech recognised?
no
yes
camera ON
capturing video
face detection
detect mouth and eyes
focus on interest points on mouth and eyes
apply trained data sets to identify emotion
recognising results

Level-2-DFD


user
graphical user interface using Tkinter
button click to start mic
google API is used to take voice input
storing the input into string
comparing string
no
yes
camera opened on terminal
checks for human face
no
keeps looking for human face
yes
detects emotions

Chapter 7: Software and Hardware Specification

7.1 Software

7.1.1 IDLE

IDLE (Integrated Development and Learning Environment) is an integrated development environment for programming in Python. It can be used by many Linux distributions.

Main Features of Python Idle are:

1. Multi-window text editor
2. Supports syntax formatting, auto-completion, smart indentation
3. Python's shell makes use of syntax highlighting
4. Debugger that is integrated by default

We have used IDLE to code in Python Programming Language. Our front-end code is developed using Tkinkter which is used to build Graphical User Interface.

We have done our coding on IDLE version 3.5.1 as it is the last most stable release.

We have downloaded APIs using "pip" command on CMD, such as:-

pip install pyaudio

pip install speech recognition

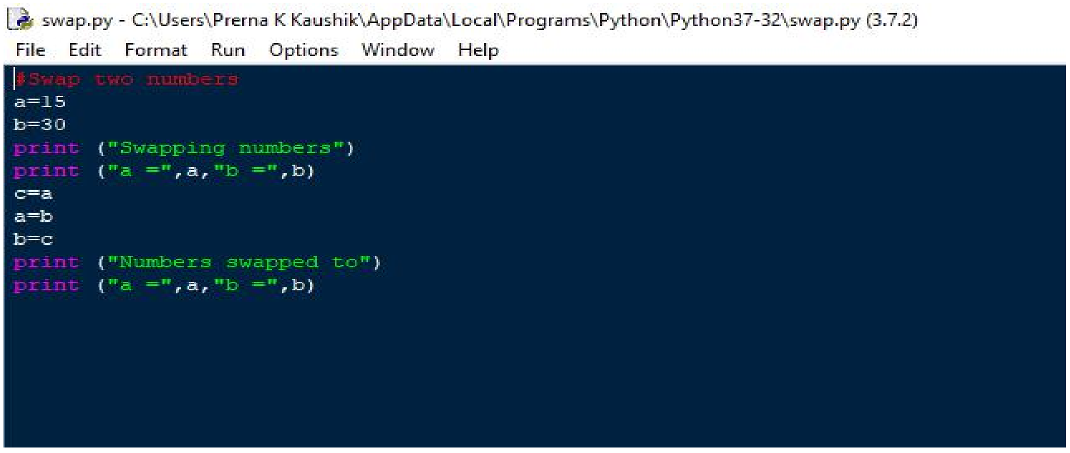
pip install opencv

pip install numpy

pip install gTTS

pip install tensorflow

We use the import command in python.


swap.py - C:\Users\Prerna K Kaushik\AppData\Local\Programs\Python\Python37-32\swap.py (3.7.2)
File Edit Format Run Options Window Help

```
#Swap two numbers
a=15
b=30
print ("Swapping numbers")
print ("a =",a,"b =",b)
c=a
a=b
b=c
print ("Numbers swapped to")
print ("a =",a,"b =",b)
```

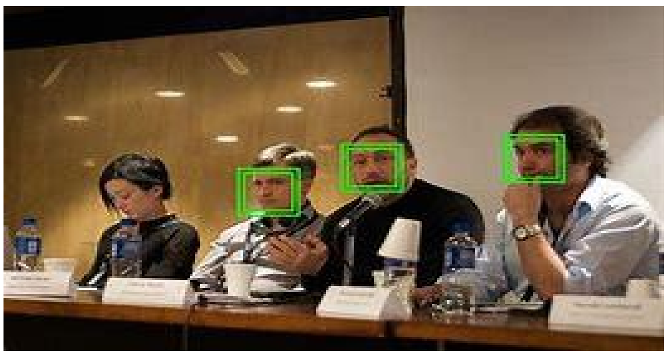
IDLE Interface

7.1.2 We have used multiple libraries in this projects which are:-

1. gTTS :- gTTS abbreviated from Google-Text-to-Speech is a Python library and CLI tool to interact with Google Translate's text-to-speech API. gTTs writes spoken mp3 data to a file and further save it as file-like object called as byte string for further audio manipulation.. It features flexible pre-processing and tokenizing, as well as automatic retrieval of supported languages. In this project we have used this library for speech input.

2. OpenCV : OpenCV stands for **Open Source Computer Vision.** As a library of programming functions, the main aim of OpenCV is Real Time Computer Vision. It is developed by Intel.It is written in C++. This is a cross-platform library and is available for free to use. It is written in Python and has a size less than 200 MB. The latest stable release of OpenCV is 4.0.1. Deep Learning Frameworks like TensorFlow, Torch/PyTorch and Caffe are supported by OpenCV.

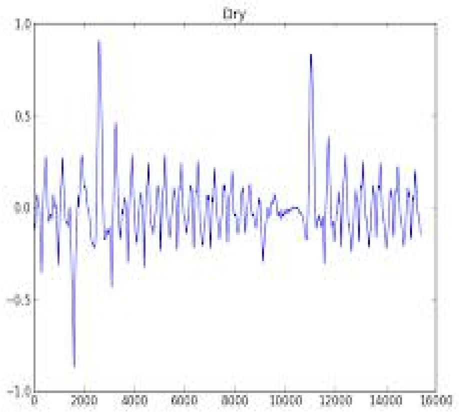
Application area of OpenCV includes Facial Recognition System, Gesture Recognition, Augmented Reality, Motion Tracking, Object Identification, Mobile Robotics and much more.

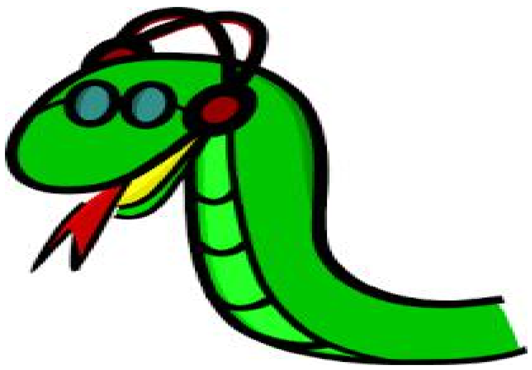


3. Numpy: Numpy is a core library and adds support for multi-dimensional arrays, matrices, high level mathematical functions for operation on arrays to the Python Programming Language. It is a cross platform library. It is an open source software The current stable version of NumPy is 1.16.3. It is an open source software.

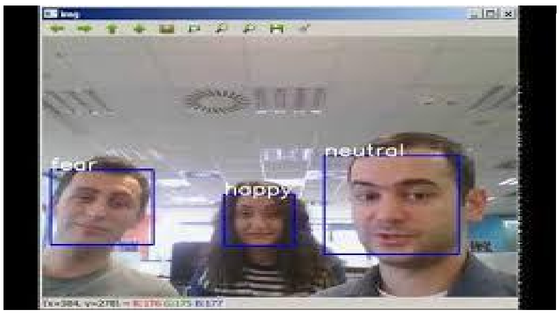
This module converts Python lists to numpy arrays as OpenCV face recognizer needs them for the face recognition process.

4. Pyaudio : PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.





5. **Tensorflow:** TensorFlow is developed by Google Brain Team. It was initially released in November 9, 2015. It is written in Python, C++ and CUDA. It is a machine learning open-source software library. It can be used for research as well as production. It is also used for Machine Learning Applications e.g. CNN (Convolutional Neural Network) and DNN (Deep Neural Network).Tensorflow framework will help train the model for Emotion Recognition



7.2 Hardware

1. Raspberry Pi :

The Raspberry Pi is a small single-board computers developed in United Kingdom via the Raspberry Pi Foundation.

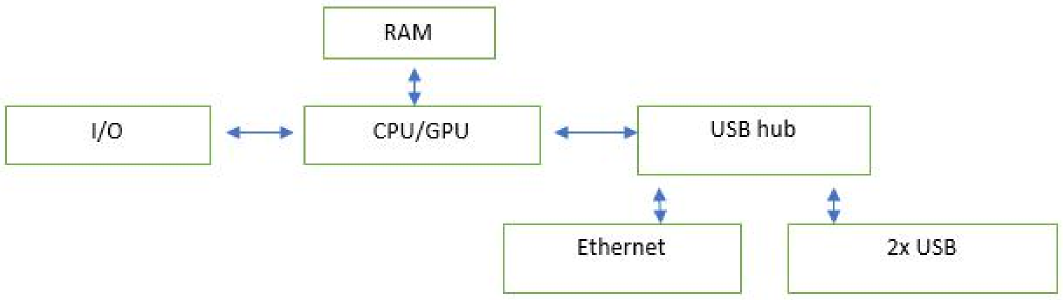
According to, Raspberry Pi foundation, more than five million Raspberry Pis were sold by February 2015, making it the fine-promoting British computer. By the end of November 2016 they had sold 11 million units, and 12.5m with the aid of March 2017, making it the third quality-promoting "General Purpose Computer". In July 2017, sales reached almost 15 million. In March 2018, sales reached 19 million. Several generations of Raspberry Pis have been released.

All models features a Broadcom machine on a chip (SoC) with an integrated ARM-well matched crucial processing unit (CPU) and on-chip pix processing unit (GPU).

It also provides a set of 40 GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).

Raspberry Pi Model 3B+

Block Diagram of Raspberry Pi Model B and B+



Family	Model	Form Factor	Ethernet	Wireless	GPIO	Released
Raspberry Pi-1	B	Standard (85.60 x 56.5mm)	Yes	No	26-pin	2012
	A		No			2013
	B+		Yes		40-pin	2014
	A+	Compact (65 x 56.5 mm)	No			2014
Raspberry Pi2	B	Standard	Yes	No		2015
Raspberry Pi Zero	Zero	Zero(65 x 30 mm)	No	No		2015
	W/WH			Yes		2017
Raspberry Pi 3	B	Standard	Yes	Yes		2016
	A+	Compact	No			2018
	B+	Standard	Yes			2018

Various Model Releases of Raspberry Pi and its specification.

3. USB mouse:-The Universal Serial Bus **(USB)** is an industry standard that was developed in the mid-1990s. The USB port provides power to the attached device hence eliminating the need for power connectors. The USB ports support hot plugging where you connect or remove device without turning off your computer.

We are using mouse is used to click the button to start the application.

3. USB Camera: A USB webcam is a camera that connects to a computer, usually through plugging it in to a USB port on the machine. The video is fed to the computer where a software application lets you view the pictures and also transfer them to the Internet. USB webcam software can also be used to provide a live feed directly from the computer to your website.

USB camera is used to capture the images of user's face expression.

4. Bluetooth mic: Bluetooth is a widely known and used technology for short-range voice transmission. **Bluetooth mic** is used to give audio input to the application.

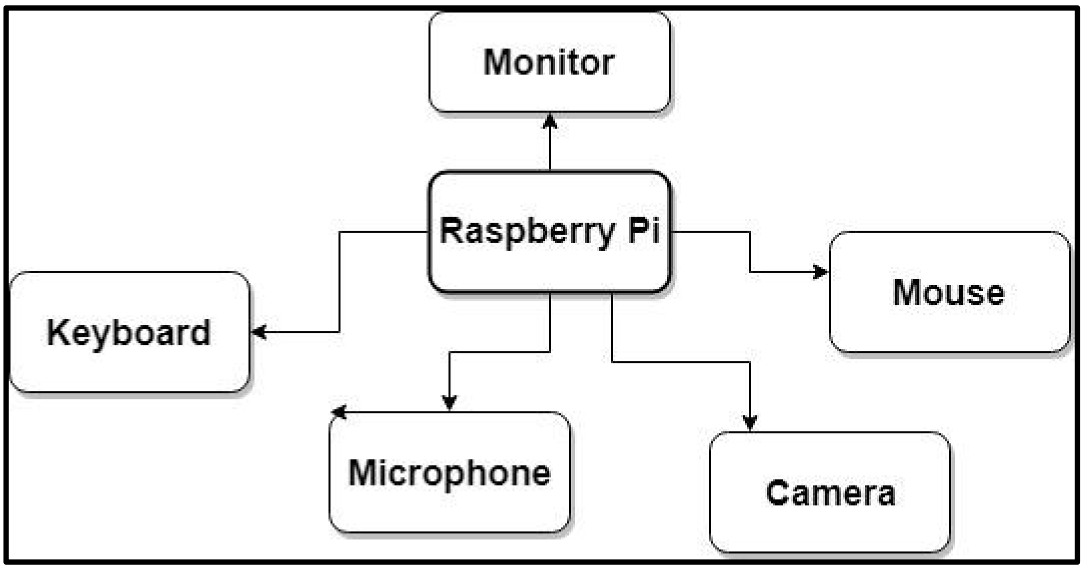
5. HDMI cable: HDMI stands for **High-Definition Multimedia Interface**. It is an audio and video interface used for transmitting video data(uncompressed) and digital audio data(uncompressed/compressed) from HDMI accommodating source device like display controller to a compatible Monitor, Projector, Television or Digital Audio Device.

It is used to connect the monitor with Raspberry pi as display screen.

6. **Monitor:** Monitor is a display device or output device that showcase information in form of pictures or frames. Monitors were earlier used for Data Processing only but nowadays single monitor can be used for many purposes.

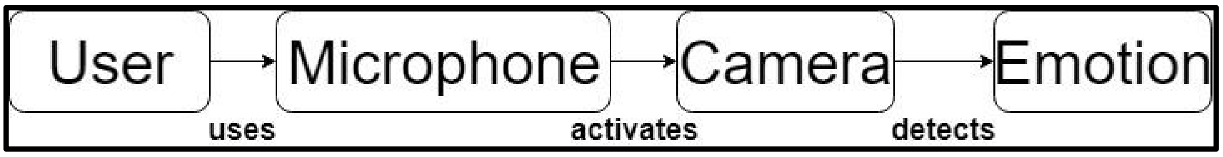
It is used as the output screen for Raspberry pi in our project

Chapter 8: SYSTEM DESIGN



The Raspberry Pi is the heart of our project, as it serves as the main connection point for all the hardware. We are using USB 2.0 Mouse, USB 2.0 Keyboard, USB 2.0 Camera and a wireless Bluetooth microphone along with a Monitor which we will connect using an HDMI cable.

8.1 Usage Process :



The end user will use the microphone to activate the camera through “start” command, after which the emotion is detected in real time within the camera frame.

8.2 Modules :

1. User Interface using Tkinter

Tkinter: Tkinter is the most commonly used Graphical User Interface. Python and Tkinter together provide fastest way to design GUI because Tkinter is a standard Python interface.

Steps to create Tkinter:

1. Import Tkinter Module
2. Creating the main window
3. Adding widgets to main window
4. Applying Event Trigger on the widgets.

We are using Tkinter to create a graphical user interface that takes in input from user, a button is used to enable mic:

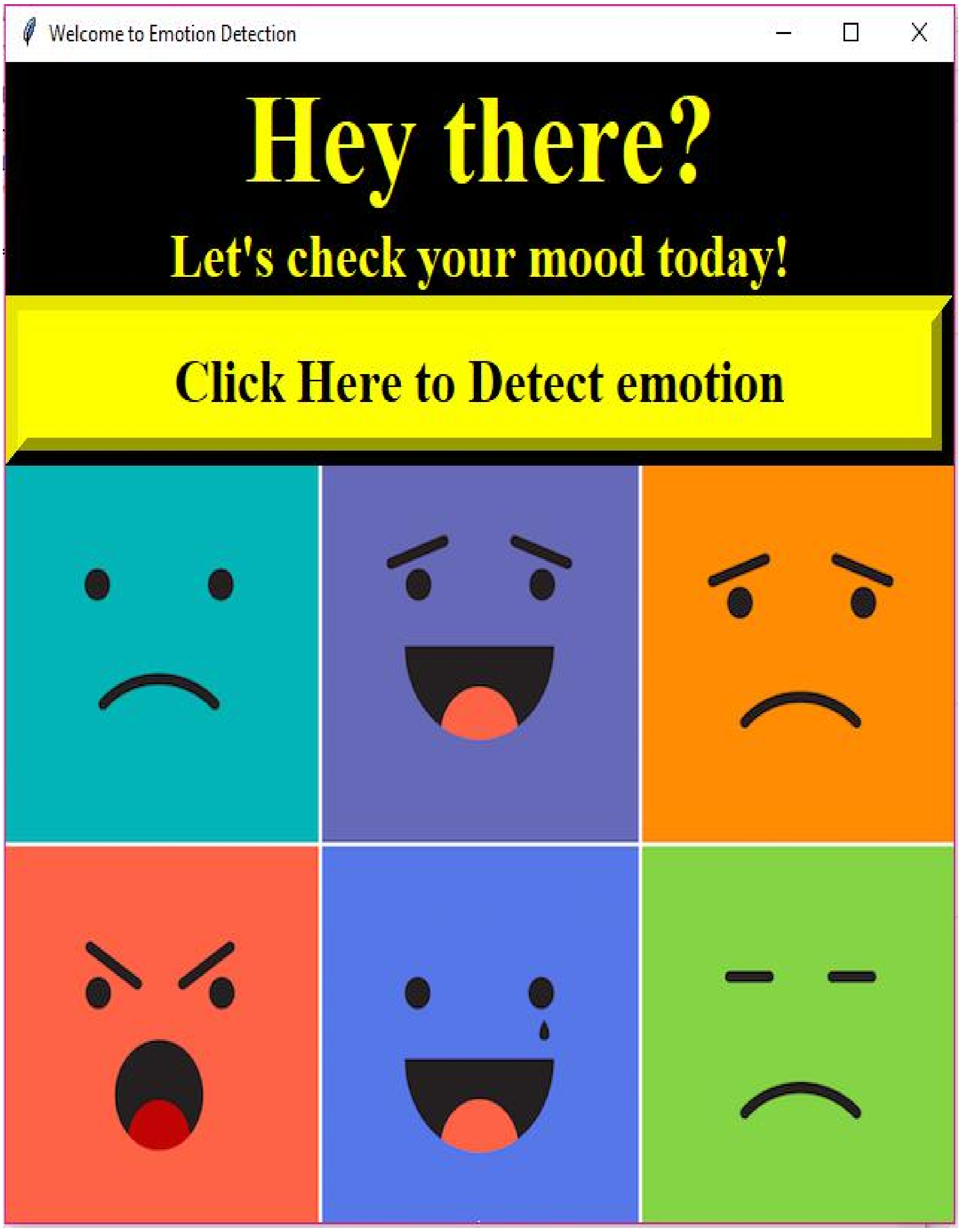
Widgets which we used in our Tkinter Application:

1. **Button:** To add a button in our application

Syntax: button = tk.button ( )

2. **Canvas**: To draw complex layout like graphics, widget, and text and draw picture.

Syntax: w = Canvas (width=40, height=20)


Welcome to Emotion Detection
Hey there?
Let's check your mood today!
Click Here to Detect emotion

Tkinter GUI for interacting with Emotion Detection Module

2. Audio Input Module

Using gTTs (Google text to speech) we have accessed Google API to identify voice and store the same in a string. We take in input as voice from the user as a command for Raspberry pi. So when the speech satisfies the condition, the camera module is started.

3. Camera Module

We are using camera module to capture emotion through live video detection. We are using USB 2.0 Web cam.

4. OpenCV HAAR CASCADES Module

We are using the "Frontal Face Alt" Classifier for detecting the presence of Face in the WebCam.

We load the xml file
classifier = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

5. Tensorflow Image Classifier Module

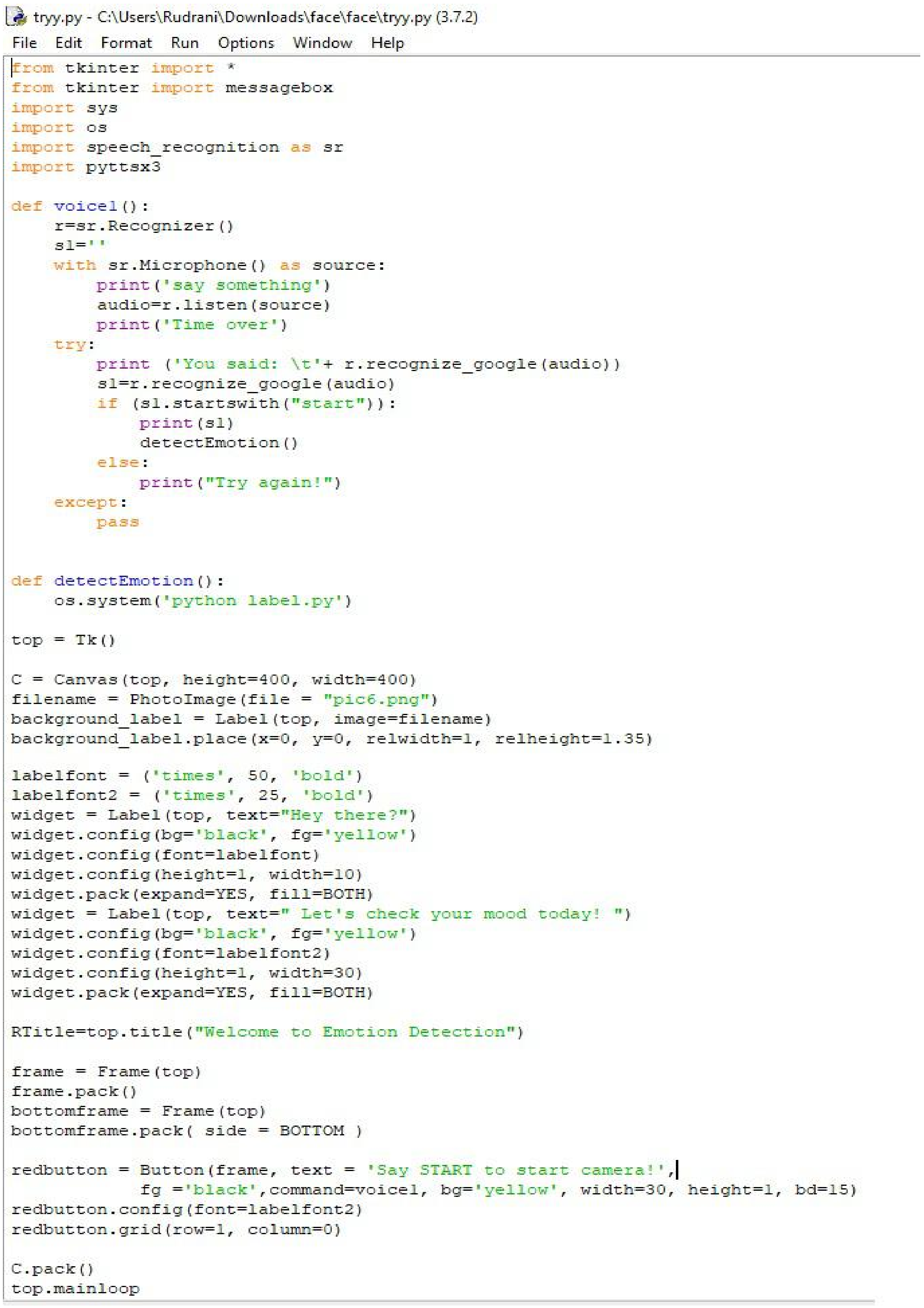
We are going to create an Image classifier that identifies a person's emotion and then show this text on the OpenCV Window. This step will consist of several sub steps:
First, we created a directory named images. In this directory, we created five sub directories with names like Happy, Sad, Angry, Surprise and Neutral.

Then we filled these directories with respective images by downloading them from the Internet. E.g. In "Happy" directory, we filled only those images of a person who is happy.
Then we run the "face-crop.py" program to align and crop all images that we downloaded
Then we retrain the network. For this purpose we are using Mobile net Model which is quite fast and accurate.
To run the training, open CMD/Terminal then open file location, and type the following:

```
python retrain.py --output_graph=retrained_graph.pb --
output_labels=retrained_labels.txt --architecture=MobileNet_1.0_224 --
image_dir=images
```

Chapter 9: Screenshots of Code

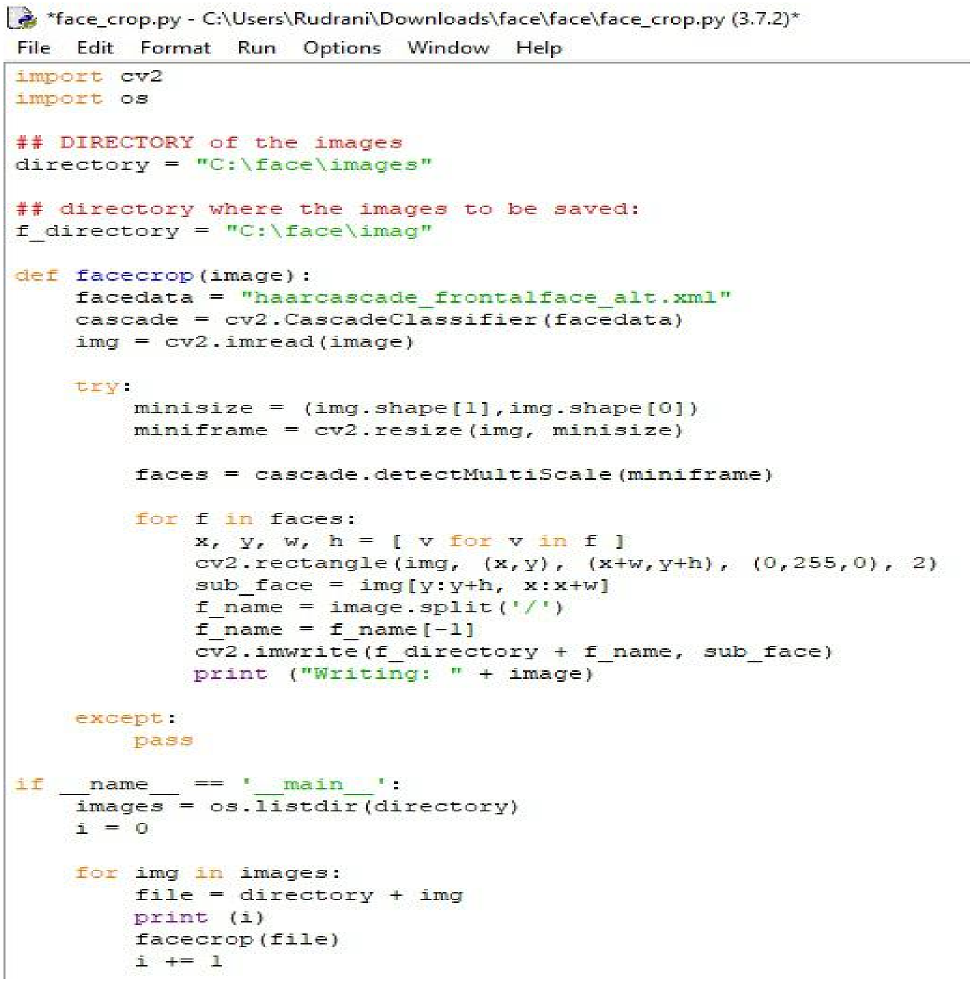
Front End User Interface:



tryy.py - C:\Users\Rudrani\Downloads\face\face\tryy.py (3.7.2)
File Edit Format Run Options Window Help

```python
from tkinter import *
from tkinter import messagebox
import sys
import os
import speech_recognition as sr
import pyttsx3

def voice1():
    r=sr.Recognizer()
    s1=''
    with sr.Microphone() as source:
        print('say something')
        audio=r.listen(source)
        print('Time over')
    try:
        print ('You said: \t'+ r.recognize_google(audio))
        s1=r.recognize_google(audio)
        if (s1.startswith("start")):
            print(s1)
            detectEmotion()
        else:
            print("Try again!")
    except:
        pass


def detectEmotion():
    os.system('python label.py')

top = Tk()

C = Canvas(top, height=400, width=400)
filename = PhotoImage(file = "pic6.png")
background_label = Label(top, image=filename)
background_label.place(x=0, y=0, relwidth=1, relheight=1.35)

labelfont = ('times', 50, 'bold')
labelfont2 = ('times', 25, 'bold')
widget = Label(top, text="Hey there?")
widget.config(bg='black', fg='yellow')
widget.config(font=labelfont)
widget.config(height=1, width=10)
widget.pack(expand=YES, fill=BOTH)
widget = Label(top, text=" Let's check your mood today! ")
widget.config(bg='black', fg='yellow')
widget.config(font=labelfont2)
widget.config(height=1, width=30)
widget.pack(expand=YES, fill=BOTH)

RTitle=top.title("Welcome to Emotion Detection")

frame = Frame(top)
frame.pack()
bottomframe = Frame(top)
bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text = 'Say START to start camera!',
            fg ='black',command=voice1, bg='yellow', width=30, height=1, bd=15)
redbutton.config(font=labelfont2)
redbutton.grid(row=1, column=0)

C.pack()
top.mainloop
```

Cropping downloaded Images :

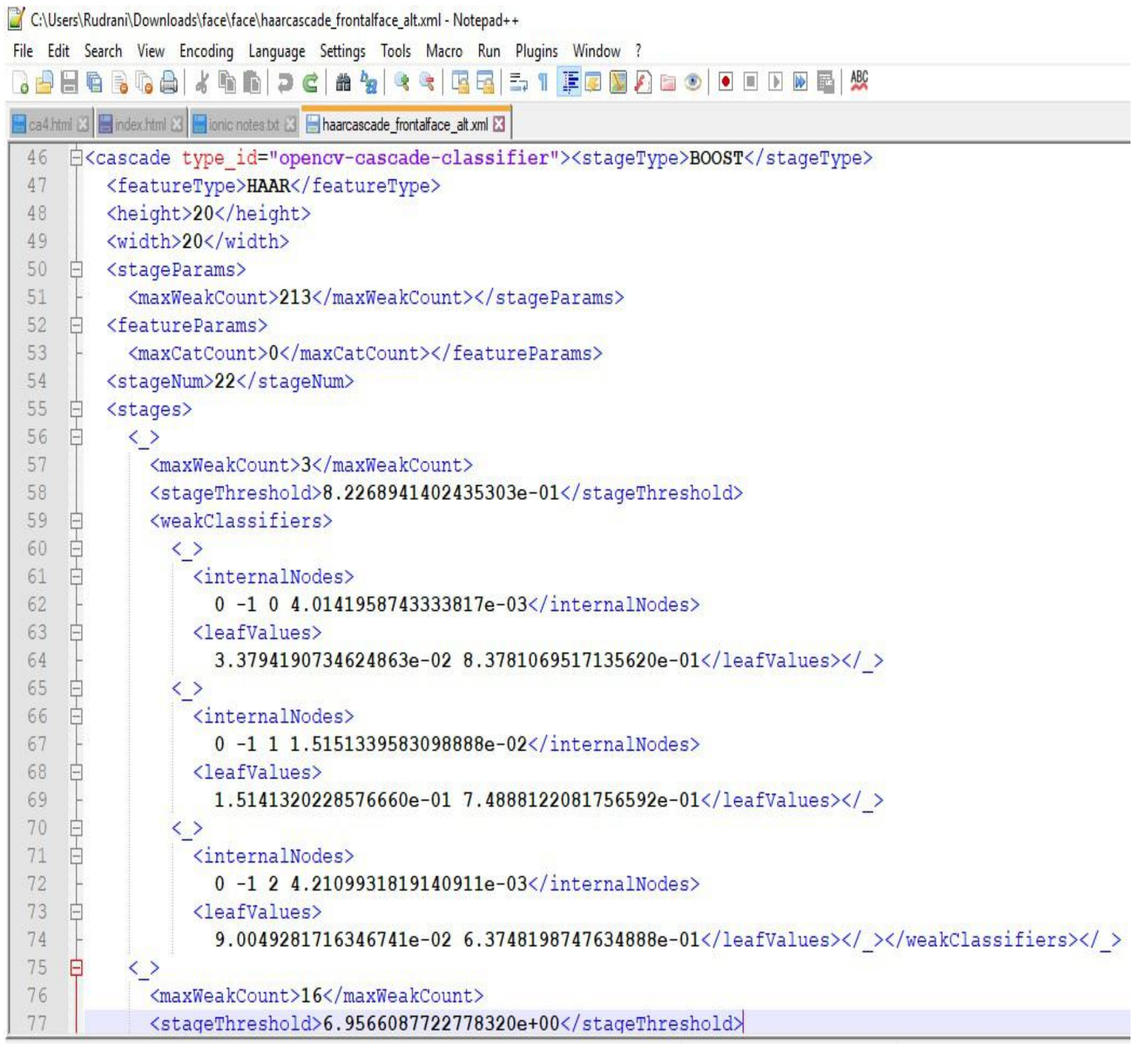


face_crop.py - C:\Users\Rudrani\Downloads\face\face\face_crop.py (3.7.2)

```python
import cv2
import os

## DIRECTORY of the images
directory = "C:\face\images"

## directory where the images to be saved:
f_directory = "C:\face\imag"

def facecrop(image):
    facedata = "haarcascade_frontalface_alt.xml"
    cascade = cv2.CascadeClassifier(facedata)
    img = cv2.imread(image)

    try:
        minisize = (img.shape[1],img.shape[0])
        miniframe = cv2.resize(img, minisize)

        faces = cascade.detectMultiScale(miniframe)

        for f in faces:
            x, y, w, h = [ v for v in f ]
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
            sub_face = img[y:y+h, x:x+w]
            f_name = image.split('/')
            f_name = f_name[-1]
            cv2.imwrite(f_directory + f_name, sub_face)
            print ("Writing: " + image)

    except:
        pass

if __name__ == '__main__':
    images = os.listdir(directory)
    i = 0

    for img in images:
        file = directory + img
        print (i)
        facecrop(file)
        i += 1
```

This is how pictures are cropped after running the above code:

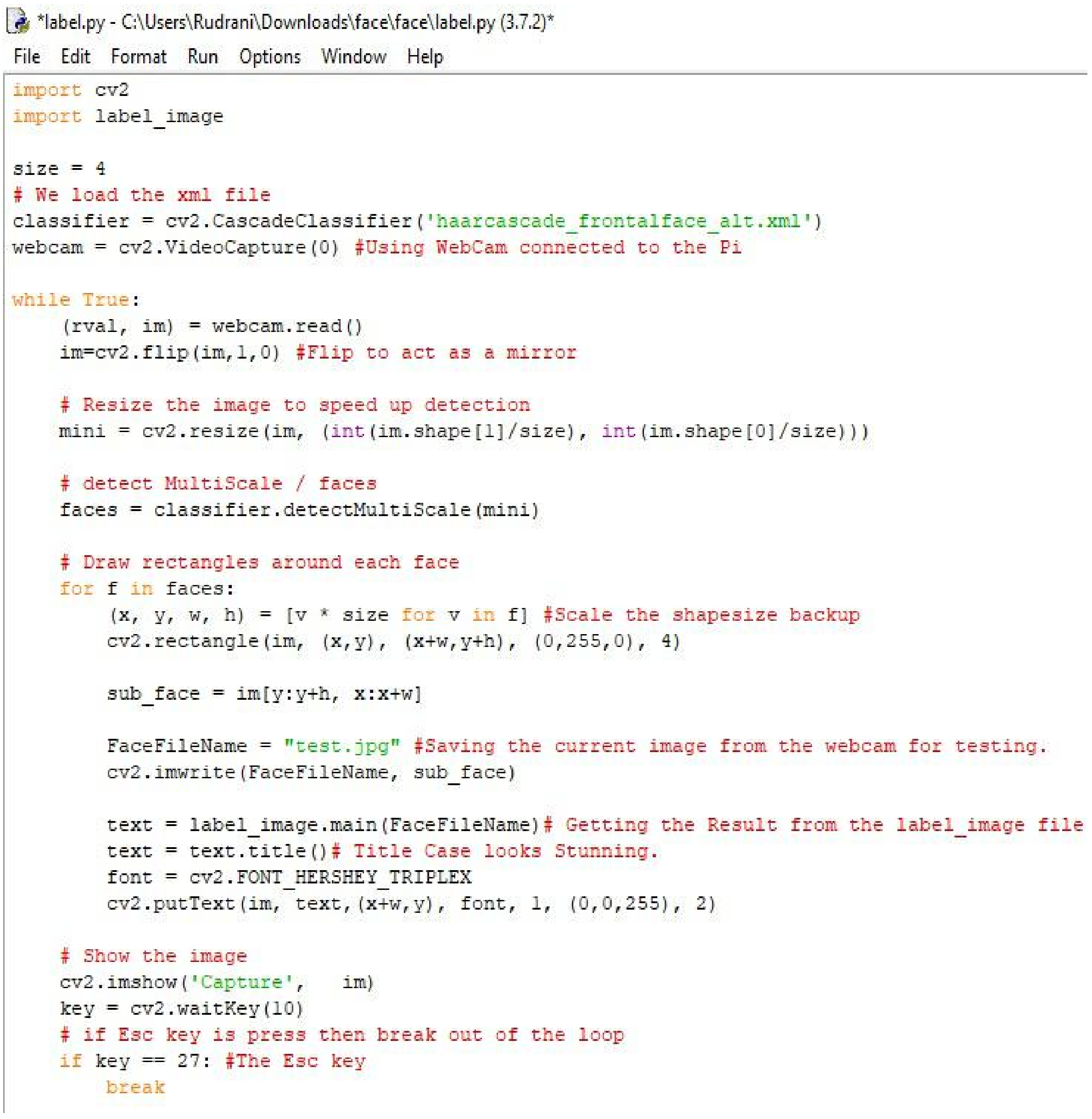
HAAR Cascades Image Classifier using OpenCv


C:\Users\Rudrani\Downloads\face\face\haarcascade_frontalface_alt.xml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ca4.html
index.html
ionic notes.txt
haarcascade_frontalface_alt.xml

```
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>20</height>
  <width>20</width>
  <stageParams>
    <maxWeakCount>213</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>22</stageNum>
  <stages>
    <_>
      <maxWeakCount>3</maxWeakCount>
      <stageThreshold>8.2268941402435303e-01</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 4.0141958743333817e-03</internalNodes>
          <leafValues>
            3.3794190734624863e-02 8.3781069517135620e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.5151339583098888e-02</internalNodes>
          <leafValues>
            1.5141320228576660e-01 7.4888122081756592e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 4.2109931819140911e-03</internalNodes>
          <leafValues>
            9.0049281716346741e-02 6.3748198747634888e-01</leafValues></_></weakClassifiers></_>
    <_>
      <maxWeakCount>16</maxWeakCount>
      <stageThreshold>6.9566087722778320e+00</stageThreshold>
```

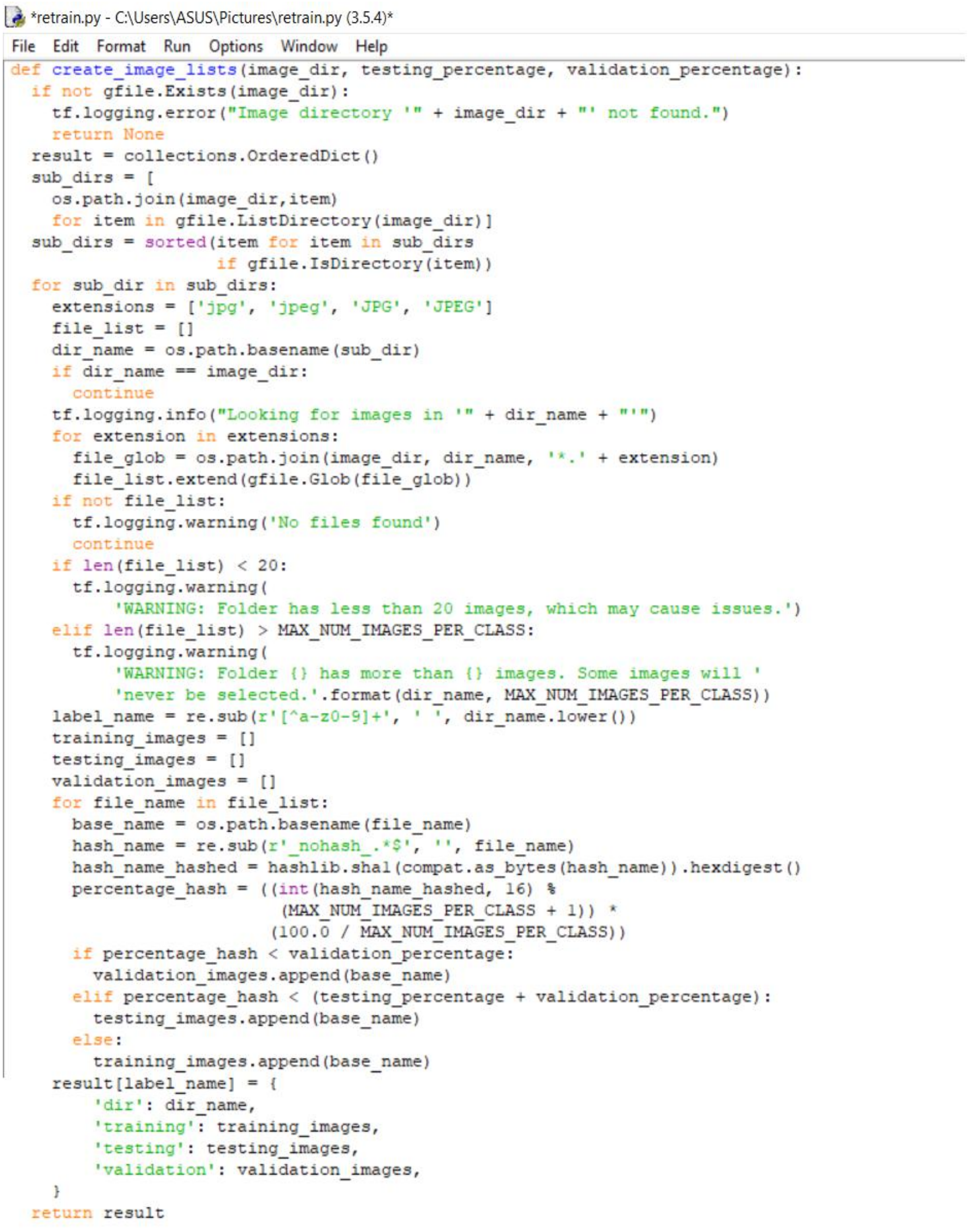
A Haar Cascade is basically a classifier which is used to detect particular objects from the source. The **haarcascade_frontalface_default.xml** is a haar cascade designed by OpenCV to detect the frontal face.

Code for Detecting Emotion after Re-Scaling image from video in real time


label.py - C:\Users\Rudrani\Downloads\face\face\label.py (3.7.2)
File Edit Format Run Options Window Help

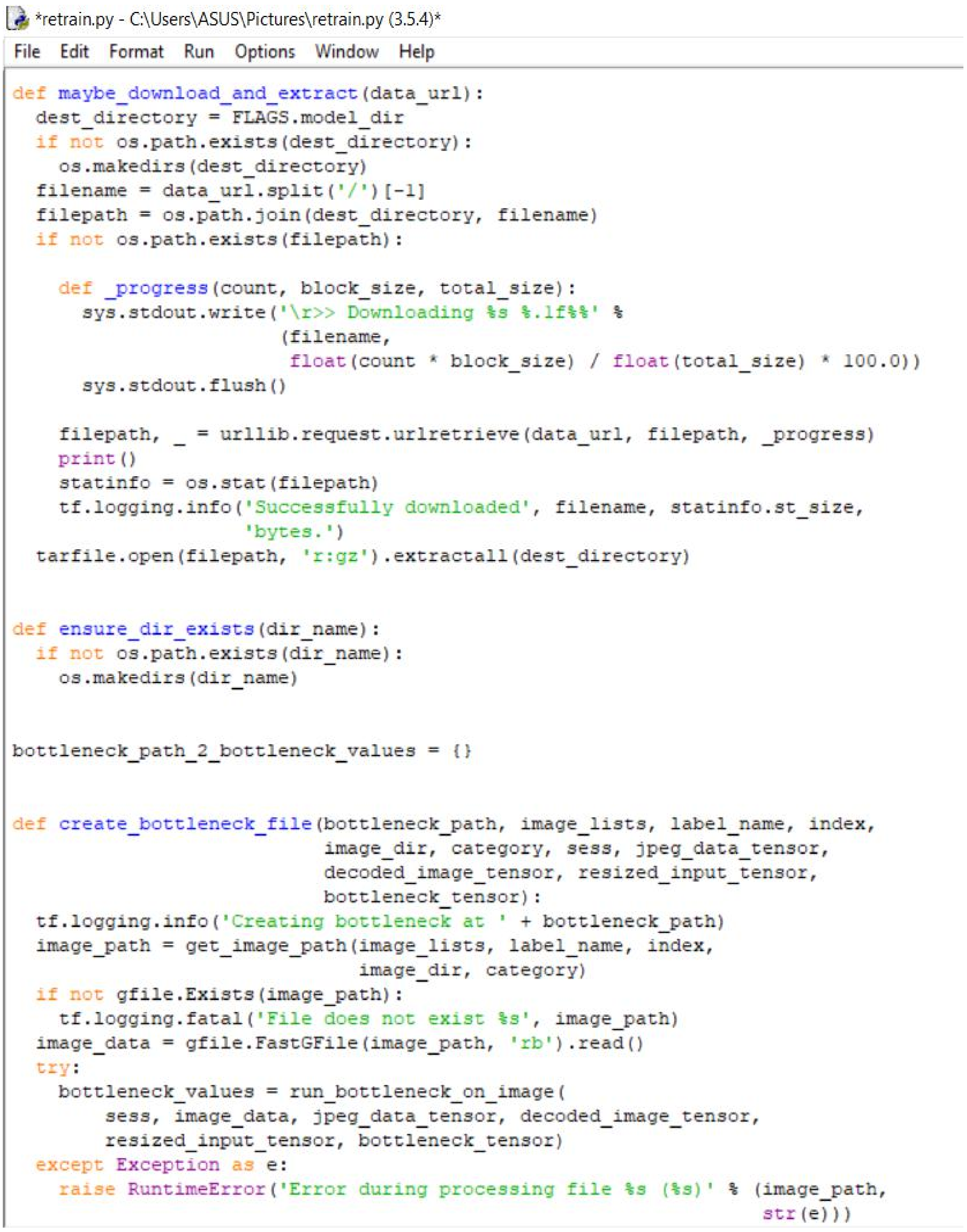
```
import cv2
import label_image

size = 4
# We load the xml file
classifier = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
webcam = cv2.VideoCapture(0) #Using WebCam connected to the Pi

while True:
    (rval, im) = webcam.read()
    im=cv2.flip(im,1,0) #Flip to act as a mirror

    # Resize the image to speed up detection
    mini = cv2.resize(im, (int(im.shape[1]/size), int(im.shape[0]/size)))

    # detect MultiScale / faces
    faces = classifier.detectMultiScale(mini)

    # Draw rectangles around each face
    for f in faces:
        (x, y, w, h) = [v * size for v in f] #Scale the shapesize backup
        cv2.rectangle(im, (x,y), (x+w,y+h), (0,255,0), 4)

        sub_face = im[y:y+h, x:x+w]

        FaceFileName = "test.jpg" #Saving the current image from the webcam for testing.
        cv2.imwrite(FaceFileName, sub_face)

        text = label_image.main(FaceFileName)# Getting the Result from the label_image file
        text = text.title()# Title Case looks Stunning.
        font = cv2.FONT_HERSHEY_TRIPLEX
        cv2.putText(im, text,(x+w,y), font, 1, (0,0,255), 2)

    # Show the image
    cv2.imshow('Capture',   im)
    key = cv2.waitKey(10)
    # if Esc key is press then break out of the loop
    if key == 27: #The Esc key
        break
```

Code to train data-set


retrain.py - C:\Users\ASUS\Pictures\retrain.py (3.5.4)
File Edit Format Run Options Window Help

```
def create_image_lists(image_dir, testing_percentage, validation_percentage):
  if not gfile.Exists(image_dir):
    tf.logging.error("Image directory '" + image_dir + "' not found.")
    return None
  result = collections.OrderedDict()
  sub_dirs = [
    os.path.join(image_dir,item)
    for item in gfile.ListDirectory(image_dir)]
  sub_dirs = sorted(item for item in sub_dirs
                    if gfile.IsDirectory(item))
  for sub_dir in sub_dirs:
    extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
    file_list = []
    dir_name = os.path.basename(sub_dir)
    if dir_name == image_dir:
      continue
    tf.logging.info("Looking for images in '" + dir_name + "'")
    for extension in extensions:
      file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
      file_list.extend(gfile.Glob(file_glob))
    if not file_list:
      tf.logging.warning('No files found')
      continue
    if len(file_list) < 20:
      tf.logging.warning(
          'WARNING: Folder has less than 20 images, which may cause issues.')
    elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
      tf.logging.warning(
          'WARNING: Folder {} has more than {} images. Some images will '
          'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
    label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
    training_images = []
    testing_images = []
    validation_images = []
    for file_name in file_list:
      base_name = os.path.basename(file_name)
      hash_name = re.sub(r'_nohash_.*$', '', file_name)
      hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
      percentage_hash = ((int(hash_name_hashed, 16) %
                          (MAX_NUM_IMAGES_PER_CLASS + 1)) *
                         (100.0 / MAX_NUM_IMAGES_PER_CLASS))
      if percentage_hash < validation_percentage:
        validation_images.append(base_name)
      elif percentage_hash < (testing_percentage + validation_percentage):
        testing_images.append(base_name)
      else:
        training_images.append(base_name)
    result[label_name] = {
        'dir': dir_name,
        'training': training_images,
        'testing': testing_images,
        'validation': validation_images,
    }
  return result
```

```
def get_image_path(image_lists, label_name, index, image_dir, category):
  if label_name not in image_lists:
    tf.logging.fatal('Label does not exist %s.', label_name)
  label_lists = image_lists[label_name]
  if category not in label_lists:
    tf.logging.fatal('Category does not exist %s.', category)
  category_list = label_lists[category]
  if not category_list:
    tf.logging.fatal('Label %s has no images in the category %s.',
                     label_name, category)
  mod_index = index % len(category_list)
  base_name = category_list[mod_index]
  sub_dir = label_lists['dir']
  full_path = os.path.join(image_dir, sub_dir, base_name)
  return full_path

def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
                        category, architecture):
  return get_image_path(image_lists, label_name, index, bottleneck_dir,
                        category) + '_' + architecture + '.txt'


def create_model_graph(model_info):
  with tf.Graph().as_default() as graph:
    model_path = os.path.join(FLAGS.model_dir, model_info['model_file_name'])
    with gfile.FastGFile(model_path, 'rb') as f:
      graph_def = tf.GraphDef()
      graph_def.ParseFromString(f.read())
      bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(
          graph_def,
          name='',
          return_elements=[
              model_info['bottleneck_tensor_name'],
              model_info['resized_input_tensor_name'],
          ]))
  return graph, bottleneck_tensor, resized_input_tensor


def run_bottleneck_on_image(sess, image_data, image_data_tensor,
                            decoded_image_tensor, resized_input_tensor,
                            bottleneck_tensor):
  resized_input_values = sess.run(decoded_image_tensor,
                                  {image_data_tensor: image_data})
  bottleneck_values = sess.run(bottleneck_tensor,
                               {resized_input_tensor: resized_input_values})
  bottleneck_values = np.squeeze(bottleneck_values)
  return bottleneck_values
```

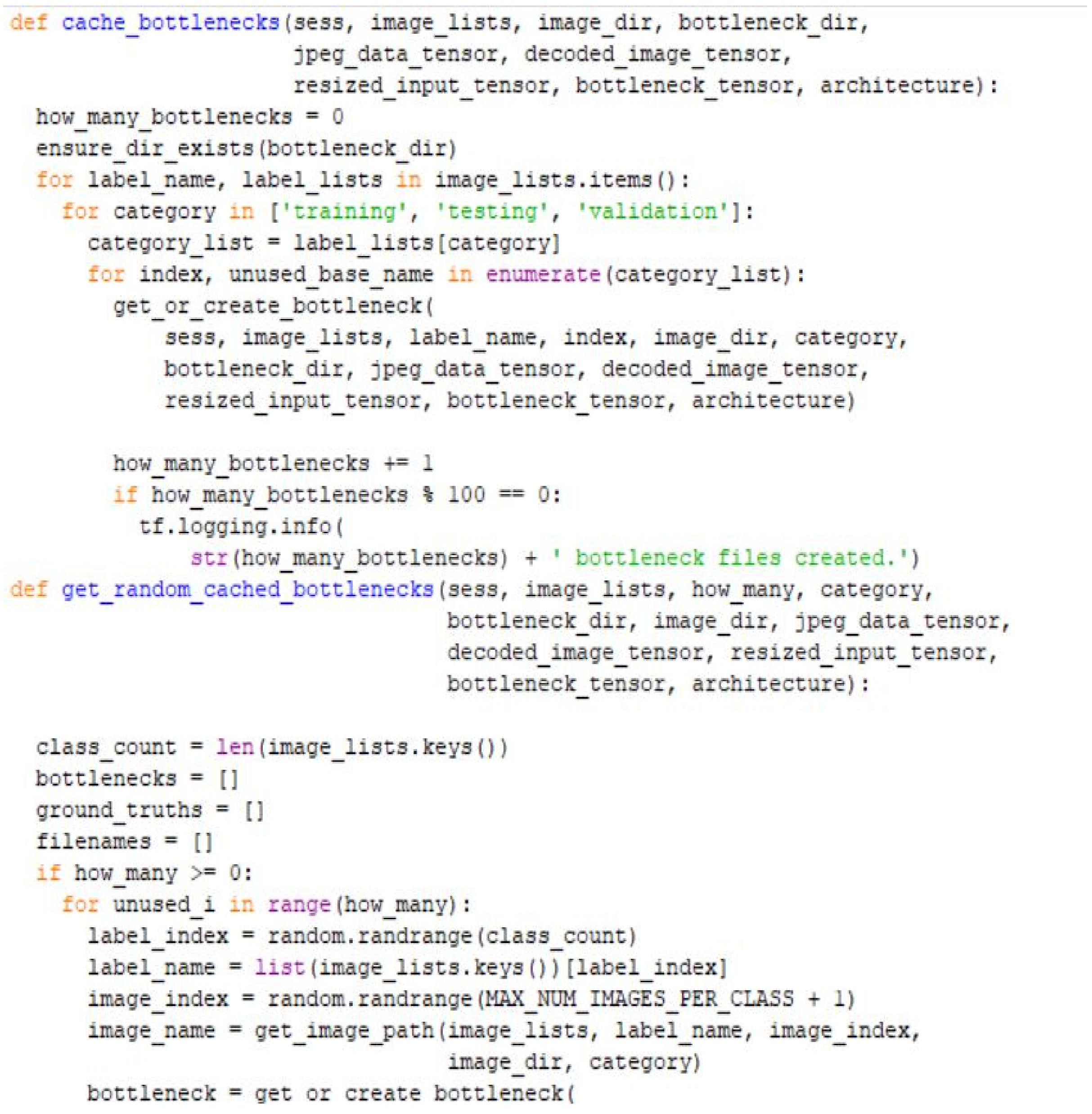

retrain.py - C:\Users\ASUS\Pictures\retrain.py (3.5.4)

File Edit Format Run Options Window Help

```
def maybe_download_and_extract(data_url):
  dest_directory = FLAGS.model_dir
  if not os.path.exists(dest_directory):
    os.makedirs(dest_directory)
  filename = data_url.split('/')[-1]
  filepath = os.path.join(dest_directory, filename)
  if not os.path.exists(filepath):

    def _progress(count, block_size, total_size):
      sys.stdout.write('\r>> Downloading %s %.1f%%' %
                       (filename,
                        float(count * block_size) / float(total_size) * 100.0))
      sys.stdout.flush()

    filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
    print()
    statinfo = os.stat(filepath)
    tf.logging.info('Successfully downloaded', filename, statinfo.st_size,
                    'bytes.')
  tarfile.open(filepath, 'r:gz').extractall(dest_directory)


def ensure_dir_exists(dir_name):
  if not os.path.exists(dir_name):
    os.makedirs(dir_name)


bottleneck_path_2_bottleneck_values = {}


def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor):
  tf.logging.info('Creating bottleneck at ' + bottleneck_path)
  image_path = get_image_path(image_lists, label_name, index,
                              image_dir, category)
  if not gfile.Exists(image_path):
    tf.logging.fatal('File does not exist %s', image_path)
  image_data = gfile.FastGFile(image_path, 'rb').read()
  try:
    bottleneck_values = run_bottleneck_on_image(
        sess, image_data, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor)
  except Exception as e:
    raise RuntimeError('Error during processing file %s (%s)' % (image_path,
                                                                 str(e)))
```

```
def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,
                             category, bottleneck_dir, jpeg_data_tensor,
                             decoded_image_tensor, resized_input_tensor,
                             bottleneck_tensor, architecture):
  label_lists = image_lists[label_name]
  sub_dir = label_lists['dir']
  sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
  ensure_dir_exists(sub_dir_path)
  bottleneck_path = get_bottleneck_path(image_lists, label_name, index,
                                        bottleneck_dir, category, architecture)
  if not os.path.exists(bottleneck_path):
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor)
  with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
  did_hit_error = False
  try:
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
  except ValueError:
    tf.logging.warning('Invalid float found, recreating bottleneck')
    did_hit_error = True
  if did_hit_error:
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor)
    with open(bottleneck_path, 'r') as bottleneck_file:
      bottleneck_string = bottleneck_file.read()
    # Allow exceptions to propagate here, since they shouldn't happen after a
    # fresh creation
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
  return bottleneck_values
```

```
      bottleneck = get_or_create_bottleneck(
          sess, image_lists, label_name, image_index, image_dir, category,
          bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
          resized_input_tensor, bottleneck_tensor, architecture)
      ground_truth = np.zeros(class_count, dtype=np.float32)
      ground_truth[label_index] = 1.0
      bottlenecks.append(bottleneck)
      ground_truths.append(ground_truth)
      filenames.append(image_name)
  return bottlenecks, ground_truths, filenames


def get_random_distorted_bottlenecks(
    sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
    distorted_image, resized_input_tensor, bottleneck_tensor):
  class_count = len(image_lists.keys())
  bottlenecks = []
  ground_truths = []
  for unused_i in range(how_many):
    label_index = random.randrange(class_count)
    label_name = list(image_lists.keys())[label_index]
    image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
    image_path = get_image_path(image_lists, label_name, image_index, image_dir,
                                category)
    if not gfile.Exists(image_path):
      tf.logging.fatal('File does not exist %s', image_path)
    jpeg_data = gfile.FastGFile(image_path, 'rb').read()
    # Note that we materialize the distorted_image_data as a numpy array before
    # sending running inference on the image. This involves 2 memory copies and
    # might be optimized in other implementations.
    distorted_image_data = sess.run(distorted_image,
                                    {input_jpeg_tensor: jpeg_data})
    bottleneck_values = sess.run(bottleneck_tensor,
                                 {resized_input_tensor: distorted_image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
    bottlenecks.append(bottleneck_values)
    ground_truths.append(ground_truth)
  return bottlenecks, ground_truths
```



```
def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir,
                      jpeg_data_tensor, decoded_image_tensor,
                      resized_input_tensor, bottleneck_tensor, architecture):
  how_many_bottlenecks = 0
  ensure_dir_exists(bottleneck_dir)
  for label_name, label_lists in image_lists.items():
    for category in ['training', 'testing', 'validation']:
      category_list = label_lists[category]
      for index, unused_base_name in enumerate(category_list):
        get_or_create_bottleneck(
            sess, image_lists, label_name, index, image_dir, category,
            bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
            resized_input_tensor, bottleneck_tensor, architecture)

        how_many_bottlenecks += 1
        if how_many_bottlenecks % 100 == 0:
          tf.logging.info(
              str(how_many_bottlenecks) + ' bottleneck files created.')
def get_random_cached_bottlenecks(sess, image_lists, how_many, category,
                                  bottleneck_dir, image_dir, jpeg_data_tensor,
                                  decoded_image_tensor, resized_input_tensor,
                                  bottleneck_tensor, architecture):

  class_count = len(image_lists.keys())
  bottlenecks = []
  ground_truths = []
  filenames = []
  if how_many >= 0:
    for unused_i in range(how_many):
      label_index = random.randrange(class_count)
      label_name = list(image_lists.keys())[label_index]
      image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
      image_name = get_image_path(image_lists, label_name, image_index,
                                  image_dir, category)
      bottleneck = get_or_create_bottleneck(
        bottleneck = get_or_create_bottleneck(
            sess, image_lists, label_name, image_index, image_dir, category,
            bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
            resized_input_tensor, bottleneck_tensor, architecture)
        ground_truth = np.zeros(class_count, dtype=np.float32)
        ground_truth[label_index] = 1.0
        bottlenecks.append(bottleneck)
        ground_truths.append(ground_truth)
        filenames.append(image_name)
  return bottlenecks, ground_truths, filenames


def get_random_distorted_bottlenecks(
    sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
    distorted_image, resized_input_tensor, bottleneck_tensor):
  class_count = len(image_lists.keys())
  bottlenecks = []
  ground_truths = []
  for unused_i in range(how_many):
    label_index = random.randrange(class_count)
    label_name = list(image_lists.keys())[label_index]
    image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
    image_path = get_image_path(image_lists, label_name, image_index, image_dir,
                                category)
    if not gfile.Exists(image_path):
      tf.logging.fatal('File does not exist %s', image_path)
    jpeg_data = gfile.FastGFile(image_path, 'rb').read()
    # Note that we materialize the distorted_image_data as a numpy array before
    # sending running inference on the image. This involves 2 memory copies and
    # might be optimized in other implementations.
    distorted_image_data = sess.run(distorted_image,
                                    {input_jpeg_tensor: jpeg_data})
    bottleneck_values = sess.run(bottleneck_tensor,
                                 {resized_input_tensor: distorted_image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
    bottlenecks.append(bottleneck_values)
    ground_truths.append(ground_truth)
  return bottlenecks, ground_truths
```

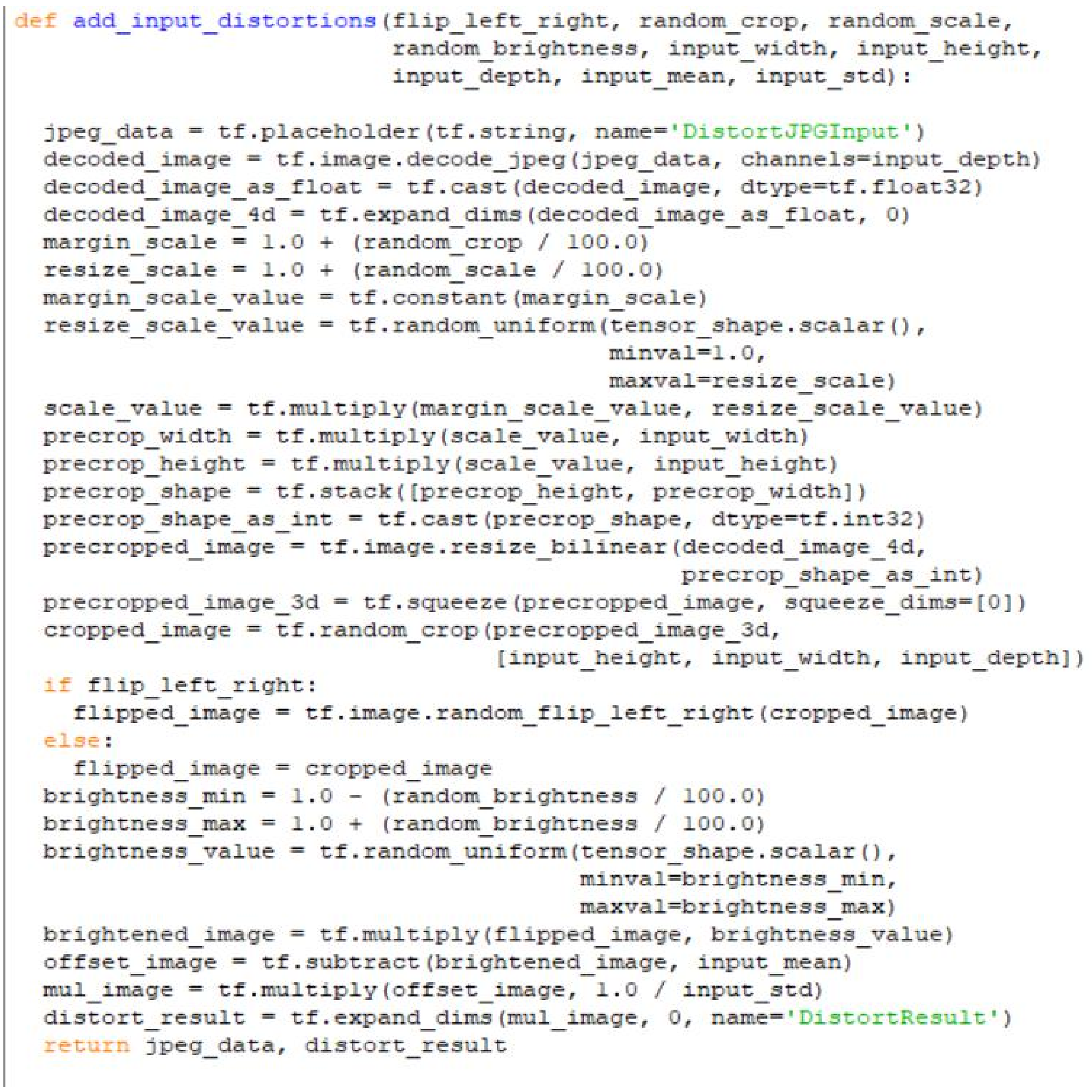
Preliminary Steps for Cropping and Scaling:-

1. Cropping

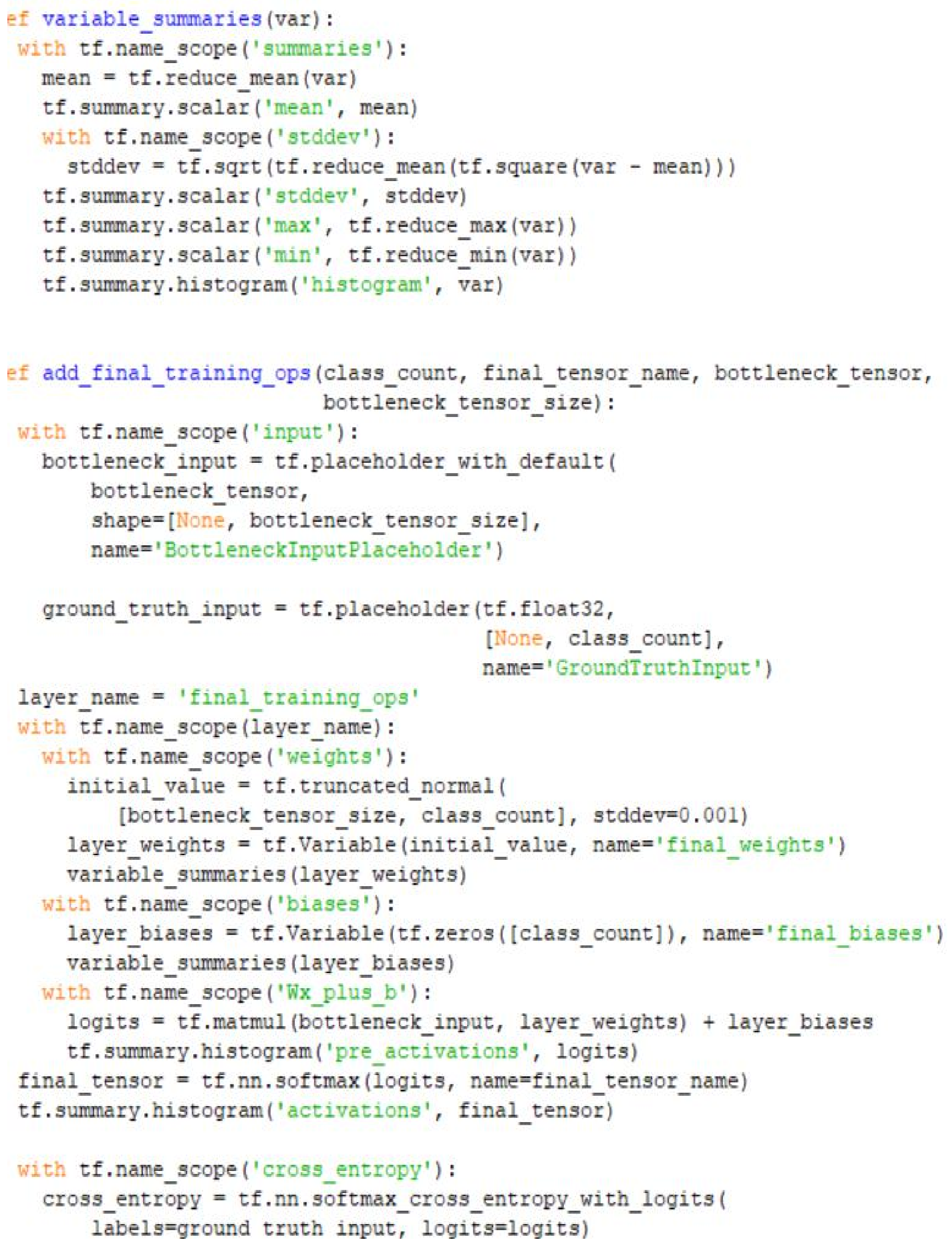
Cropping is achieved by placing a bounded square box at a random position in the full frame of the camera capturing the image. The cropping parameter controls the size of that box relative to the input processed image. If it's zero, then the box is the same size as the input and no cropping is performed. If the value is 50%, then the crop box will be half the width and height of the input.

2. Scaling

Scaling is similar to cropping, except that the bounding box is centered and its size varies randomly within the given range. For example if the scale percentage is zero, then the bounding box is the same size as the input and no scaling is applied. If it's 50%, then the bounding box will be in a random range between half the width and height and full size.

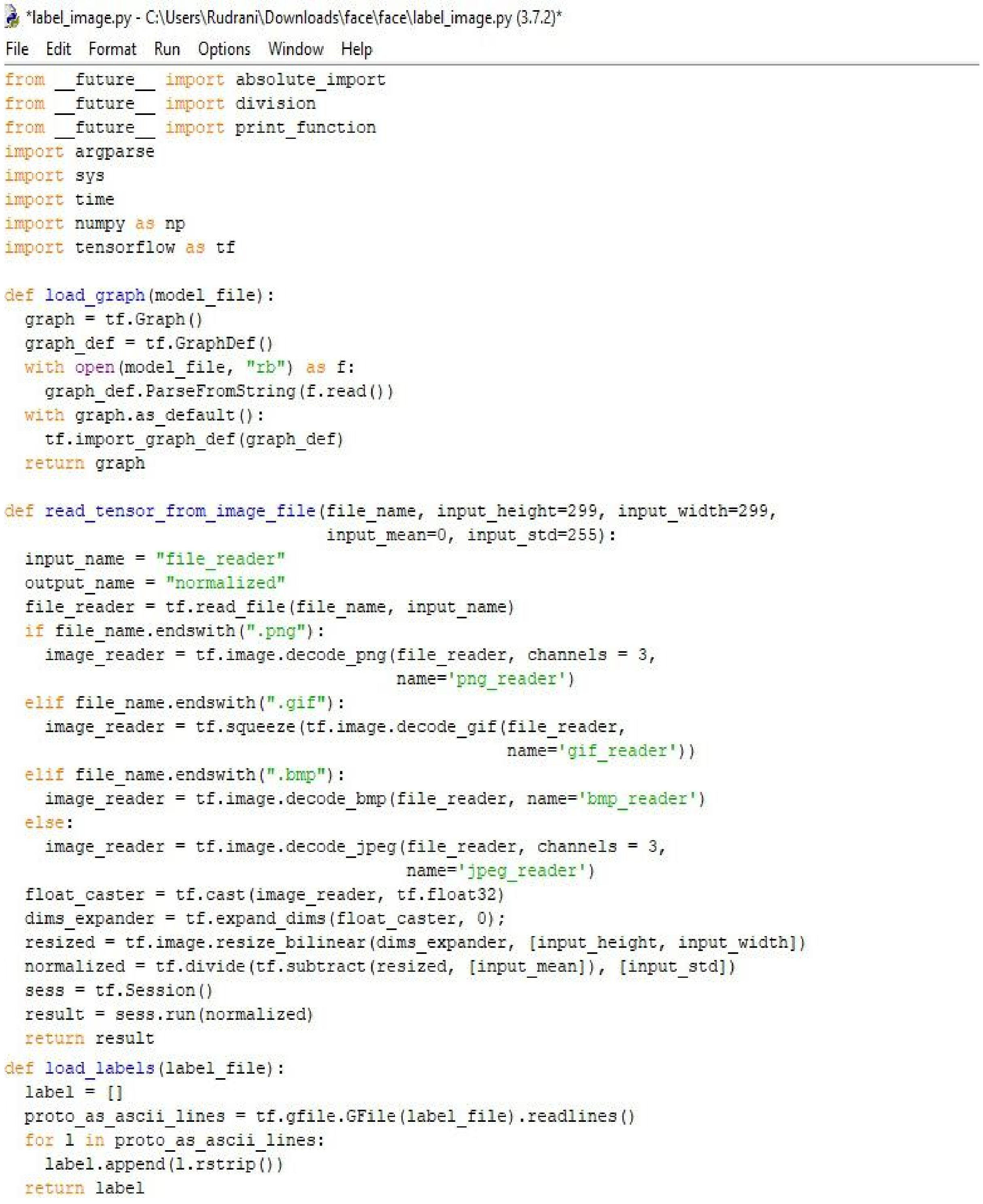


```
def add_input_distortions(flip_left_right, random_crop, random_scale,
                          random_brightness, input_width, input_height,
                          input_depth, input_mean, input_std):

  jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
  decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
  decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
  decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
  margin_scale = 1.0 + (random_crop / 100.0)
  resize_scale = 1.0 + (random_scale / 100.0)
  margin_scale_value = tf.constant(margin_scale)
  resize_scale_value = tf.random_uniform(tensor_shape.scalar(),
                                         minval=1.0,
                                         maxval=resize_scale)
  scale_value = tf.multiply(margin_scale_value, resize_scale_value)
  precrop_width = tf.multiply(scale_value, input_width)
  precrop_height = tf.multiply(scale_value, input_height)
  precrop_shape = tf.stack([precrop_height, precrop_width])
  precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
  precropped_image = tf.image.resize_bilinear(decoded_image_4d,
                                              precrop_shape_as_int)
  precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
  cropped_image = tf.random_crop(precropped_image_3d,
                                 [input_height, input_width, input_depth])
  if flip_left_right:
    flipped_image = tf.image.random_flip_left_right(cropped_image)
  else:
    flipped_image = cropped_image
  brightness_min = 1.0 - (random_brightness / 100.0)
  brightness_max = 1.0 + (random_brightness / 100.0)
  brightness_value = tf.random_uniform(tensor_shape.scalar(),
                                       minval=brightness_min,
                                       maxval=brightness_max)
  brightened_image = tf.multiply(flipped_image, brightness_value)
  offset_image = tf.subtract(brightened_image, input_mean)
  mul_image = tf.multiply(offset_image, 1.0 / input_std)
  distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
  return jpeg_data, distort_result
```



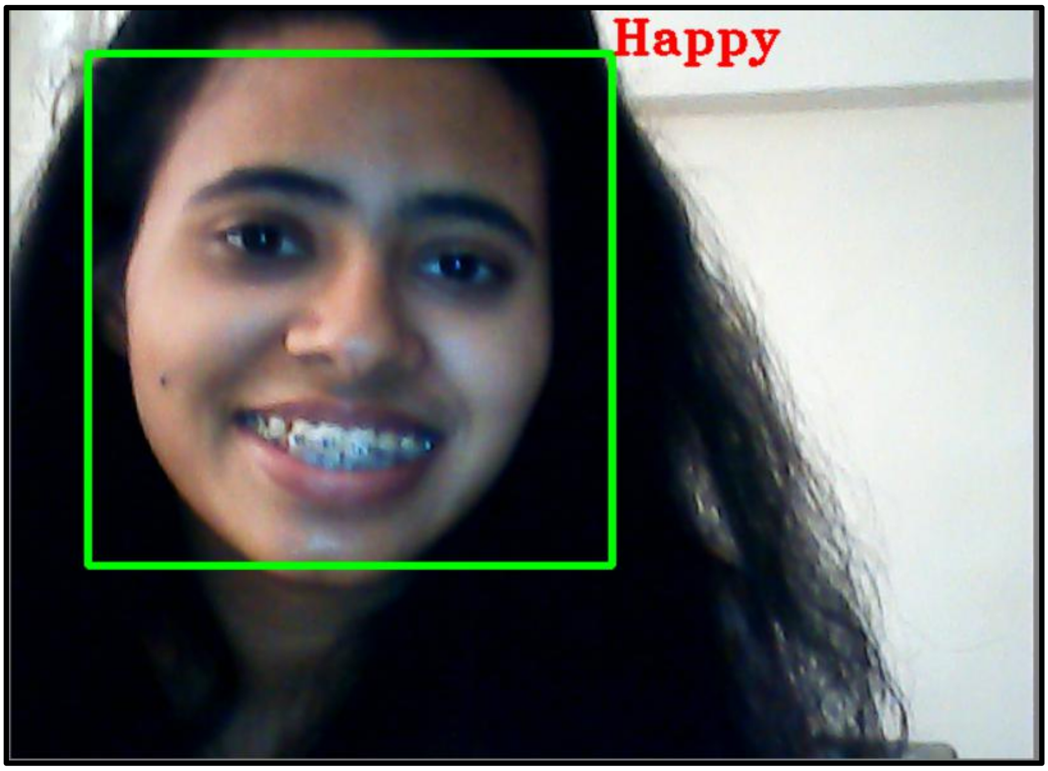
```
ef variable_summaries(var):
 with tf.name_scope('summaries'):
   mean = tf.reduce_mean(var)
   tf.summary.scalar('mean', mean)
   with tf.name_scope('stddev'):
     stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
   tf.summary.scalar('stddev', stddev)
   tf.summary.scalar('max', tf.reduce_max(var))
   tf.summary.scalar('min', tf.reduce_min(var))
   tf.summary.histogram('histogram', var)


ef add_final_training_ops(class_count, final_tensor_name, bottleneck_tensor,
                          bottleneck_tensor_size):
 with tf.name_scope('input'):
   bottleneck_input = tf.placeholder_with_default(
       bottleneck_tensor,
       shape=[None, bottleneck_tensor_size],
       name='BottleneckInputPlaceholder')

   ground_truth_input = tf.placeholder(tf.float32,
                                       [None, class_count],
                                       name='GroundTruthInput')
 layer_name = 'final_training_ops'
 with tf.name_scope(layer_name):
   with tf.name_scope('weights'):
     initial_value = tf.truncated_normal(
         [bottleneck_tensor_size, class_count], stddev=0.001)
     layer_weights = tf.Variable(initial_value, name='final_weights')
     variable_summaries(layer_weights)
   with tf.name_scope('biases'):
     layer_biases = tf.Variable(tf.zeros([class_count]), name='final_biases')
     variable_summaries(layer_biases)
   with tf.name_scope('Wx_plus_b'):
     logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
     tf.summary.histogram('pre_activations', logits)
 final_tensor = tf.nn.softmax(logits, name=final_tensor_name)
 tf.summary.histogram('activations', final_tensor)

 with tf.name_scope('cross_entropy'):
   cross_entropy = tf.nn.softmax_cross_entropy_with_logits(
       labels=ground_truth_input, logits=logits)
```

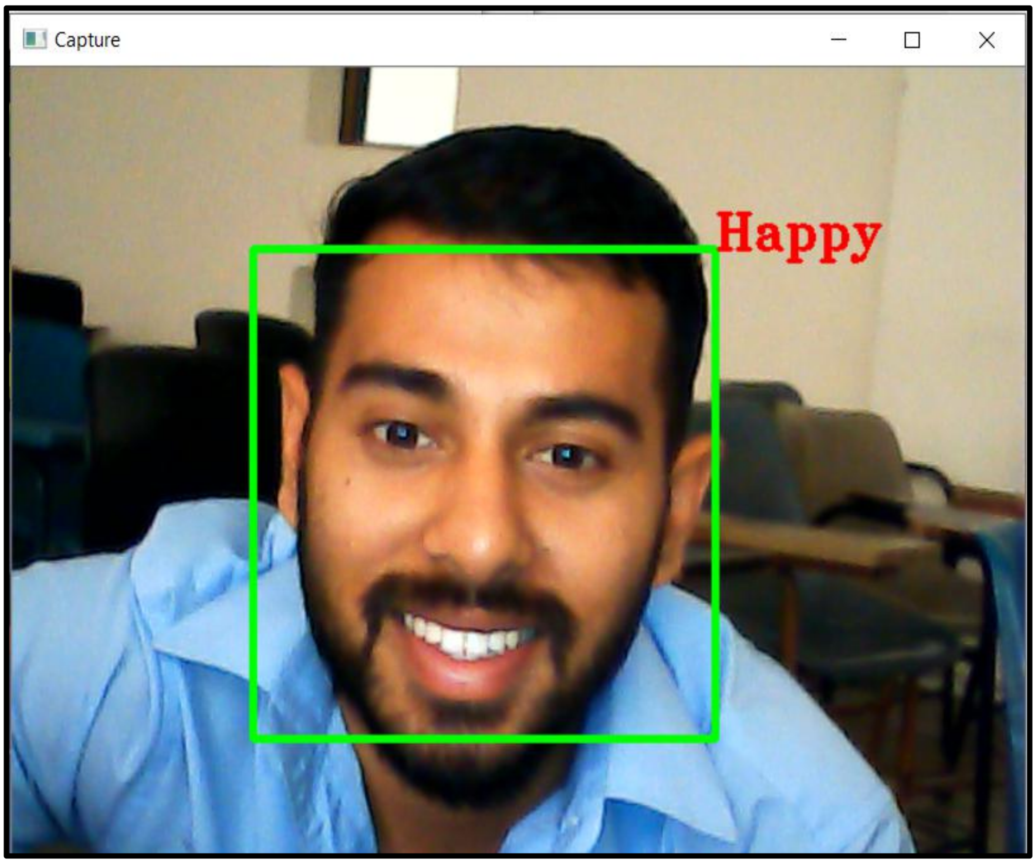
The code below deals in Image Graphing:

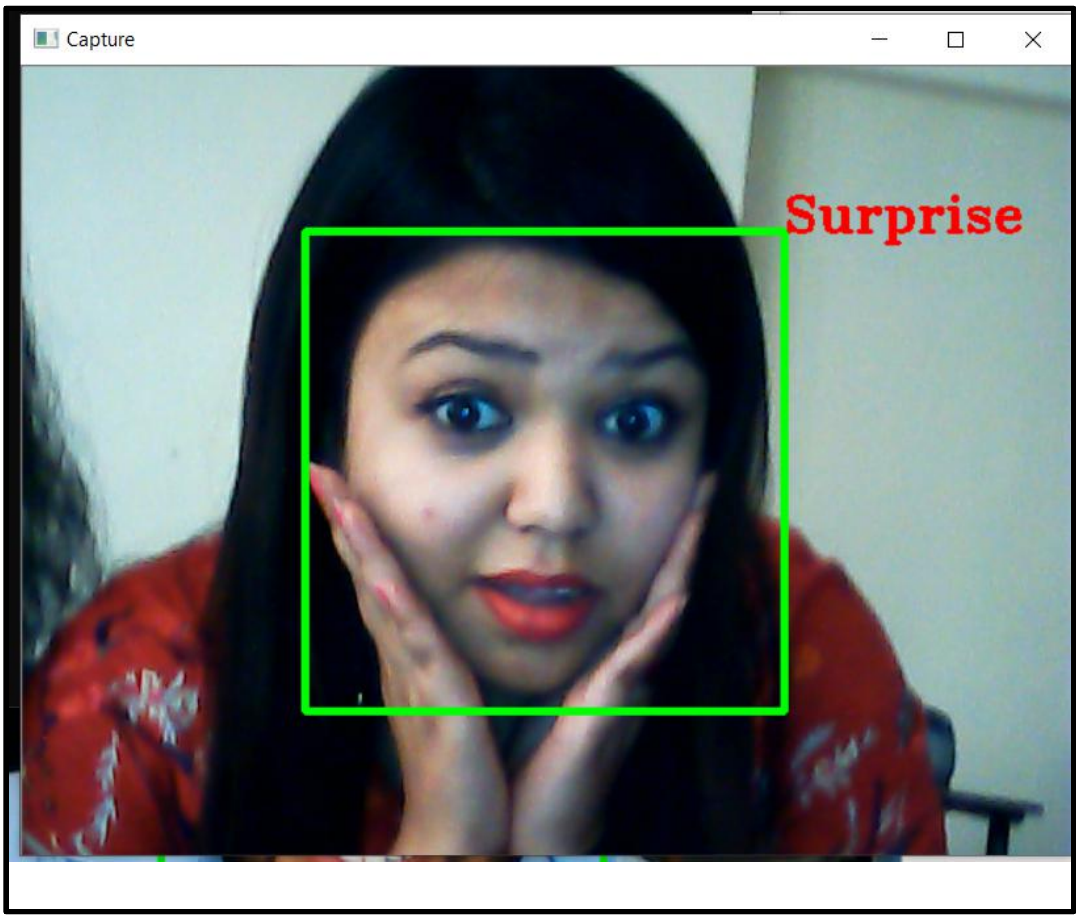

label_image.py - C:\Users\Rudrani\Downloads\face\face\label_image.py (3.7.2)
File Edit Format Run Options Window Help

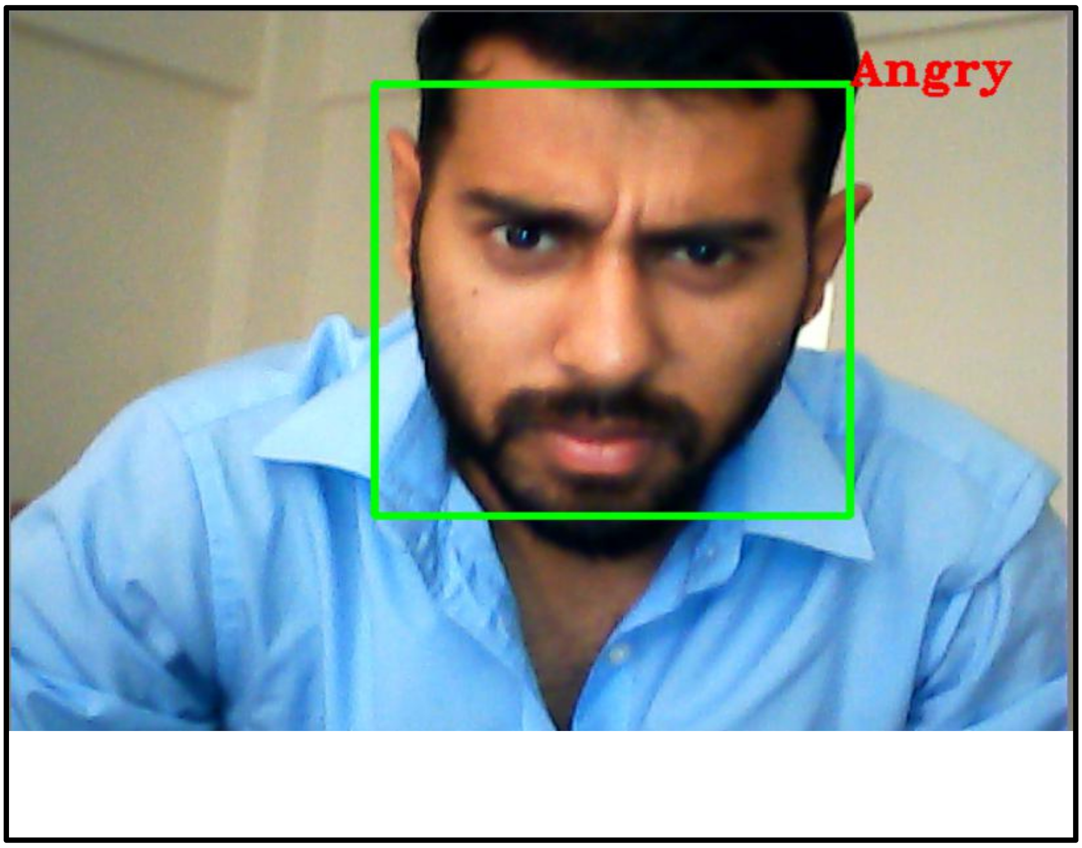
```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import argparse
import sys
import time
import numpy as np
import tensorflow as tf

def load_graph(model_file):
  graph = tf.Graph()
  graph_def = tf.GraphDef()
  with open(model_file, "rb") as f:
    graph_def.ParseFromString(f.read())
  with graph.as_default():
    tf.import_graph_def(graph_def)
  return graph

def read_tensor_from_image_file(file_name, input_height=299, input_width=299,
                                input_mean=0, input_std=255):
  input_name = "file_reader"
  output_name = "normalized"
  file_reader = tf.read_file(file_name, input_name)
  if file_name.endswith(".png"):
    image_reader = tf.image.decode_png(file_reader, channels = 3,
                                       name='png_reader')
  elif file_name.endswith(".gif"):
    image_reader = tf.squeeze(tf.image.decode_gif(file_reader,
                                                  name='gif_reader'))
  elif file_name.endswith(".bmp"):
    image_reader = tf.image.decode_bmp(file_reader, name='bmp_reader')
  else:
    image_reader = tf.image.decode_jpeg(file_reader, channels = 3,
                                        name='jpeg_reader')
  float_caster = tf.cast(image_reader, tf.float32)
  dims_expander = tf.expand_dims(float_caster, 0);
  resized = tf.image.resize_bilinear(dims_expander, [input_height, input_width])
  normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
  sess = tf.Session()
  result = sess.run(normalized)
  return result
def load_labels(label_file):
  label = []
  proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
  for l in proto_as_ascii_lines:
    label.append(l.rstrip())
  return label
```

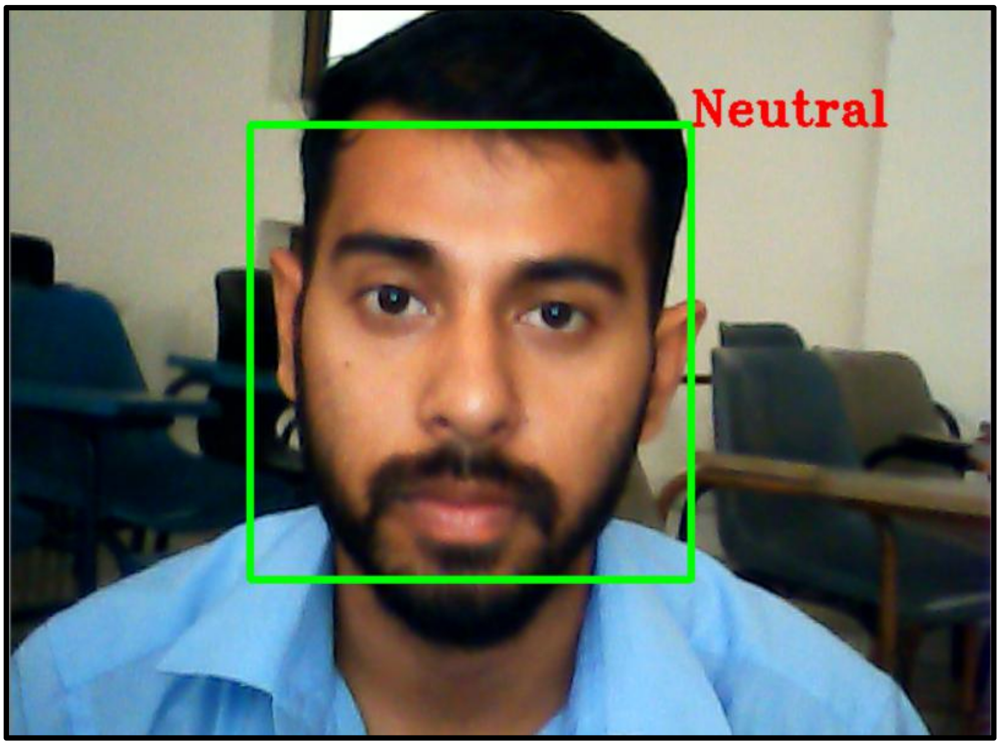
Chapter 10: Implementation and Execution

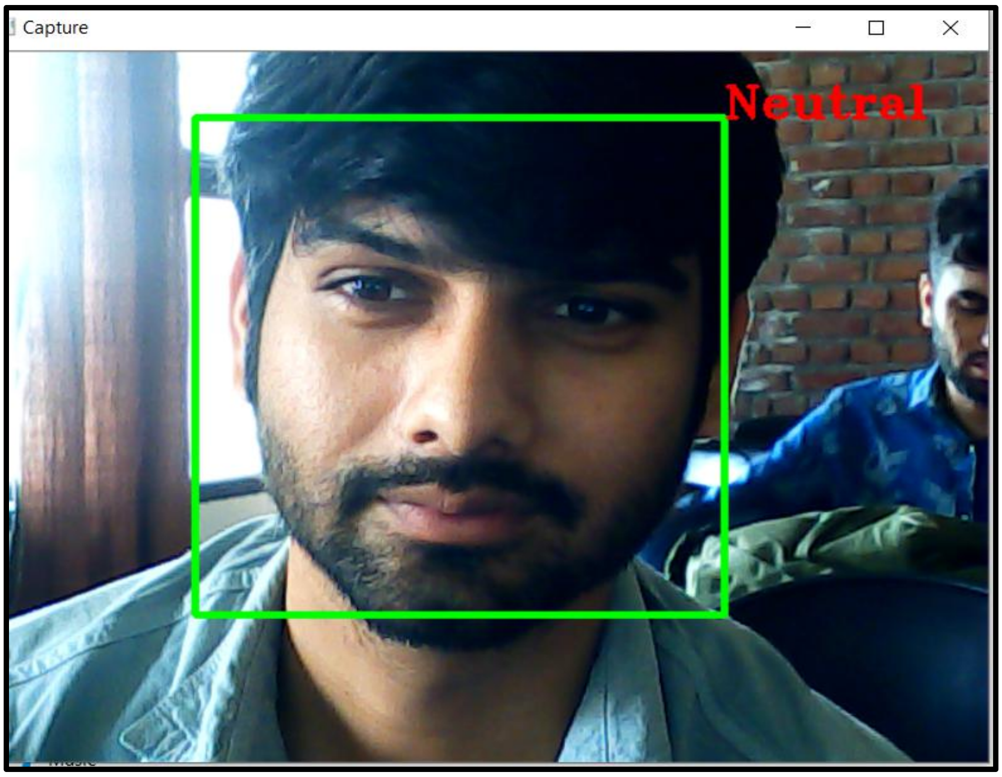





Capture
Surprise

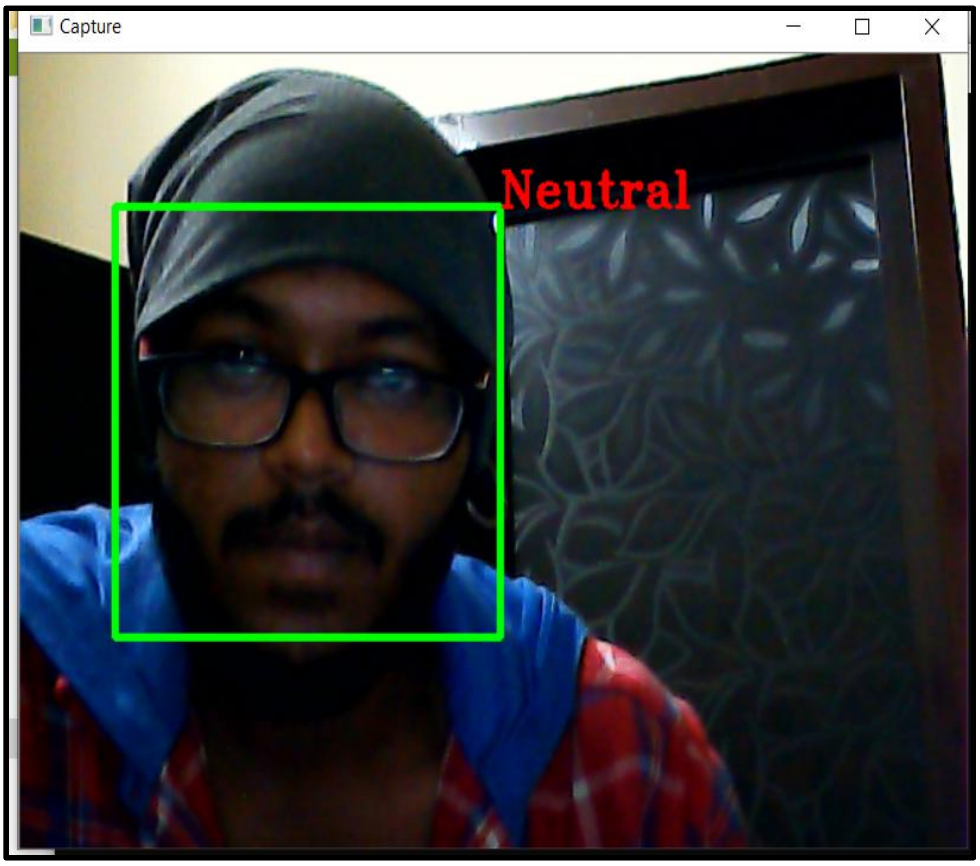

Angry


Neutral

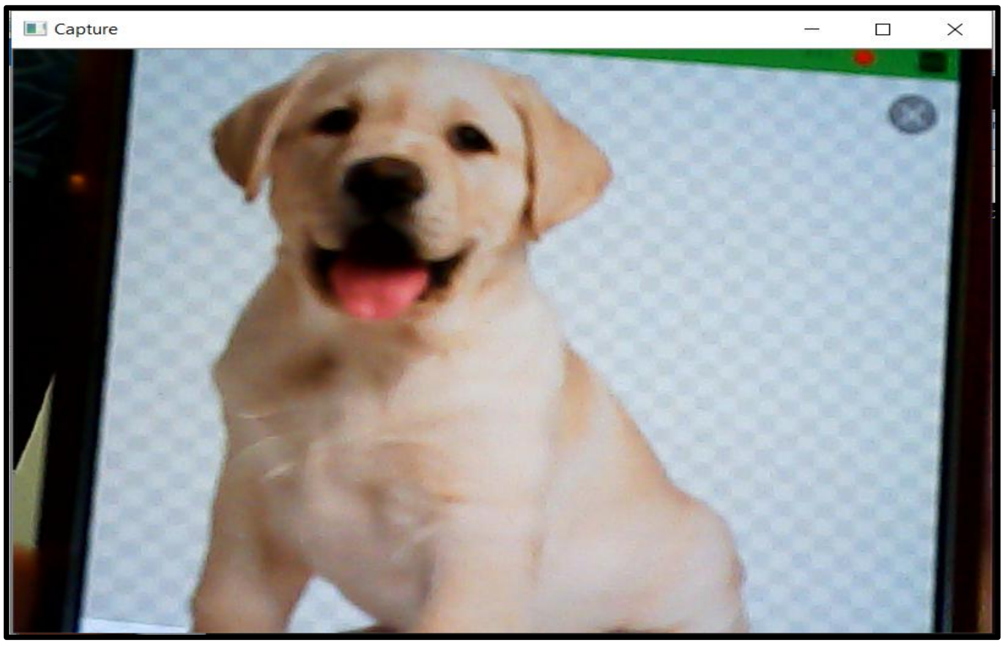

Capture
Neutral

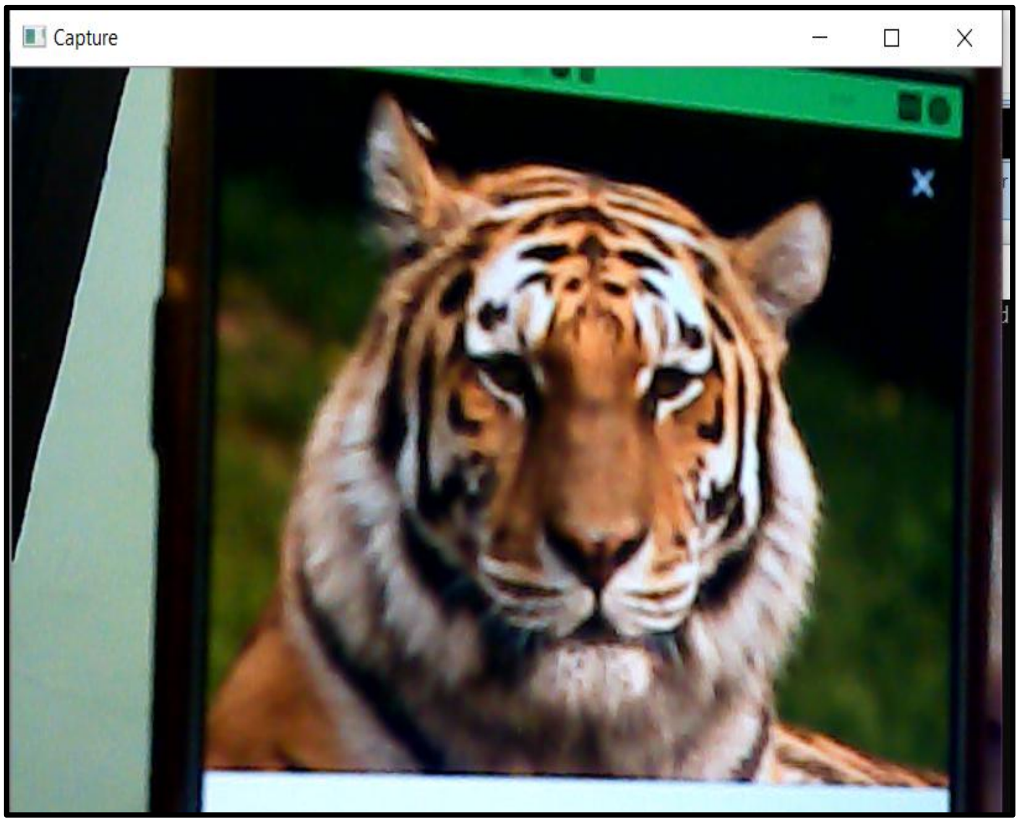
Chapter 11: Testing

Test Case 1: Testing with spectacles
Expected Outcome: Detects Emotion with Spectacles
Final Result: Pass



Test Case 2: Detecting Emotion on Animals
Expected Outcome: Do not Detect Animal Face as Human Face
Final Result: Pass

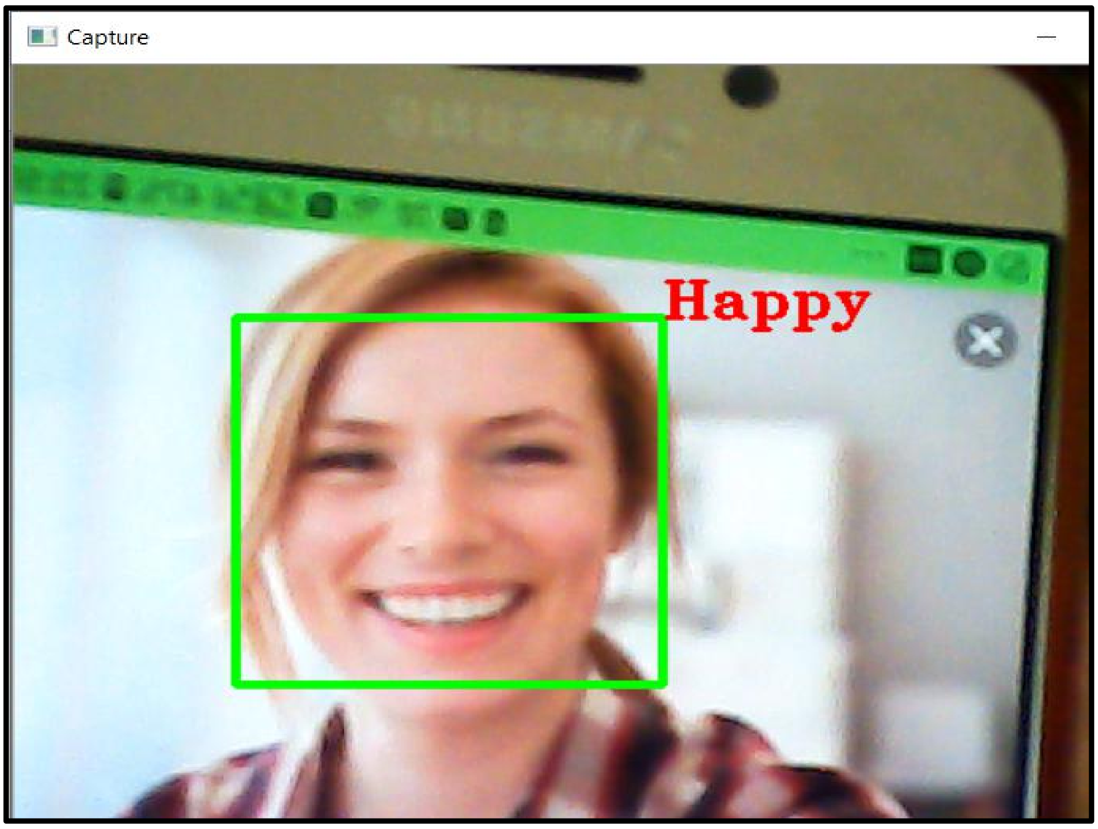


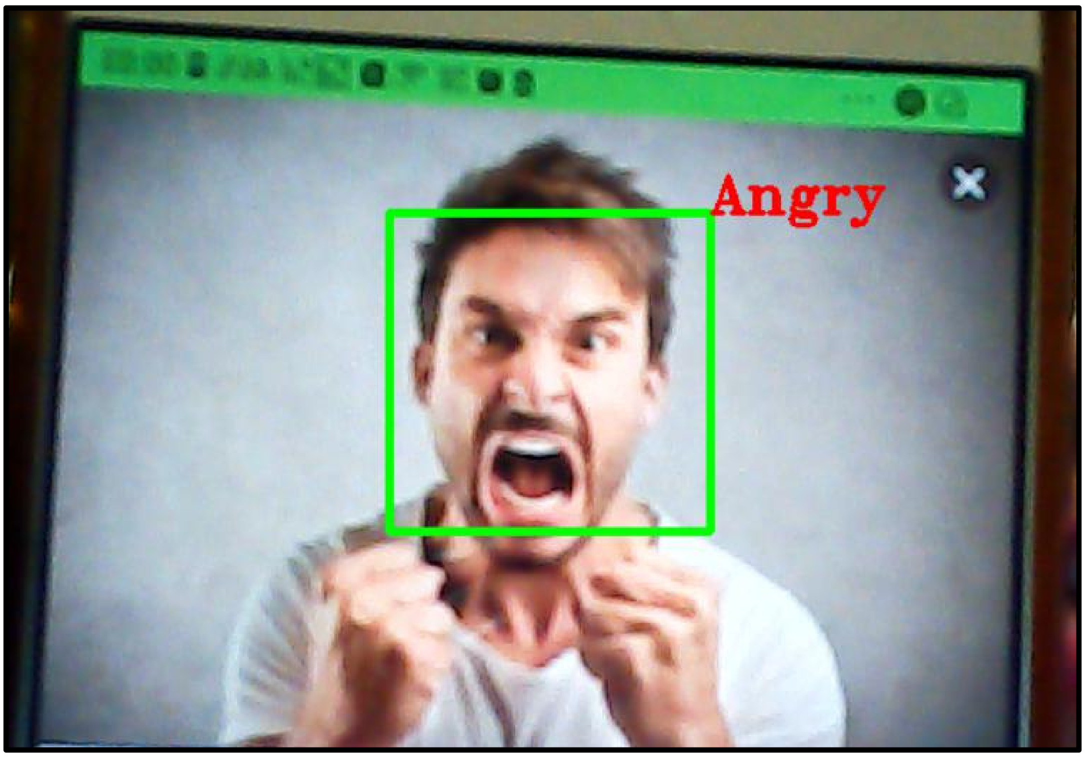


Test Case 3: Detecting Emotion from a Human Picture

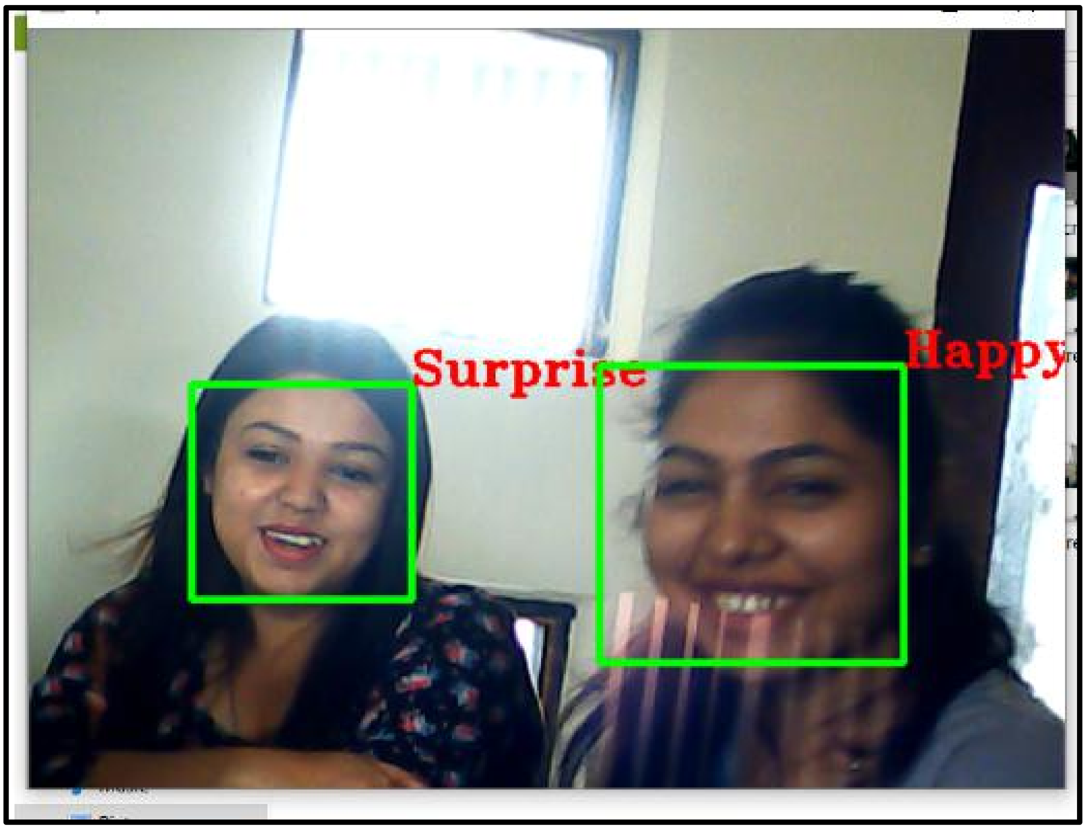
Expected Outcome: Detect Human Face from any other source as human face and detect emotion too.

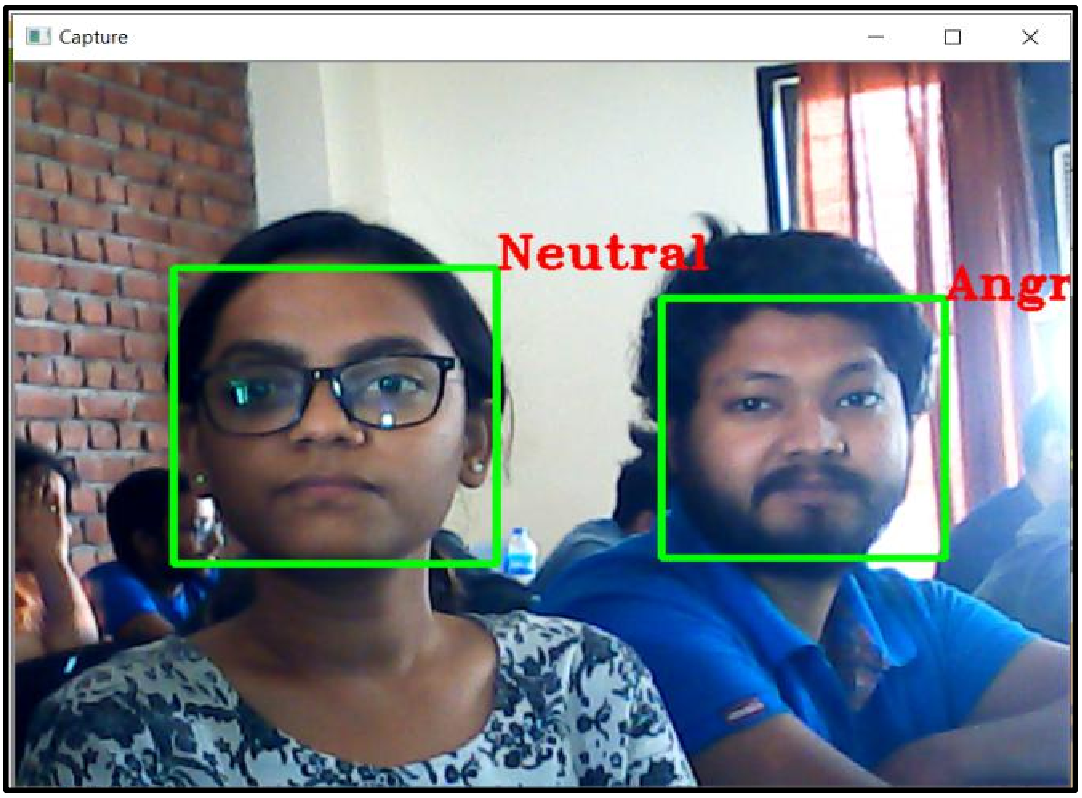
Final Result: Pass



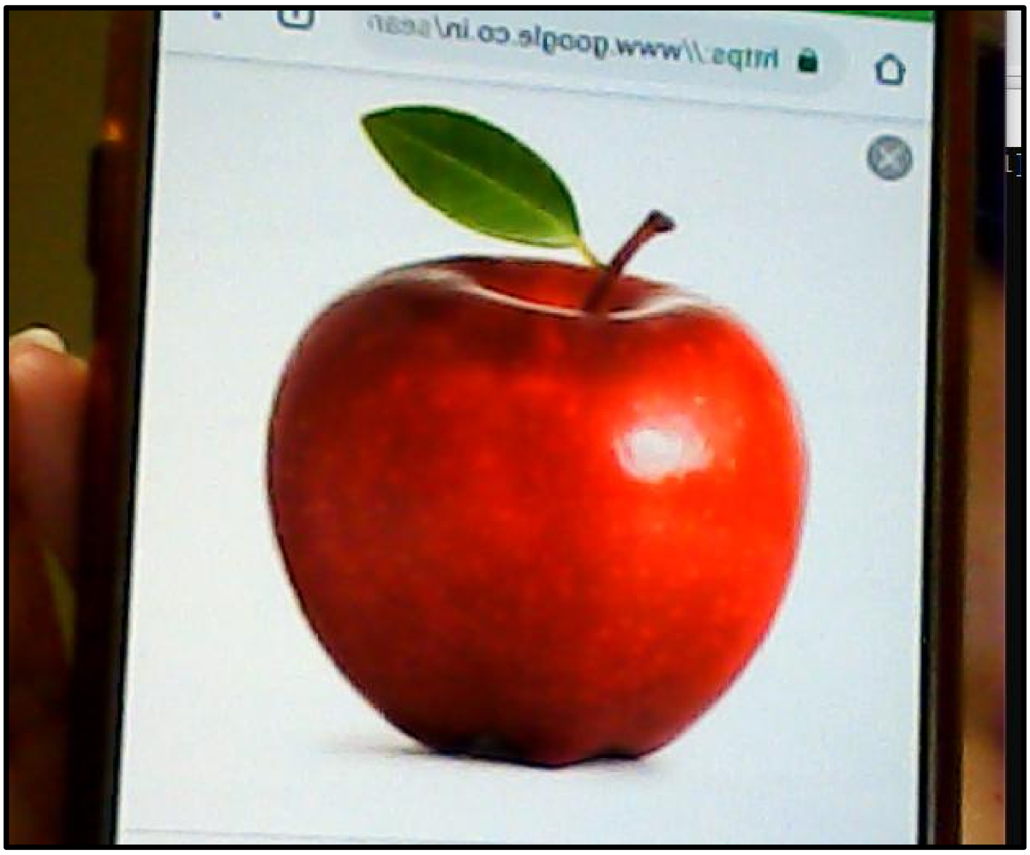


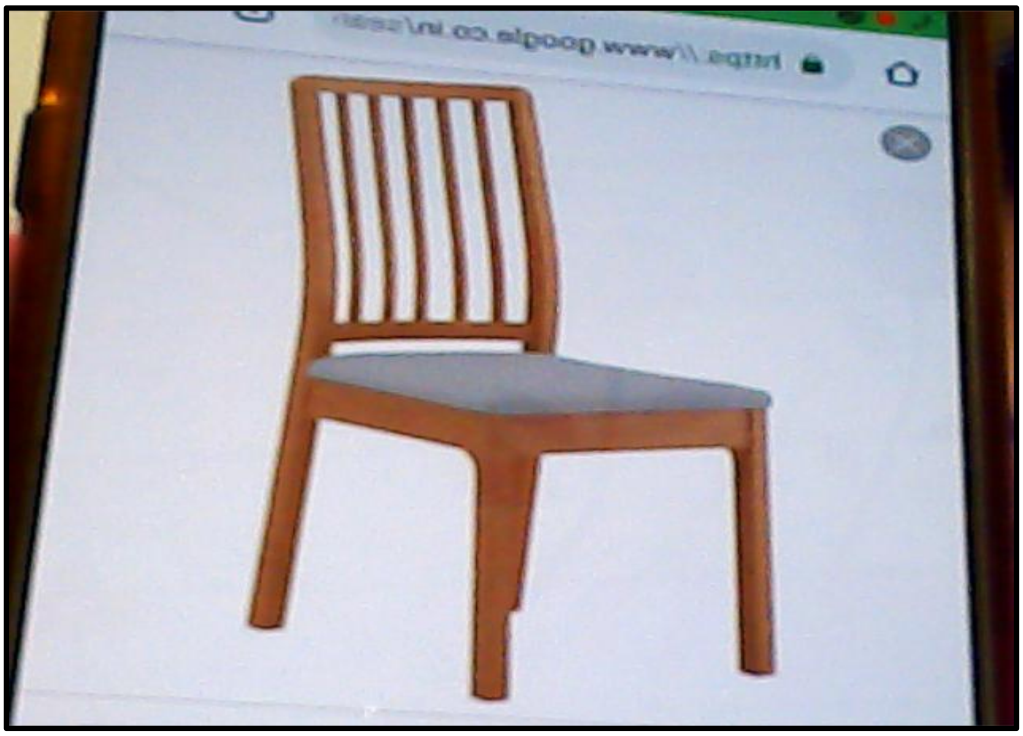
Test Case 4: Detecting Emotion of multiple faces
Expected Outcome: Detect more than one face in the video and detect emotions
Final Result: Pass





Test Case 5: Detecting Non Living Objects
Expected Outcome: Do not detect any non-living object
Final Result: Pass



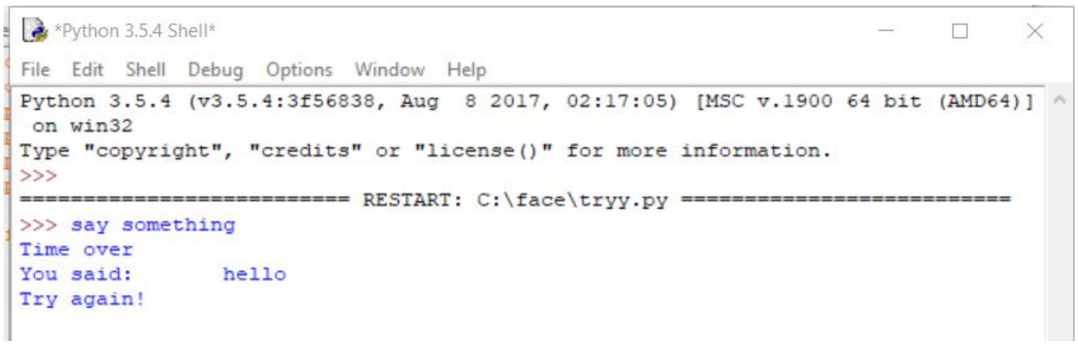


Chapter 12: Validation Check

1. Validation Check for Incorrect Voice Input: If an incorrect keyword is passed to the system via audio command due to incorrect pronunciation or voice tone an error will be shown to the user. E.g. Instead of saying “Start” if user say “State” or “Open” an error check will be displayed on screen.

User will have to keep trying until he or she can successfully say “Start” which will activate the camera at run-time to capture emotions.

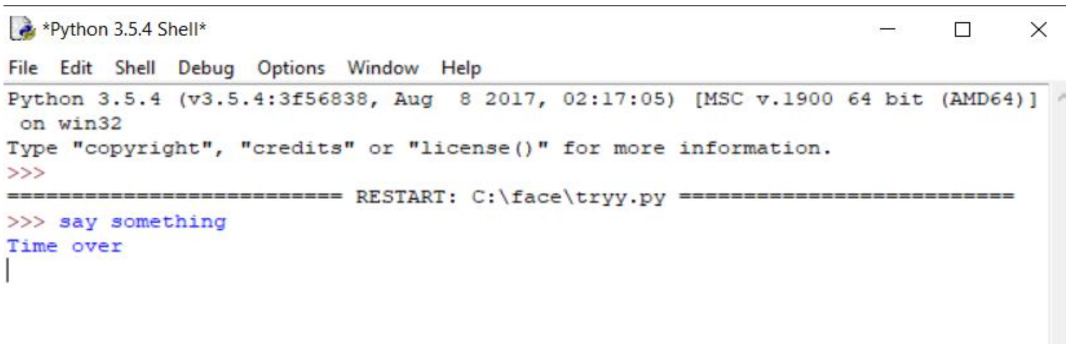
1.1 *When we enter Incorrect Speech Input*



```
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
========================== RESTART: C:\face\tryy.py ==========================
>>> say something
Time over
You said:       hello
Try again!
```

User must know the correct Keyword to start the Model.

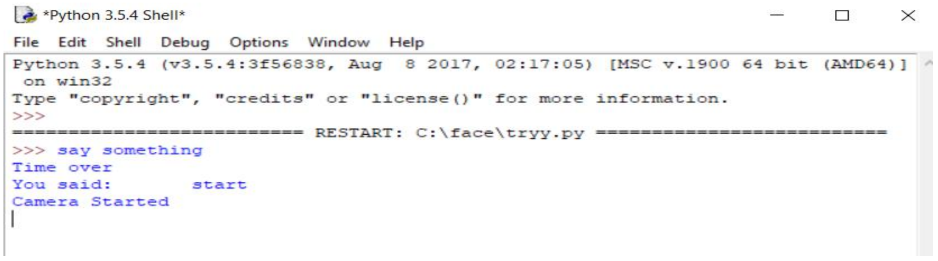
1.2 *When we do not Enter Anything*



```
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
========================== RESTART: C:\face\tryy.py ==========================
>>> say something
Time over
|
```

User must enter the correct Keyword in defined time period

1.3 *When correct Keyword is Entered in defined Time Period*

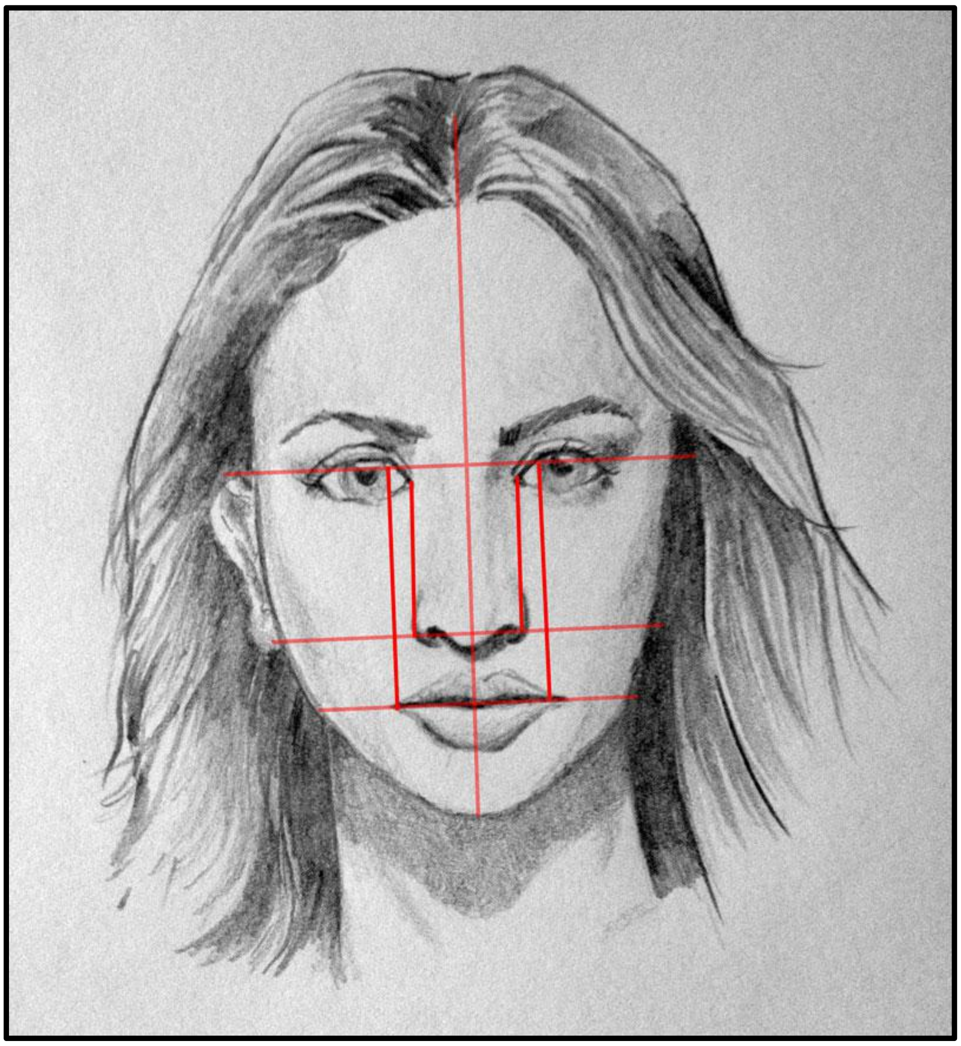


```
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
========================== RESTART: C:\face\tryy.py ==========================
>>> say something
Time over
You said:       start
Camera Started
|
```

Camera starts capturing once right keyword is entered via speech

2. Human Face Check : If human face is not detected on screen it will not detect the image. E.g if a Dog comes in front of camera, it will not detect it as a face. It will continually search for a human face.

Similarly, all other objects, living or non-living will be discarded until a human face is recognised so that it can detect emotion.



Human face is detected first, after which
facial expressions are analysed to detect emotions

Chapter 13: Future Scope

Further into the future? There are almost limitless applications of emotion recognition technology if we think outside the box. One of the good use is of using it in a retail environment. Imagine a world in which the faces of people waiting in line in a store are scanned and analysed. Not only could we reduce the need for customer experience surveys about a store visit but retailers could identify staff who make customers happy or stores that perform well and conduct A/B testing on any number of elements. Spooky? Perhaps. But exciting!

Facial expression recognition is an activity that is performed by every human in our day to day lives. Each one of us analyses the expressions of the individuals we interact with, to understand how people interact and respond with us. The malicious intentions of a thief or a person to be interviewed can be recognized with the help of his facial features and gestures.

Another best place to use facial expression recognition is in medical terms. It can be used on psychological patient to track their emotions of their day-to-day life. It can be used to solve many unsolved mysteries in medical field where its hard to understand the emotion of a patient.

Chapter 14: References

1. Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja, "Detecting Faces in Images: A Survey", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 1, JANUARY 2002.

2. Bhoi, M. N. Mohanty," Template Matching based Eye Detection in Facial Image", International Journal of Computer Applications (0975 – 8887) Volume 12– No.5, December 2010.

3. T. T. Do, T. H. Le," Facial Feature Extraction Using Geometric Feature and Independent Component Analysis", Department of Computer Sciences, University of Natural Sciences, HCMC, Vietnam.

4. Ahmad R. Naghsh-Nilchi and Mohammad Roshanzamir "An Efficient Algorithm for Motion Detection Based Facial Expression Recognition using Optical Flow" International Scholarly and Scientific Research and Innovation, 2008

5. C. Busso, Z. Deng , S. Yildirim , M. Bulut , C.M. Lee, A. Kazemzadeh , S.B. Lee, U. Neumann , S. Narayanan Analysis of Emotion Recognition using Facial Expression , Speech and Multimodal Information, ICMI'04, PY, USA, 2004

6. https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started

7. https://en.wikipedia.org/wiki/Emotion_recognition

8. https://en.wikipedia.org/wiki/Raspberry_Pi

9. Convolutional Neural Networks (CNN) With TensorFlow by Sourav from Edureka[https://www.youtube.com/watch?v=umGJ30-15_A]

10. Image Processing Facial Expression Recognition by Angana Mitra from RCC Institute of Information Technology

11.[http://rcciit.org/students_projects/projects/it/2018/GR8.pdf]

12.https://www.youtube.com/watch?v=VRcWW8q8uP8