2. INTRODUCTION

A Facial expression is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. It has been studied for a long period of time and obtaining the progress in recent decades. Though much progress has been made, recognizing facial expression with a high accuracy remains to be difficult due to the complexity and variety of facial expressions.

The system classifies facial expression of the same person into the basic emotions namely anger, disgust, fear, happiness, sadness and surprise. The main purpose of this system is efficient interaction between human beings and machines using eye gaze, facial expressions, cognitive modeling etc. Here, detection and classification of facial expressions can be used as a natural way for the interaction between man and machine and the system intensity vary from person to person and also varies along with age, gender, size and shape of face, and further, even the expressions of the same person does not remain constant with time.

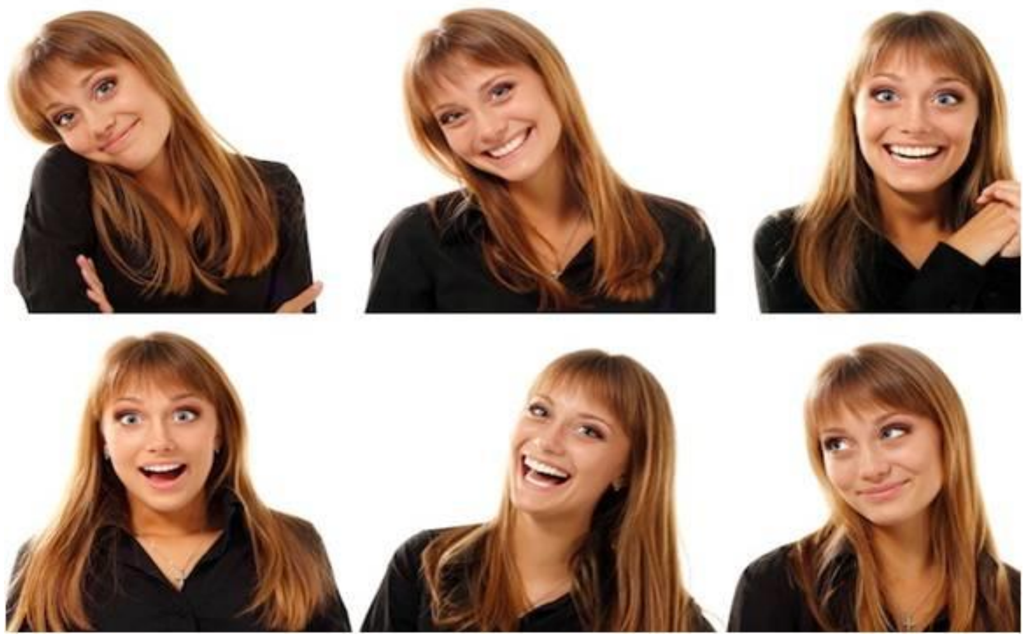


Figure 2.1 Different expressions of a human

However, the inherent variability of facial images caused by different factors like variations in illumination, pose, alignment, occlusions makes expression recognition a challenging task. Some surveys on facial feature representations for face recognition and expression analysis addressed these challenges and possible solutions in detail.

3. FACIAL EMOTION SYSTEM

The common approach to facial emotion recognition consists of three steps: face detection and tracking, feature extraction and expression classification. Face detection stage processes the facial images, without human intervention to find the face region from the input images or sequences. After face is positioned, the next step is to extract discriminative information caused by facial expressions. Facial expression recognition is the last stage of the systems. The facial changes can be identified either as prototypic emotions or as facial action units Even though humans are filled with various emotions, modern psychology defines six basic facial expressions: Happiness, Sadness, Surprise, Fear, Disgust, and Anger as universal emotions. Facial muscles movements help in identifying human emotions. The facial features are the key parameters that can be considered for recognizing emotions. The facial parameters include eyebrow, mouth, nose, eyes and cheeks.

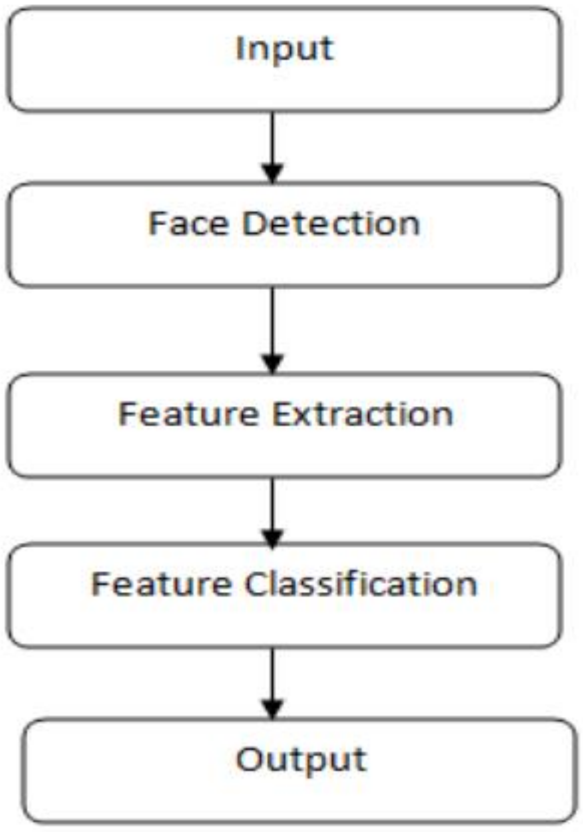


Figure 3.1 Flow chart of the process

4. SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Software Requirements

This software requirements refers to the server-side and user-side required software specifications.

Software tools used

- Python 3.6
- PyCharm
- TensorFlow

4.2Hardware Requirements

This hardware Constraints refers to the server-side and user-side required hardware specifications so that the module can be run with System.

- Operating System : Windows 10 Home
- Processor : intel i3 8th Gen
- RAM : Minimum 1GB
- Hard Disk : Minimum 32GB

5. TECHNOLOGIES

5.1 PyQt5

There are so many options provided by Python to develop GUI application and PyQt5 is one of them. PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called 'QtDesigner' to design the front-end by drag and drop method so that development can become faster and one can give more time on back-end stuff. PyQt5 is a set of Python bindings for Qt5 application framework from Digital. It is available for the Python 2.x and 3.x. This tutorial uses Python 3. Qt library is one of the most powerful GUI libraries. The official home site for PyQt5 is . PyQt5 is developed by Riverbank Computing.PyQt5 is implemented as a set of Python modules. It has over 620 classes and 6000 functions and methods. It is a multi platform toolkit which runs on all major operating systems

5.2 Tensorflow

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!

TensorFlow is basically a software library for numerical computation using data flow graphs where:

nodes in the graph represent mathematical operations.

edges in the graph represent the multidimensional data arrays (called tensors) communicated between them.

TensorFlow APIs

TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories:

Low level API

- Complete programming control
- Recommended for machine learning researchers
- Provides fine levels of control over the models
- TensorFlow Core is the low level API of TensorFlow.

High level API

- Built on top of TensorFlow Core
- Easier to learn and use than TensorFlow Core
- Make repetitive tasks easier and more consistent between different users.

6. LIBRARIES

6.1 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the and array object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of and array will create a new array and delete the original.

• The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

• NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

• A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

6.2 Keras

Keras is a high-level neural networks API, capable of running on top of Tensorflow, Theano, and CNTK. It enables fast experimentation through a high level, user-friendly, modular and extensible API. Keras can also be run on both CPU and GPU. Keras was developed and is maintained by Francois Chollet and is part of the Tensorflow core, which makes it Tensor Flows preferred high-level API.This article is the first of a series explaining how to use Keras for deep learning. In this article, we will go over the basics of Keras including the two most used Keras models (Sequential and Functional), the core layers as well as some preprocessing functionalities.

6.3 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in an easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

6.4 Face detection in OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions for real-time computer vision. The face detection part of the project was made using an OpenCV Library for Scala. The reason was that most Face APIs are restricted to doing detection on pictures only, whereas the project was required to have face detection done on a live video footage to speed up the process of checking student attendance and prevent queues before lectures. The OpenCV library proved to be flexible enough for the project as it can accurately detect a face in real time and highlight it by drawing a rectangle around the faces of the students passing by. This all happens in a window separate from the face recognition so the lecturer can keep track of both students passing by while having their faces detected and the feedback from the recognition part of the system. While faces are being detected, the application takes a snapshot of the live footage every second and then sends it to the recognition system.

7.ALGORITHM

In machine learning, support-vector machines (SVMs, also support-vector networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering[2] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

SVMs can be used to solve various real-world problems:

- SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.[citation needed] Some methods for shallow semantic parsing are based on support vector machines.

- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik

- Hand-written characters can be recognized using SVM
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support-vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

8.IMPLEMENTATION

```
from PyQt5 import QtCore, QtGui, QtWidgets
from webcam import WebCamDetection
from ImageUpload import ImageUpload
import sys
class Home(object):
    def webcamdef(self):
        try:
            self.showAlertBox("Information", "      Launching Web Cam, Press ' q ' to
Stop the Expression Detection ")
            WebCamDetection.process()
        except Exception as e:
            print("Error=" + e.args[0])
            tb = sys.exc_info()[2]
            print(tb.tb_lineno)
            print(e)
    def uploadpic(self):
        try:
            self.Dialog2 = QtWidgets.QDialog()
            self.ui2 = ImageUpload()
            self.ui2.setupUi(self.Dialog2)
            self.Dialog2.show()
        except Exception as e:
            print("Error=" + e.args[0])
            tb = sys.exc_info()[2]
            print(tb.tb_lineno)
            print(e)
    def showAlertBox(self, title, message):
        msgBox = QtWidgets.QMessageBox()
        msgBox.setIcon(QtWidgets.QMessageBox.Information)
        msgBox.setWindowTitle(title)
        msgBox.setText(message)
        msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
        msgBox.exec_()
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(766, 568)
        Dialog.setStyleSheet("background-color: rgb(186, 186, 0);")
        self.verticalLayout = QtWidgets.QVBoxLayout(Dialog)
        self.verticalLayout.setObjectName("verticalLayout")
        self.tabWidget = QtWidgets.QTabWidget(Dialog)
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.label = QtWidgets.QLabel(self.tab)
        self.label.setGeometry(QtCore.QRect(70, 20, 641, 81))
        self.label.setStyleSheet("font: 18pt \"Century Gothic\";\n"
                                 "color: rgb(255, 255, 255);")
        self.label.setObjectName("label")
        self.tableView = QtWidgets.QTableView(self.tab)
        self.tableView.setGeometry(QtCore.QRect(40, 120, 661, 371))
        self.tableView.setStyleSheet("border-image: url(Camera.jpg);")
        self.tableView.setObjectName("tableView")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.label_2 = QtWidgets.QLabel(self.tab_2)
        self.label_2.setGeometry(QtCore.QRect(70, 20, 641, 81))
        self.label_2.setStyleSheet("font: 18pt \"Century Gothic\";\n"
                                   "color: rgb(255, 255, 255);")
        self.label_2.setObjectName("label_2")
```

```
        self.uploadimage = QtWidgets.QPushButton(self.tab_2)
        self.uploadimage.setGeometry(QtCore.QRect(140, 170, 190, 200))
        self.uploadimage.setStyleSheet("image: url(Images-icon.png);")
        self.uploadimage.setText("")
        self.uploadimage.setObjectName("uploadimage")
        self.uploadimage.clicked.connect(self.uploadpic)
        self.webcam = QtWidgets.QPushButton(self.tab_2)
        self.webcam.setGeometry(QtCore.QRect(420, 170, 190, 200))
        self.webcam.setStyleSheet("image: url(e502608e8b.png);")
        self.webcam.setText("")
        self.webcam.setObjectName("webcam")
        self.webcam.clicked.connect(self.webcamdef)
        self.tabWidget.addTab(self.tab_2, "")
        self.verticalLayout.addWidget(self.tabWidget)
        self.retranslateUi(Dialog)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
        def retranslateUi(self, Dialog):
        translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.label.setText(_translate("Dialog",
        "<html><head/><body><p align=\"center\">Human Face Detection      and
Facial      Expressions  </p><p
align=\"center\">Identification</p></body></html>"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
_translate("Dialog", "Home"))
        self.label_2.setText(_translate("Dialog",
                              "<html><head/><body><p align=\"center\">Human Face
Detection and Facial Expressions  </p><p
align=\"center\">Identification</p></body></html>"))
```

```
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
_translate("Dialog", "Options"))



if __name__ == "__main__":
    import sys

    app = QtWidgets.QApplication(sys.argv)
    Dialog = QtWidgets.QDialog()
    ui = Home()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())




from scipy.io import loadmat
import pandas as pd
import numpy as np
from random import shuffle
import os
import cv2



class DataManager(object):
    """Class for loading fer2013 emotion classification dataset or
        imdb gender classification dataset."""
    def __init__(self, dataset_name='imdb',
                 dataset_path=None, image_size=(48, 48)):

        self.dataset_name = dataset_name
        self.dataset_path = dataset_path
        self.image_size = image_size
        if self.dataset_path is not None:
            self.dataset_path = dataset_path
        elif self.dataset_name == 'imdb':
            self.dataset_path = '../datasets/imdb_crop/imdb.mat'
        elif self.dataset_name == 'fer2013':
            self.dataset_path = '../datasets/fer2013/fer2013.csv'
        elif self.dataset_name == 'KDEF':
            self.dataset_path = '../datasets/KDEF/'
        else:
            raise Exception(
                    'Incorrect dataset name, please input imdb or fer2013')

    def get_data(self):
        if self.dataset_name == 'imdb':
            ground_truth_data = self._load_imdb()
        elif self.dataset_name == 'fer2013':
            ground_truth_data = self._load_fer2013()
        elif self.dataset_name == 'KDEF':
            ground_truth_data = self._load_KDEF()
        return ground_truth_data

    def _load_imdb(self):
        face_score_treshold = 3
        dataset = loadmat(self.dataset_path)
image_names_array = dataset['imdb']['full_path'][0, 0][0]
gender_classes = dataset['imdb']['gender'][0, 0][0]
face_score = dataset['imdb']['face_score'][0, 0][0]
second_face_score = dataset['imdb']['second_face_score'][0, 0][0]
        face_score_mask = face_score > face_score_treshold
        second_face_score_mask = np.isnan(second_face_score)
        unknown_gender_mask = np.logical_not(np.isnan(gender_classes))
        mask = np.logical_and(face_score_mask, second_face_score_mask)
        mask = np.logical_and(mask, unknown_gender_mask)
        image_names_array = image_names_array[mask]
        gender_classes = gender_classes[mask].tolist()
        image_names = []
        for image_name_arg in range(image_names_array.shape[0]):
            image_name = image_names_array[image_name_arg][0]
            image_names.append(image_name)
        return dict(zip(image_names, gender_classes))

    def _load_fer2013(self):
        data = pd.read_csv(self.dataset_path)
        pixels = data['pixels'].tolist()
        width, height = 48, 48
        faces = []
        for pixel_sequence in pixels:
            face = [int(pixel) for pixel in pixel_sequence.split(' ')]
            face = np.asarray(face).reshape(width, height)
            face = cv2.resize(face.astype('uint8'), self.image_size)
            faces.append(face.astype('float32'))
        faces = np.asarray(faces)
        faces = np.expand_dims(faces, -1)
        emotions = pd.get_dummies(data['emotion']).as_matrix()
        return faces, emotions

    def _load_KDEF(self):
        class_to_arg = get_class_to_arg(self.dataset_name)
        num_classes = len(class_to_arg)
        file_paths = []
        for folder, subfolders, filenames in os.walk(self.dataset_path):
            for filename in filenames:
                if filename.lower().endswith(('.jpg')):
                    file_paths.append(os.path.join(folder, filename))

        num_faces = len(file_paths)
        y_size, x_size = self.image_size
        faces = np.zeros(shape=(num_faces, y_size, x_size))
        emotions = np.zeros(shape=(num_faces, num_classes))
        for file_arg, file_path in enumerate(file_paths):
            image_array = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
            image_array = cv2.resize(image_array, (y_size, x_size))
            faces[file_arg] = image_array
file_basename = os.path.basename(file_path)
            file_emotion = file_basename[4:6]
            # there are two file names in the dataset
            # that don't match the given classes
            try:
                emotion_arg = class_to_arg[file_emotion]
            except:
                continue
            emotions[file_arg, emotion_arg] = 1
        faces = np.expand_dims(faces, -1)
        return faces, emotions



def get_labels(dataset_name):
    if dataset_name == 'fer2013':
        return {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy',
                4: 'sad', 5: 'surprise', 6: 'neutral'}
    elif dataset_name == 'imdb':
        return {0: 'woman', 1: 'man'}
    elif dataset_name == 'KDEF':
        return {0: 'AN', 1: 'DI', 2: 'AF', 3: 'HA', 4: 'SA', 5: 'SU', 6: 'NE'}
    else:
        raise Exception('Invalid dataset name')


def get_class_to_arg(dataset_name='fer2013'):
    if dataset_name == 'fer2013':
        return {'angry': 0, 'disgust': 1, 'fear': 2, 'happy': 3, 'sad': 4,
                'surprise': 5, 'neutral': 6}
    elif dataset_name == 'imdb':
        return {'woman': 0, 'man': 1}
    elif dataset_name == 'KDEF':
        return {'AN': 0, 'DI': 1, 'AF': 2, 'HA': 3, 'SA': 4, 'SU': 5, 'NE': 6}
    else:
        raise Exception('Invalid dataset name')


def split_imdb_data(ground_truth_data, validation_split=.2, do_shuffle=False):
    ground_truth_keys = sorted(ground_truth_data.keys())
    if do_shuffle is not False:
        shuffle(ground_truth_keys)
    training_split = 1 - validation_split
    num_train = int(training_split * len(ground_truth_keys))
    train_keys = ground_truth_keys[:num_train]
    validation_keys = ground_truth_keys[num_train:]
    return train_keys, validation_keys
```

```
def split_data(x, y, validation_split=.2):
    num_samples = len(x)
    num_train_samples = int((1 - validation_split)*num_samples)
    train_x = x[:num_train_samples]
    train_y = y[:num_train_samples]
    val_x = x[num_train_samples:]
    val_y = y[num_train_samples:]
    train_data = (train_x, train_y)
    val_data = (val_x, val_y)
    return train_data, val_data

 from statistics import mode
   import os
   import cv2
   from keras.models import load_model
import numpy as np
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

class ImageDetection:


    @staticmethod
    def process(image):


        os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
        # parameters for loading data and images
        detection_model_path =
'C:/Users/Admin/Documents/Emotion/trained_models/detection_models/haarcascade
_frontalface_default.xml'
        emotion_model_path =
'C:/Users/Admin/Documents/Emotion/trained_models/emotion_models/fer2013_min
i_XCEPTION.102-0.66.hdf5'
        emotion_labels = get_labels('fer2013')

        # hyper-parameters for bounding boxes shape
        frame_window = 10
        emotion_offsets = (20, 40)

        # loading models
        face_detection = load_detection_model(detection_model_path)
        emotion_classifier = load_model(emotion_model_path, compile=False)

        # getting input model shapes for inference
        emotion_target_size = emotion_classifier.input_shape[1:3]

        # starting lists for calculating modes
        emotion_window = []

        # starting video streaming
        cv2.namedWindow('window_frame')
        #video_capture = cv2.VideoCapture(0)
        # Load an color image in grayscale
        img = cv2.imread(image)

        bgr_image = img
        gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
        rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
        faces = detect_faces(face_detection, gray_image)

        for face_coordinates in faces:

            x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
            gray_face = gray_image[y1:y2, x1:x2]
            try:
                gray_face = cv2.resize(gray_face, (emotion_target_size))
            except:
                continue

            gray_face = preprocess_input(gray_face, True)
            gray_face = np.expand_dims(gray_face, 0)
            gray_face = np.expand_dims(gray_face, -1)
            emotion_prediction = emotion_classifier.predict(gray_face)
            motion_probability = np.max(emotion_prediction)
            emotion_label_arg = np.argmax(emotion_prediction)
            emotion_text = emotion_labels[emotion_label_arg]
            emotion_window.append(emotion_text)
            if len(emotion_window) > frame_window:
                emotion_window.pop(0)
            try:
                emotion_mode = mode(emotion_window)
            except:
                continue
if emotion_text == 'angry':
                color = emotion_probability * np.asarray((255, 0, 0))
            elif emotion_text == 'sad':
                color = emotion_probability * np.asarray((0, 0, 255))
            elif emotion_text == 'happy':
                color = emotion_probability * np.asarray((255, 255, 0))
            elif emotion_text == 'surprise':
                color = emotion_probability * np.asarray((0, 255, 255))
            else:
                color = emotion_probability * np.asarray((0, 255, 0))

            color = color.astype(int)
            color = color.tolist()

            draw_bounding_box(face_coordinates, rgb_image, color)
            draw_text(face_coordinates, rgb_image, emotion_mode,
                      color, 0, -45, 1, 1)
            bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
            cv2.imwrite("out.jpg",bgr_image)
            #os.system("powershell -c out.jpg")
if __name__=="__main__":
ImageDetection.process('C:/Users/sajid/PycharmProjects/FaceExpression/venv/CT/i
mage.JPG')
```

9.OUTPUT

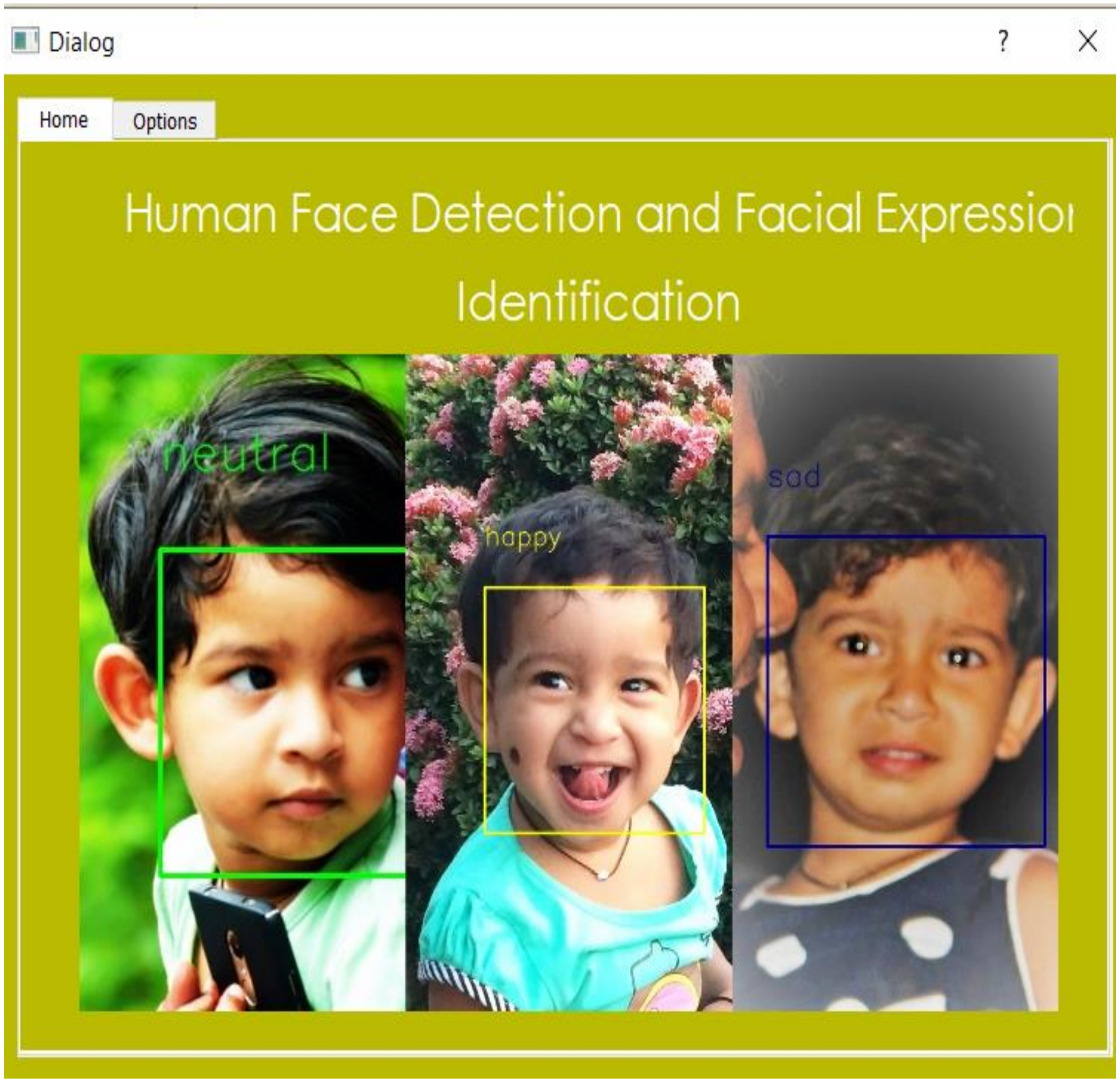


Figure 9.1 Output of home page

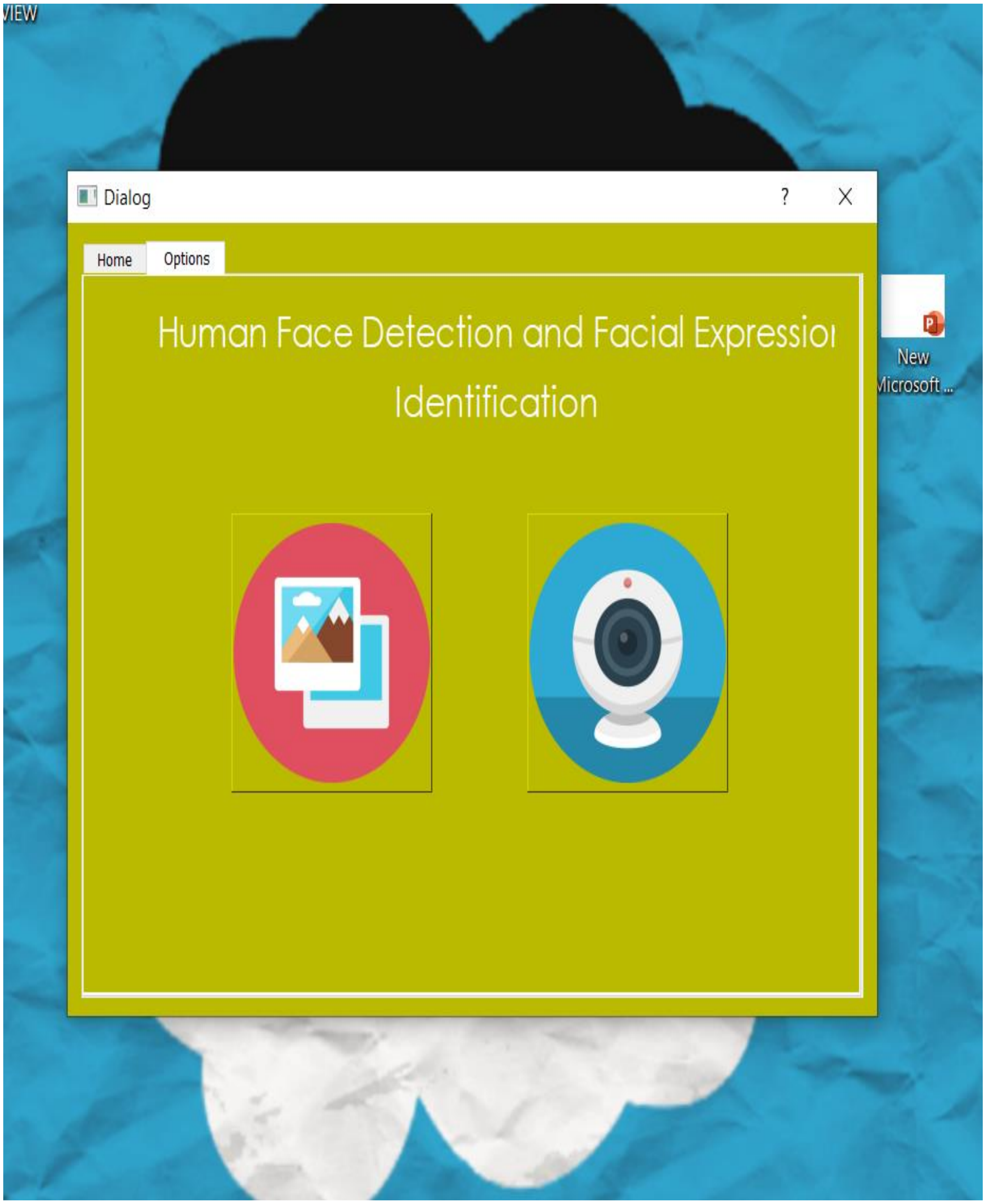
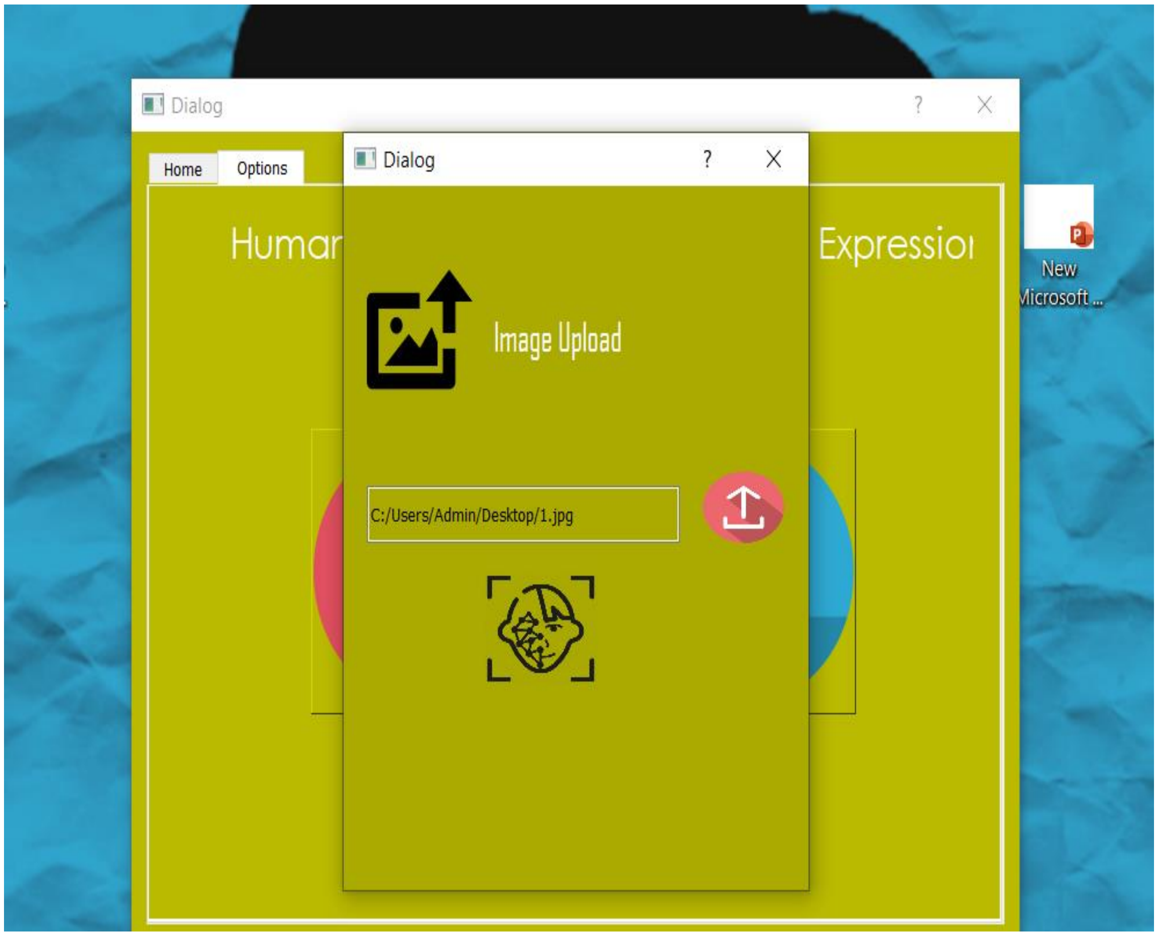
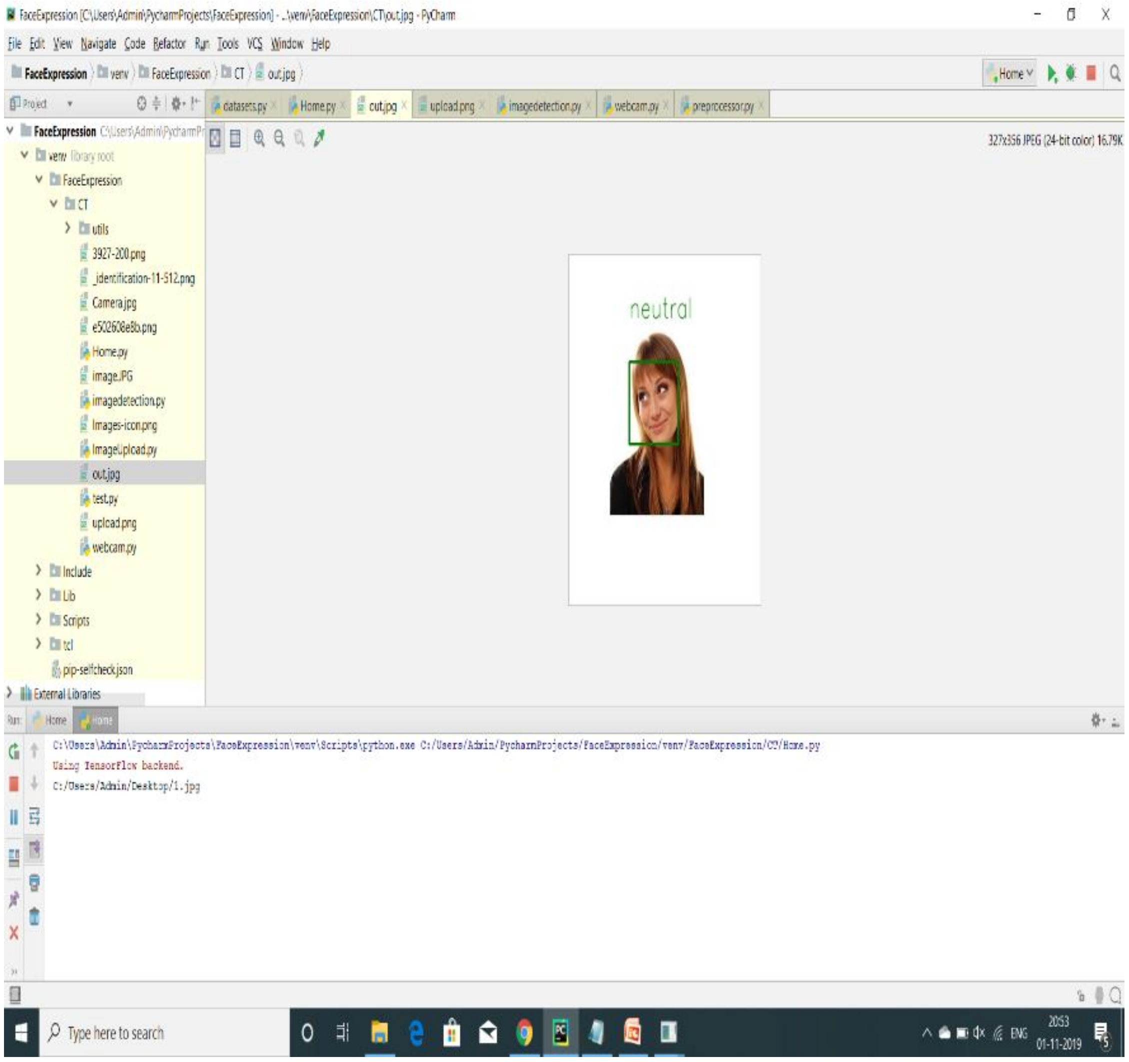


Figure 9.2 Output for the options



C:/Users/Admin/Desktop/1.jpg

Figure 9.3 Image



```
C:\Users\Admin\PycharmProjects\FaceExpression\venv\Scripts\python.exe C:/Users/Admin/PycharmProjects/FaceExpression/venv/FaceExpression/CT/Home.py
Using TensorFlow backend.
C:/Users/Admin/Desktop/1.jpg
```

Figure 9.4 Output for image upload

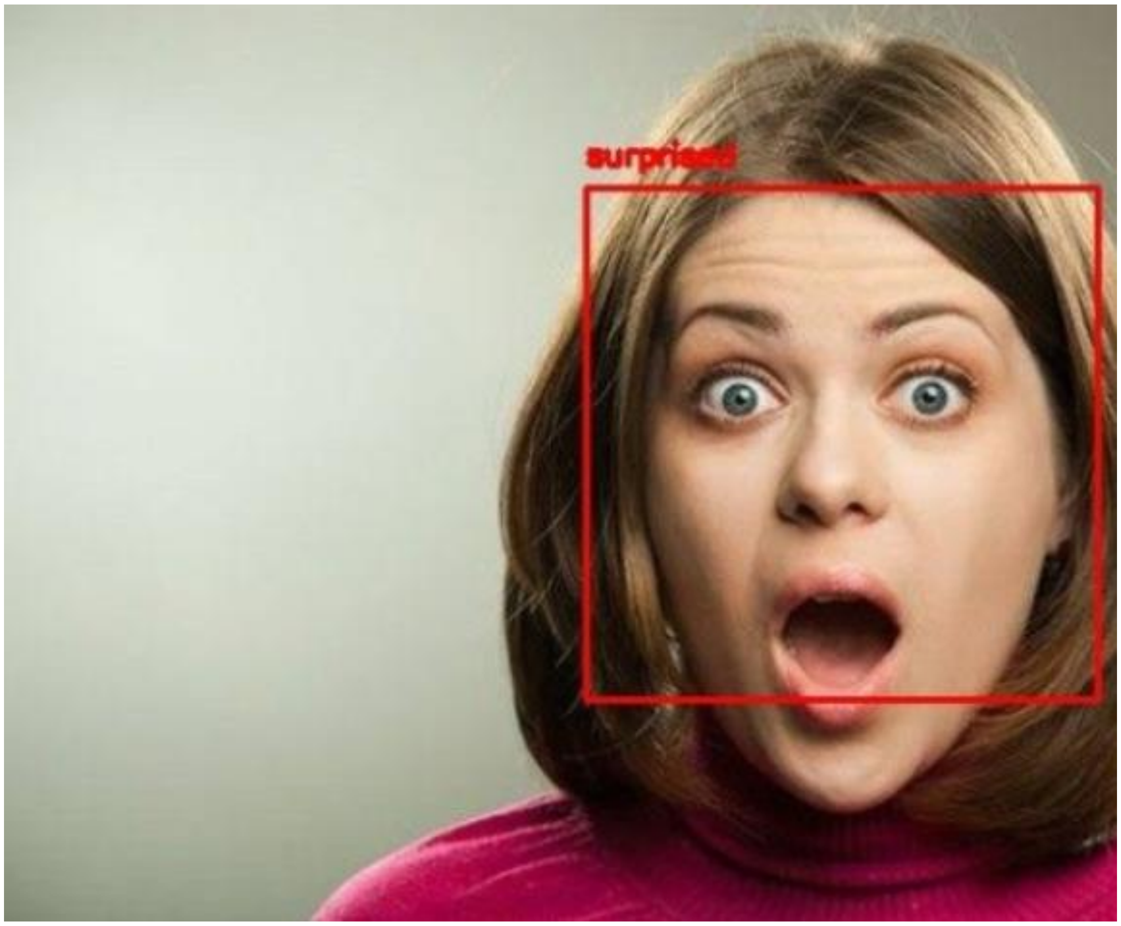


Figure.9.5 Output on clicking camera

10.CONCLUSION AND FUTURESCOPE

The facial expression recognition system presented in this research work contributes a resilient face recognition model based on the mapping of behavioral characteristics with the physiological biometric characteristics. The physiological characteristics of the human face with relevance to various expressions such as happiness, sadness, fear, anger, surprise and disgust are associated with geometrical structures which restored as base matching template for the recognition system. The behavioral aspect of this system relates the attitude behind different expressions as property base. The property bases are alienated as exposed and hidden category in genetic algorithmic genes. The gene training set evaluates the expressional uniqueness of individual faces and provide a resilient expressional recognition model in the field of biometric security. The general experimental evaluation of the face expressional system guarantees better face recognition rates. Having examined techniques to cope with expression variation, in future it may be investigated in more depth about the face classification problem and optimal fusion of color and depth information.

REFERENCES

[1] P. Abhang, S. Rao, B. W. Gawali, and P. Rokade, “Article: Emotion recognition using speech and eeg signal a review,” International Journal of Computer Applications, vol. 15, pp. 37–40, February 2011.

[2] P. Ekman, Universals and cultural differences in facial expressions of emotion. Nebraska, USA: Lincoln University of Nebraska Press, 1971.

[3] P. Ekman and W. V. Friesen, “Universals and cultural differences in the judgements of facial expressions of emotion,” Journal of Personality and Social Psychology, vol. 53, 1987