A MINI PROJECT REPORT
ON
Identifying Gender Using Face Recognition

Submitted to Mumbai University

In the partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

In

COMPUTER ENGINEERING

By

Shaikh Safiya Naaz	**(16CO14)**
Mohd Salman Ansari	**(16CO20)**
Shaikh Shafaque Naushad	**(16CO15)**

Under the guidance of

Mr. Muhammed Salman Shamsi

Assistant Professor

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

DECLARATION BY THE CANDIDATE

Shaikh Safiya Naaz, Mohd Salman Ansari, Shaikh Shafaque Nuashad bearing **Roll number: 16CO14**,**16CO20,16CO15** respectively, hereby declare that the mini project report entitled **"Identifying Gender Using Face Recognition"** is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results of this project report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Shaikh Safiya Naaz	**(16CO14)**
Mohd Salman Ansari	**(16CO20)**
Shaikh Shafaque naushad	**(16CO15)**

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

CERTIFICATE

This is to certify that the project report entitled **"Identifying Gender using Face Recognition"**, submitted by **Ms. Shaikh Safiya Naaz, Mr.Mohd Salman Ansari, Ms. Shaikh Shafaque Naushad** bearing **Roll. No: 16CO14,16CO20,16CO15** respectively in the partial fulfillment of the requirements for the award of the degree of **Bachelor of Computer Engineering** is a record of bonafide work carried out by us for the course **Mini Project CSM605**.

Mini Project Guide

(Prof. Muhammed Salman Shamsi)

Mini Project Coordinator

(Prof. Muhammed Salman Shamsi)

Program Owner

(Prof. Tabrez Khan)

INDEX

CONTENTS

<u>CHAPTER 1</u>

INTRODUCTION

1.1 INTRODUCTION

Face is one of the most Important part in human, Because Images of human faces provide a lot of information such as age, gender, expression etc. In the past several years Identifying the gender of human faces using a computer has gained immense attention. Studies have shown that as compare to humans it is difficult for a machine to differentiate between male and female. There exist some distinguishable features between male and female which are used by machine to classify gender. Gender recognition is a pattern recognition problem. This project addresses the problem of gender classification using frontal facial images as training images, where the goal of this project is to automatically detect faces on images quickly and reliably classify the gender of the detected faces.

In our Project we are going to classify Gender using Fisher Faces method with Opencv (Open Computer Vision). The advantage of using Fisher Faces method is, it instead tries to maximize the variance between classes instead of variance with a class. Hence, it is good suited for the Gender Classification task. Fisher faces uses LDA (Linear Discriminant Analysis). LDA performs a class specific dimensionality reduction and was invented by the great statistician Sir R.A Fisher. The LDA maximizes the ratio of between classes to within-classes scatter, instead of maximizing the overall scatter. The classification is restricted to two classes only male and female[4].

1.2 SCOPE

This project “Identifying Gender Using Face Recognition” focuses on improving accuracy of gender classification by analyzing different classification technique and compare their accuracy improvement on gender classification. The classification can be done on only human faces Images. The classification is restricted to only two classes male and female. Additionally, this project only focus on investigate and analyze the use of various classification algorithms, so only frontal facial images were used as training set. But we also have few limitations like the system can’t recognize more than 1 people in the image it predicts wrong results. For Face Recognition it is important to have Image, webcam cannot be used to Recognize faces and classify Gender.

1.3 PROBLEM STATEMENT

The gender recognition is important for many applications such as human computer interaction and computer aided physiological and psychological analysis, since it contains a wide range of information regarding the characteristics difference between male and female.Gender recognition by face is consider one of the active research area of computer vision there are huge numbers of applications where gender recognition can be used in biometric, authentication, high-tech surveillance, security systems, criminology and augmented reality. Given a set of face images labeled with the person gender identify (learning set) and an unlabeled set of face images (test set), seeks to identify each person’s gender in the test images.

CHAPTER 2

SYSTEM SPECIFICATION

2.1 SYSTEM REQUIREMENT

2.1.1 HARDWARE REQUIREMENTS

- Intel Core i3 3rd gen processor or later
- 200Mb disk space
- 1 GB RAM

2.1.2SOFTWARE REQUIREMENTS

- Microsoft Windows XP or later / Ubuntu 12.0 LTS or later /MAC OS 10.1 or later
- Python Interpreter (3.7 or later)
- PyQT5, Tkinter module
- Python OpenCV2, numpy, glob,random,os,tkinter,PyQt5 libraries

2.2 SYSTEM FEATURES

- Adaptive Face detection based on the image scanning
- Classify gender by face detection
- User Friendly-Easy to use
- Best graphical user interface
- Comprehensive user experience
- More accurate model
- Consumes less time for classifications

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

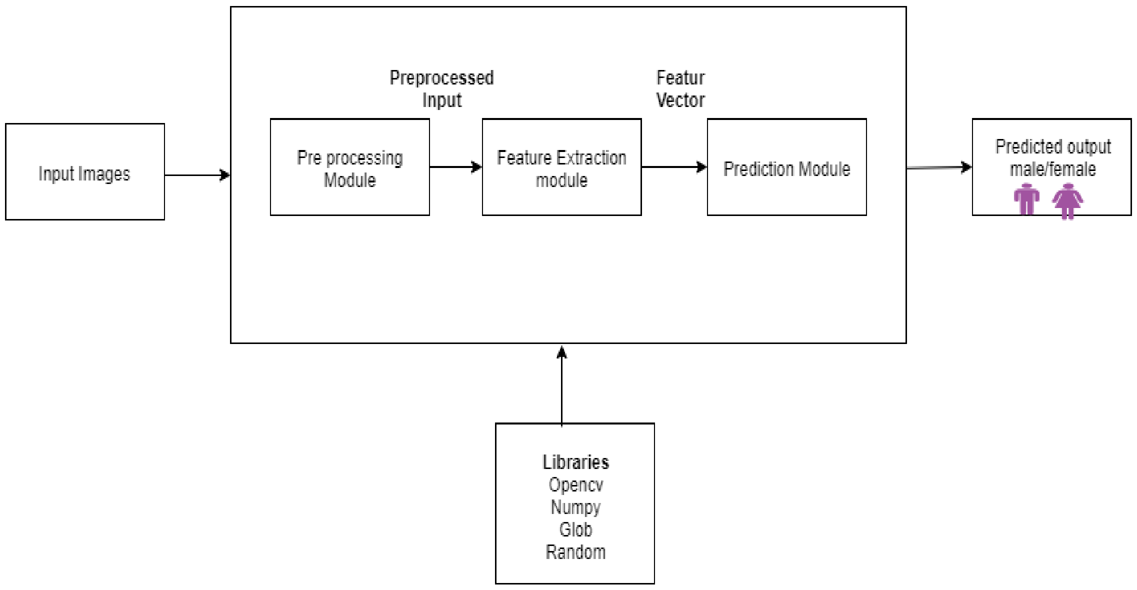


Fig 1: System architecture for Identifying gender using face recognition

3.2 MODULES IN THE SYSTEM

Data Pre-processing: In this module we have used Cascade Classifier Class of OpenCV in which we have used pre trained classifier known as Haar Cascade Classifier for object detection. As the object that is faces are detected we have cropped the faces then We have convert the image into grayscale image and resize an image to an equal size because Fisher Face method works on same input size image.

For rgb to gray

cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

For Resizing an Image

cv2.resize()

OpenCV comes with a function cv.resize() for this purpose. The size of the image can be specified manually, or you can specify the scaling factor.

➢ **Extracting faces:** Now as the object are detected we are going to read each image and then resize them as OpenCV Fisher Faces algorithm works on same input image size. We have used glob module of python for getting paths of file. The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order.

glob. **glob**(*pathname*)

Return a possibly-empty list of path names that match *pathname*, which must be a string containing a path specification. *pathname* can be either absolute (like /usr/src/Python-1.5/Make file) or relative (like ../../Tools/*/*.gif), and can contain shell-style wildcards. Broken sym links are included in the results (as in the shell).

To Read an Image:

- cv2.imread('image path ', 'mode of image')
  Use the function **cv2.imread()** to read an image. The image should be in the working directory or a full path of image should be given.
  Second argument is a flag which specifies the way image should be read.
- cv2.IMREAD_COLOR: Loads a color image. Any transparency of image will be neglected. It is the default flag.
- cv2.IMREAD_GRAYSCALE: Loads image in grayscale mode
- cv2.IMREAD_UNCHANGED: Loads image as such including alpha channel
- Instead of these three flags, you can simply pass integers 1,0 or -1 respectively.

➢ **Training and Testing Module:** In this module we have trained our training data for that we have first split our gender data into training and prediction. We have used Fisher Face Recognizer of OpenCV. Using this recognizer we will used train function and train our data and save it as a model in .xml file. Also we have predict function to predict from recognizer.
Every Face Recognizer supports the:

- **Training** of a Face Recognizer with FaceRecognizertrain() on a given set of images (your face database!).
- **Prediction** of a given sample image, that means a face. The image is given as a Mat.

- **Loading/Saving** the model state from/to a given XML or YAML.
- **Setting/Getting labels info**, that is storaged as a string. String labels info is useful for keeping names of the recognized people.

<u>Face Recognizer train:</u>

<u>Parameters:</u>

- src – The training images, that means the faces you want to learn. The data has to be given as a `vector<Mat>`.
- labels – The labels corresponding to the images have to be given either as a vector.

<u>Face recognizer predict:</u>

Predicts a label and associated confidence (e.g. distance) for a given input image.

<u>Parameters:</u>

src – Sample image to get a prediction from.
label – The predicted label for the given image.
confidence – Associated confidence (e.g. distance) for the predicted label.

- **Displaying Output:** In this module we have created a window on faces by importing WINDOW_NORMAL from cv2. We have used namedWindow function of cv2 to name our window. If it's male it will show Blue colour window if it's female it will show a Red color window with a text above window "Male" or "Female". For Text we have used putText function of opencv.

3.3 Use case Diagram

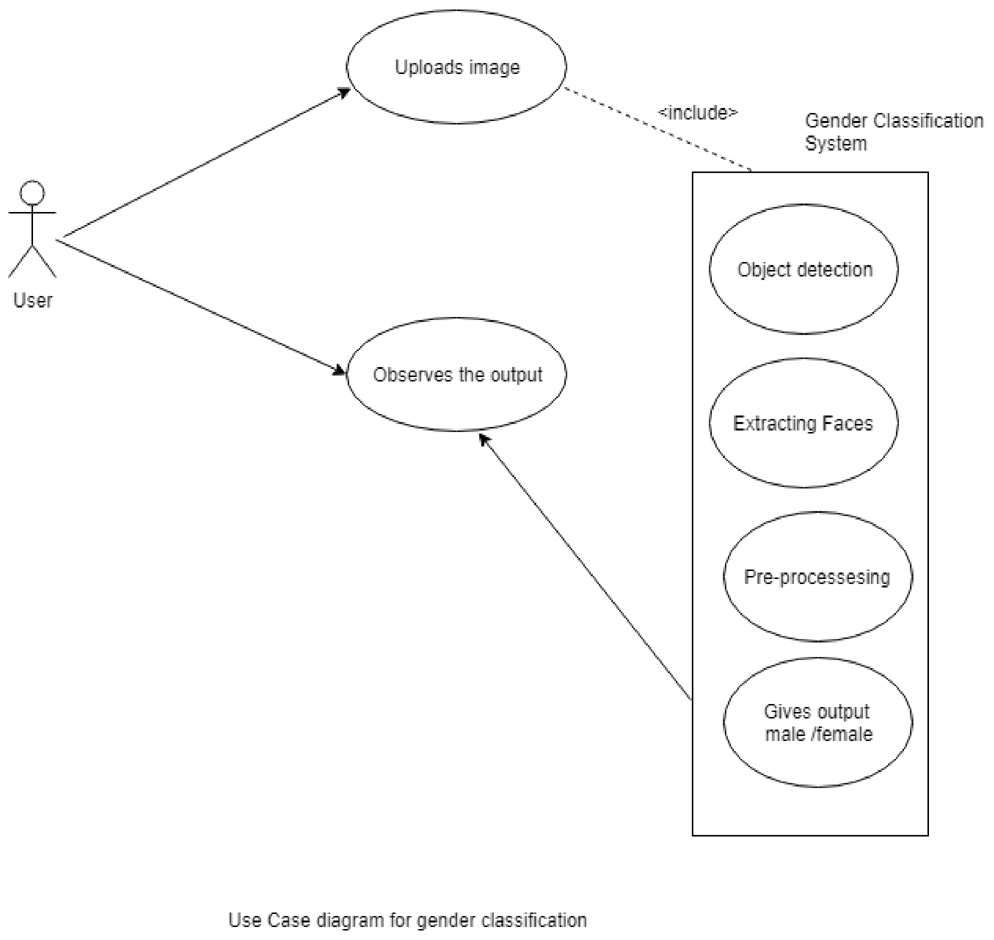

Uploads image
<include>
Gender Classification System
User
Object detection
Observes the output
Extracting Faces
Pre-processesing
Gives output male /female
Use Case diagram for gender classification

Fig 2: Use case diagram for Identifying gender using face recognition

3.4 Activity / Sequence Diagram

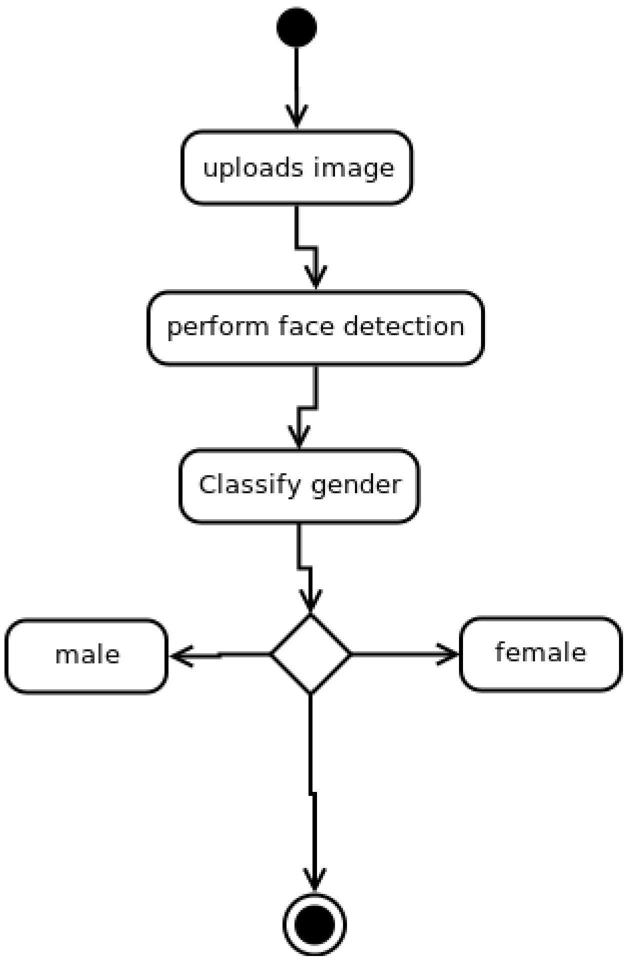


Fig 3: Activity Diagram for Identifying gender using face recognition

CHAPTER 4

IMPLEMENTATION

4.1 CODE SNIPPETS

face_detector.py

```
import cv2


faceCascade = cv2.CascadeClassifier('models/haarcascade_frontalface_alt.xml')


def find_faces(image):
    coordinates = locate_faces(image)
    cropped_faces = [image[y:y + h, x:x + w] for (x, y, w, h) in coordinates]
    normalized_faces = [normalize_face(face) for face in cropped_faces]
    return zip(normalized_faces, coordinates)


def normalize_face(face):
    face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
    face = cv2.resize(face, (350, 350))

    return face;


def locate_faces(image):
    faces = faceCascade.detectMultiScale(
        image,
        scaleFactor=1.1,
        minNeighbors=15,
        minSize=(70, 70)
    )

    return faces
```

data_preprocessing.py

```
import cv2
import glob
import os

from face_detector import find_faces

def remove_face_data():
    print("Removing previous processed faces...")
    filelist = glob.glob("../data/face/assorted/*")
    for file in filelist:
        os.remove(file)

    print("Done!")

def extract_faces(genders):
    print("Extracting faces")
    if not os.path.exists('../data'):
        os.makedirs('../data')
    if not os.path.exists('../data/gender'):
        os.makedirs('../data/gender')
    for gender in genders:
        images = glob.glob('../data/raw_gender/%s/*.jpg' % gender)

        if not os.path.exists('../data/gender/%s' % gender):
            os.makedirs('../data/gender/%s' % gender)
        for file_number, image in enumerate(images):
            frame = cv2.imread(image)
            faces = find_faces(frame)

            for face in faces:
                try:
                    cv2.imwrite("../data/gender/%s/%s.jpg" % (gender, (file_number + 1)),
face[0])
                except:
                    print("Error in processing %s" % image)
    print("Face extraction finished")

if __name__ == '__main__':
    genders = ["female", "male"]
    extract_faces(genders)
```

train_model.py

```
import cv2
import glob
import numpy as np
import random

fisher_face = cv2.face.FisherFaceRecognizer_create()

def get_files(gender, training_set_size):
    files = glob.glob("../data/gender/%s/*" % gender)
    random.shuffle(files)
    training = files[:int(len(files) * training_set_size)]
    prediction = files[-int(len(files) * (1 - training_set_size)):]
    return training, prediction

def make_sets():
    training_data = []
    training_labels = []
    prediction_data = []
    prediction_labels = []
    for gender in genders:
        training, prediction = get_files(gender, 0.8)

        for item in training:
            image = cv2.imread(item, 0)
            training_data.append(image)
            training_labels.append(genders.index(gender))

        for item in prediction:
            item = cv2.imread(item, 0)
            prediction_data.append(image)
            prediction_labels.append(genders.index(gender))
return training_data, training_labels, prediction_data, prediction_labels


def run_recognizer():
    training_data, training_labels, prediction_data, prediction_labels = make_sets()

    print("size of training set is:", len(training_labels), "images")
    print("begin training")
    fisher_face.train(training_data, np.asarray(training_labels))

    print("predicting classification set")
    correct = 0
    for idx, image in enumerate(prediction_data):
        if (fisher_face.predict(image)[0] == prediction_labels[idx]):
            correct += 1

    percentage = (correct * 100) / len(prediction_data)

    return correct, percentage

if __name__ == '__main__':
    genders = ["female", "male"]
    correct, percentage = run_recognizer()
    ("Processed ", correct, " data correctly")
    fisher_face.write('models/gender_classifier_model.xml')
```

gui.py

```
from PyQt5.QtGui import QImage
from PyQt5.QtGui import QPixmap
from PyQt5 import QtCore,QtGui,QtWidgets,uic
from tkinter import filedialog
from tkinter import *
import tkinter as tk
import keyboard
import cv2
import numpy as np
import os.path
from cv2 import WINDOW_NORMAL
from facedetection import find_faces

ESC = 27
name=""

class MyWindow(QtWidgets.QMainWindow):
	def __init__(self):
		super(MyWindow,self).__init__()
		uic.loadUi('upload.ui',self)
		self.upload.clicked.connect(self.open)
		self.webcam.clicked.connect(self.web)

	def open(self):
		root=Tk()
		root.withdraw()
		root.filename =  filedialog.askopenfilename(initialdir = "/",title = "Select 
file",filetypes = (("Image File","*.jpg"),("all files","*.*")))
		name=root.filename

		fisher_face_gender = cv2.face.FisherFaceRecognizer_create()
		fisher_face_gender.read('models/gender_classifier_model.xml')
		run_loop = True
		if os.path.isfile(name):
```

```
                    print('true')
                    image = cv2.imread(name, 1)

                    image1 = cv2.imread(name, 1)
                    image1=cv2.resize(image1,(250, 241),cv2.COLOR_BGR2RGB)
                    image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
                    height2, width2, channel2 = image1.shape
                    step2 = channel2 * width2
            qImg2 = QImage(image1.data, width2, height2, step2, QImage.Format_RGB888)
                    self.label.setPixmap(QPixmap.fromImage(qImg2))

                    for normalized_face, (x, y, w, h) in find_faces(image):
                        gender_prediction =
        fisher_face_gender.predict(normalized_face)
                        if (gender_prediction[0] == 0):
                            cv2.rectangle(image, (x,y), (x+w, y+h), (0,0,255), 2)
                        else:
                            cv2.rectangle(image, (x,y), (x+w, y+h), (255,0,0), 2)
                        if (gender_prediction[0] == 0):
                            self.textBrowser.setText("Female")
                        else:
                            self.textBrowser.setText("Male")

                    key = cv2.waitKey(0)
                    if key == ESC:
                        cv2.destroyWindow(window_name)

                else:
                    print('false')
            def web(self):
              model_gender = cv2.face.FisherFaceRecognizer_create()
              model_gender.read('models/gender_classifier_model.xml')
              window_size=(1280, 720)
              window_name='live'
              update_time=1
              cv2.namedWindow(window_name, WINDOW_NORMAL)
              if window_size:
```

```
        width, height = window_size
        cv2.resizeWindow(window_name, width, height)

    video_feed = cv2.VideoCapture(0)
    video_feed.set(3, width)
    video_feed.set(4, height)
    read_value, webcam_image = video_feed.read()

    delay = 0
    init = True
    while read_value:
        read_value, webcam_image = video_feed.read()
        for normalized_face, (x, y, w, h) in find_faces(webcam_image):
            if init or delay == 0:
                init = False
                gender_prediction = model_gender.predict(normalized_face)
            if (gender_prediction[0] == 0):
                cv2.rectangle(webcam_image, (x,y), (x+w, y+h), (0,0,255), 2)
            else:
                cv2.rectangle(webcam_image, (x,y), (x+w, y+h), (255,0,0), 2)
            text1="Male"
            text2="Female"
            if (gender_prediction[0] == 0):
                cv2.putText(webcam_image,text2,(x,y-10),
cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2,cv2.LINE_AA)
            else:
                cv2.putText(webcam_image,text1,(x,y-10),
cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2,cv2.LINE_AA)
            delay += 1
            delay %= 20
            cv2.imshow(window_name, webcam_image)
            key = cv2.waitKey(update_time)
            if key == ESC:
                break


    cv2.destroyWindow(window_name)
```

```
if __name__=='__main__':
import sys
app=QtWidgets.QApplication(sys.argv)
window=MyWindow()
window.show()
print(name)
sys.exit(app.exec())
```

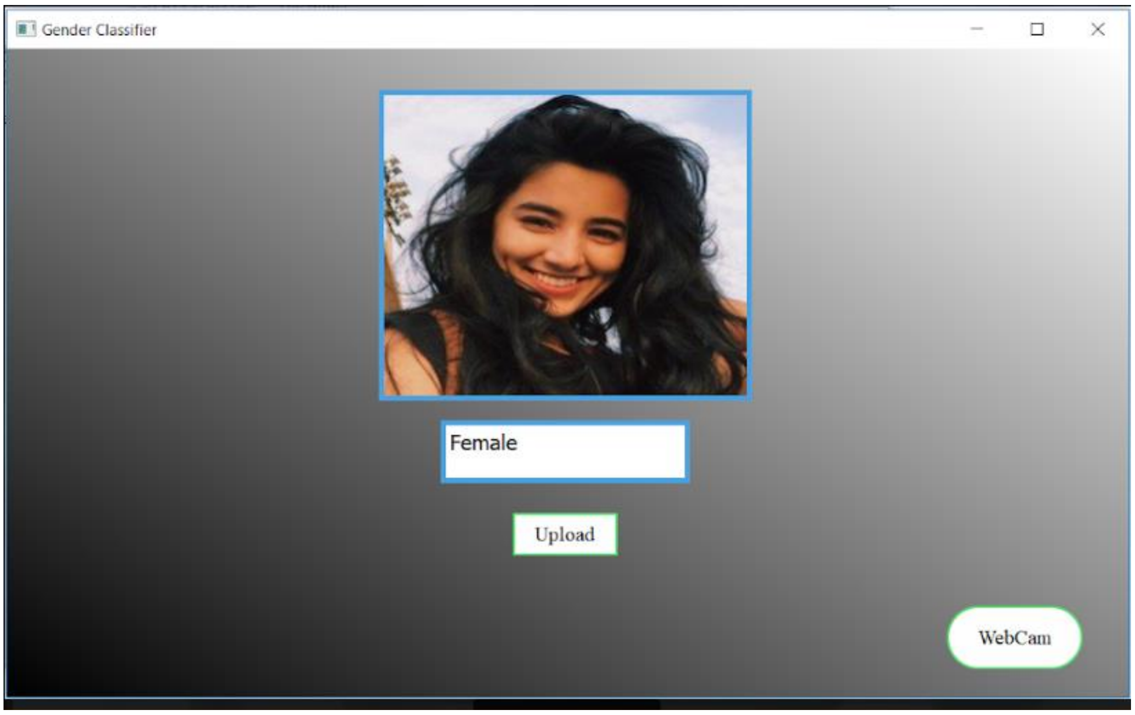
upload.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1067</width>
    <height>623</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Gender Classifier</string>
  </property>
  <property name="autoFillBackground">
   <bool>false</bool>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: qlineargradient(spread:pad, x1:0, y1:1, x2:1, y2:0,
stop:0 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255, 255));</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="upload">
    <property name="geometry">
     <rect>
```

```
      <x>490</x>
      <y>450</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">QPushButton#upload{
color:black;
background-color:#fefefe;
border: 2px solid #58e870; border-radius:30px;
font: 12pt &quot;TImes New Roman&quot;;
}
QPushButton#upload:hover
{
   background-color:red;
            color:white;
}</string>
    </property>
    <property name="text">
     <string>Upload</string>
    </property>
   </widget>
   <widget class="QLabel" name="label">
    <property name="geometry">
     <rect>
      <x>360</x>
      <y>40</y>
      <width>361</width>
      <height>301</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border: 5px solid #4ba1df;
background-color: rgb(255, 255, 255);
color:white;
font-size:20px;</string>
```

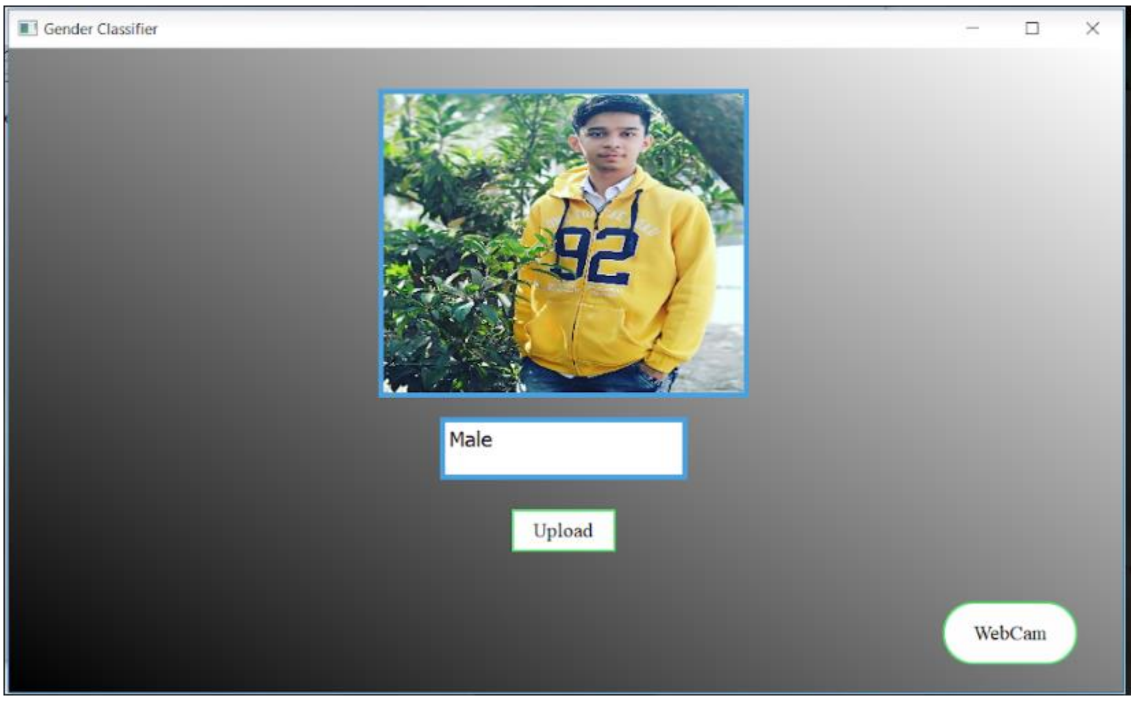
```
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QTextBrowser" name="textBrowser">
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>360</y>
      <width>241</width>
      <height>61</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border: 5px solid #4ba1df;
background-color: rgb(255, 255, 255);
color:black;
font-size:20px;</string>
    </property>
   </widget>
  </widget>
  <action name="actionUpload_Image">
   <property name="text">
    <string>Upload Image</string>
   </property>
  </action>
  <action name="actionExit">
   <property name="text">
    <string>Exit</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```

4.2 SCREENSHOTS



Female

Fig 4: Output for Female



Male

Fig 5: Output for Male

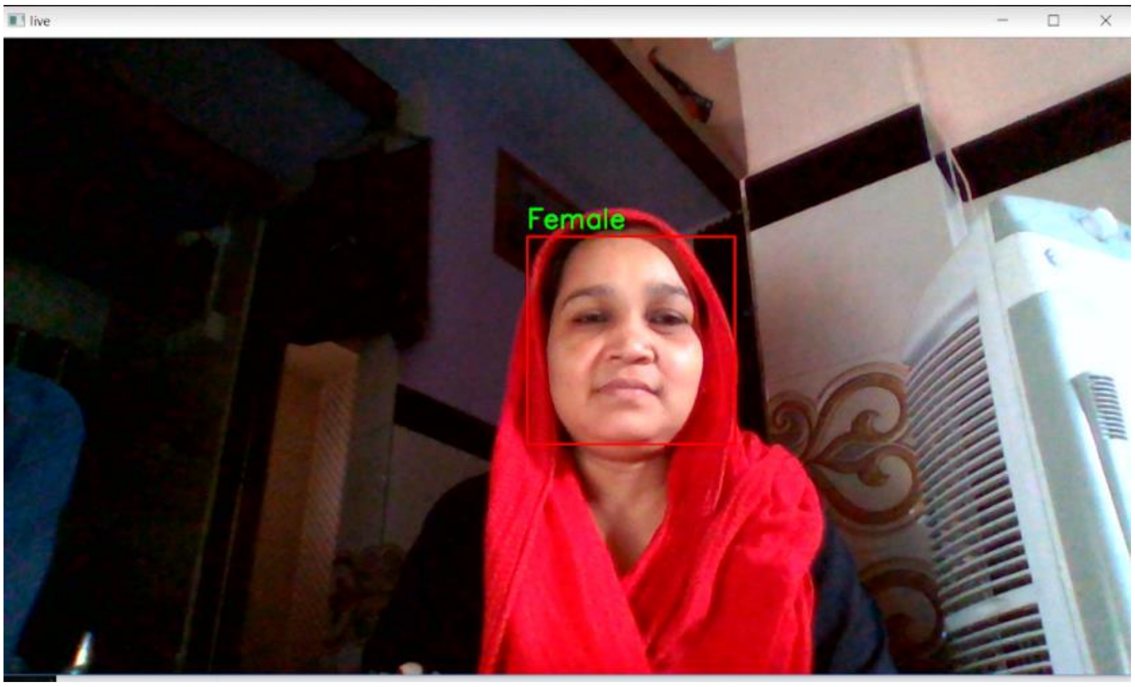


Fig 6: Output for Female using Webcam

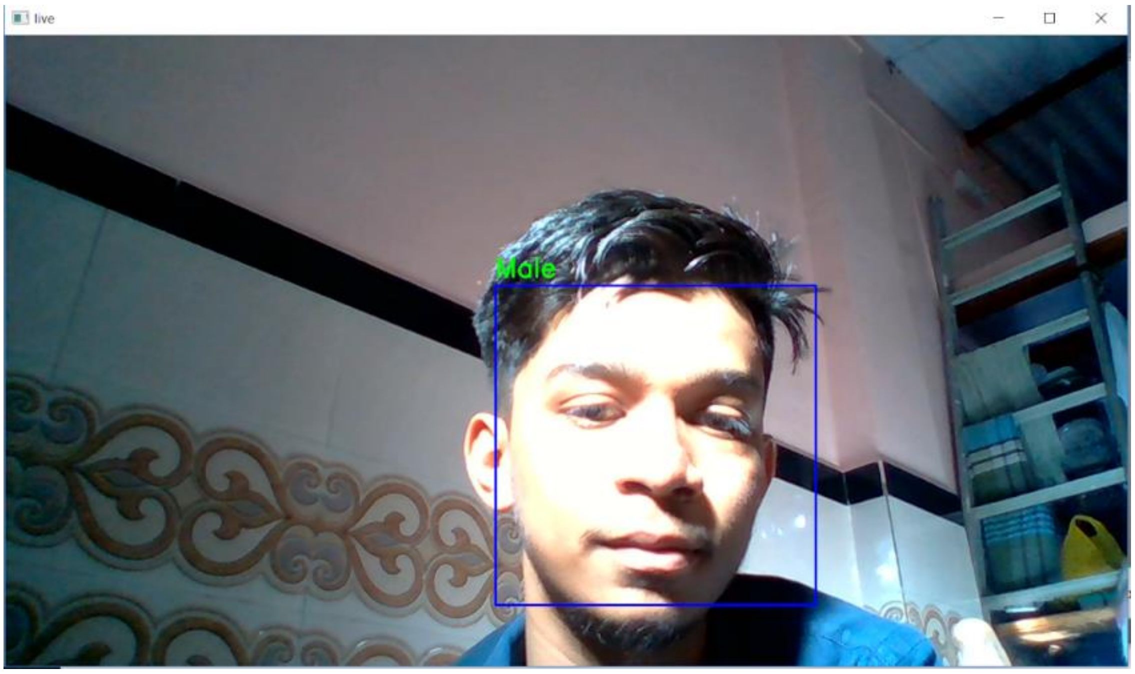


Fig 4: Output for male using Webcam

CHAPTER 5

CONCLUSION

5.1 Conclusion:

In this project we have discussed the application of gender classification and how facial features can play a important role and help us to classify gender using images. Also we can conclude that this is one method with some accuracy for classification, We can try many other methods with fine accuracy.

5.2 Future Scope:

- Being able to distinguish people by their gender or age, would be of good use in the sale of product. You could customize ads in stores depending on the gender.
- In the forensic field, where often only the description of a suspect is available, usually the data that is always available are the gender and an age range, which would be very useful in reducing the list of candidates to be compared with the sketches to get the answers to be more precise.
- Gender classification can be helpful and useful for marketing in future.
- It can be use to improve usability of devices, software or services
- For example, app, music can recommend based on gender.

REFERENCES:

[1] IEEE paper related to Eigen faces vs Fisher Faces
"https://cseweb.ucsd.edu/classes/wi14/cse152-a/fisherface-pami97.pdf"

[2] Gender Classification with OpenCv documentation
"https://docs.opencv.org/2.4/modules/contrib/doc/facerec/tutorial/facerec_gender_classification.html"

[3] Reasearch Paper on Gender Classification of Human Faces
"https://www.researchgate.net/publication/307512589_Gender_Classification_Of_Human_Faces"

[4] Gender recognition using PCA and LDA with improve preprocessing
"https://www.researchgate.net/publication/320652523_Gender_recognition_using_PCA_and_LDA_with_improve_preprocessing_and_classification_technique"

[5]OpenCv Python Tutorial
"https://youtu.be/PmZ29Vta7Vc"