Facial Recognition Attendance System

Hasan Usmani 2020UEA6561

EACPC09 - Machine Learning & AI

Lab Project

Submitted to - MS. Pratima Singh

Netaji Subhas University of Technology

Sector-3, Dwarka, New Delhi – 110078

Table of Content

Abstract

Managing attendance can be a tedious job when implemented by traditional methods like calling out roll calls or taking a student's signature. A smart and authenticated attendance system needs to be implemented to solve this issue. Generally, biometrics such as face recognition, fingerprint, DNA, retina, iris recognition, hand geometry, etc. are used to execute smart attendance systems. The face is a unique identification of humans due to their distinct facial features. Face recognition systems are helpful in many real-life applications. In the proposed method, all the students will initially be enrolled by storing their facial images with a unique ID. At the time of attendance, real-time images will be captured, and the faces in those images will be matched with the faces in the pre-trained dataset. The Haar cascade algorithm is used for face detection. The local Binary Patterns Histogram (LBPH) algorithm is used for face recognition and training of the stored dataset, which generates the histogram for stored and real-time images. The difference between histograms of real-time images & dataset images is calculated for face recognition. The lower difference gives the best match, displaying that student's name & roll number. Attendance of the student is automatically updated in the excel sheet.

Keywords — Face detection & recognition, Haar Cascade Classifier, Local binary pattern histogram (LBPH).

Chapter 1

INTRODUCTION:

Face recognition is a major challenge encountered in multidimensional visual model analysis and is a hot area of research. The art of recognizing the human face is quite difficult as it exhibits varying characteristics like expressions, age, change in hairstyle, etc. Face recognition plays a crucial role in applications such as security systems, credit card verification, identifying criminals in airports, railway stations, etc. Although many methods have been proposed to detect and recognize human faces developing a computational model for a large database is still a challenging task. Face recognition is considered a high-level computer vision task in which techniques can be developed to achieve accurate results. Few popular methods known for face recognition are neural network group-based tree, neural nets, artificial neural networks, and principal component analysis. The proposed methodology is implemented in two stages. Since the human face can be identified by certain facial characters in the first step, the relevant features from the facial images are extracted. They are then quantized to be easy to recognize the facial forms. For face detection Viola-Jones algorithm, which works on Haar features, and Ada boost classifier as a modifier is used. To recognize the face detected, a fusion of classifier algorithms and artificial neural networks are used to obtain accurate results. The proposed methodology aims to detect the face in an image and identify the person using a standard image database with better efficiency and accuracy than the existing methods.

Problem Statement

Traditional student attendance marking techniques is often facing a lot of trouble. The face recognition attendance system emphasizes simplicity by eliminating classical student attendance marking techniques such as calling student names or checking respective identification cards. There are not only disturbing the teaching process but also cause a distraction for students during exam sessions. An attendance sheet is passed around the classroom during the lecture sessions, apart from calling names. The lecture class, especially the class with many students, might find it challenging to have the attendance sheet being passed around the class. Thus, a face recognition attendance system is proposed to replace the manual signing in students' presence, which is burdensome and causes students to get distracted to sign for their attendance. Furthermore, the face recognition-based automated student attendance system can overcome the problem of fraudulent approach, and lecturers do not have to count the number of students several times to ensure the presence of the students. The previous papers proposed by various authors have listed the difficulties of facial identification. One of the difficulties of facial identification is identifying between known and unknown images. Another discovery was that the training process for the face recognition student attendance system is slow and time-consuming. It was also mentioned that different lighting and head poses are often the problems that could degrade the performance of a face recognition-based student attendance system. Hence, there is a need to develop a real-time student attendance system which means the identification process must be done within defined time constraints to prevent omission. The extracted features from facial images representing the students' identity must be consistent with a change in background, illumination, pose, and expression. High accuracy and fast computation time will be the evaluation points of the performance.

Aims and Objectives

The objective of this project is to develop a face recognition attendance system. Expected achievements to fulfill the objectives are:

• To detect the face segment from the video frame.
• To extract the useful features from the face detected.
• To classify the features in order to recognize the face detected.
• To record the attendance of the identified student.

Flow Chart

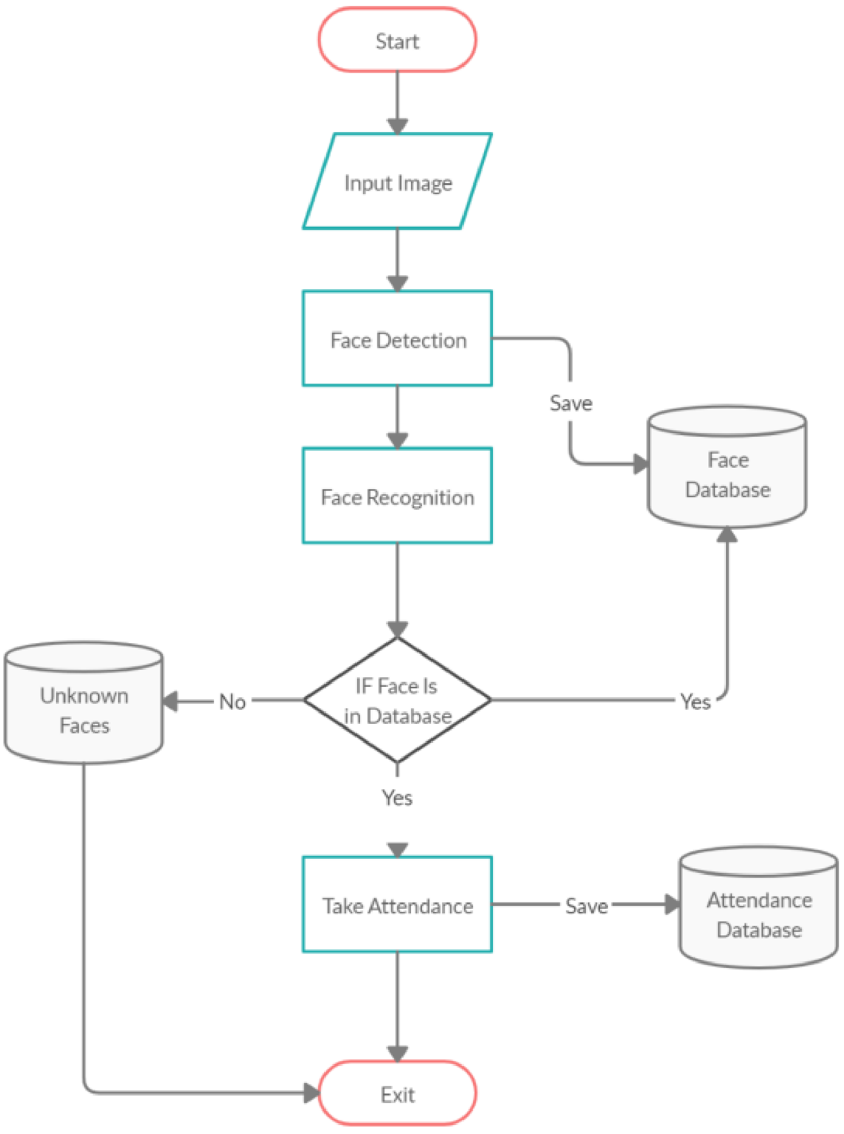


Fig: Flow Chart

Scope of the project

The main intention of this project is to solve the issues encountered in the old attendance system while reproducing a brand new innovative smart system that can provide convenience to the institution

For the initial development phase, we have divided our project into three modules. First, we experimented with some facial detection and recognition techniques discussed in research papers to study and find the best approach for our final model. In the second module, we developed our project using the Haar cascade technique for facial detection and LBPH for facial recognition. In the third module, we have used transfer learning to develop a more accurate model. Our project presents a prototype of the main idea, which is to take real-time attendance using facial recognition. So currently, we have used the laptop's inbuilt camera, which can later be replaced with CCTV cameras and other camera modules. For the GUI, we have used Tkinter to make it easier for the user to go through the model while experimenting with the model.

Apart from that, an excel sheet is created which shows the students attendance and is directly mailed to the respected faculty.

Chapter 2

Image Representation in a Digital Computer

An image is a 2-Dimensional light intensity function. A digital image is discretized both in spatial coordinates by grids and in brightness by quantization.. Effectively, the image can be represented as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point. These elements are referred to as pixels or pels. Typically following image processing applications, the image size used is $\mathbf{256} \times \mathbf{256}$, elements, $\mathbf{640} \times \mathbf{480}$ pels or $\mathbf{1024}\times \mathbf{1024}$ pixels. Quantization of these matrix pixels is done at 8 bits for black and white images and 24 bits for colored images (because of the three color planes Red, Green and Blue each at 8 bits).

Digital Image Processing

Digital Image Processing is the processing of images which are digital in nature by a digital computer. Digital image processing techniques are motivated by three major applications mainly:
• Improvement of pictorial information for human perception
• Image processing for autonomous machine application
• Efficient storage and transmission

Face Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content.

Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a three dimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images. Face Recognition is therefore simply the task of identifying an already detected face as a known or unknown face and in more advanced cases telling exactly whose face it is.

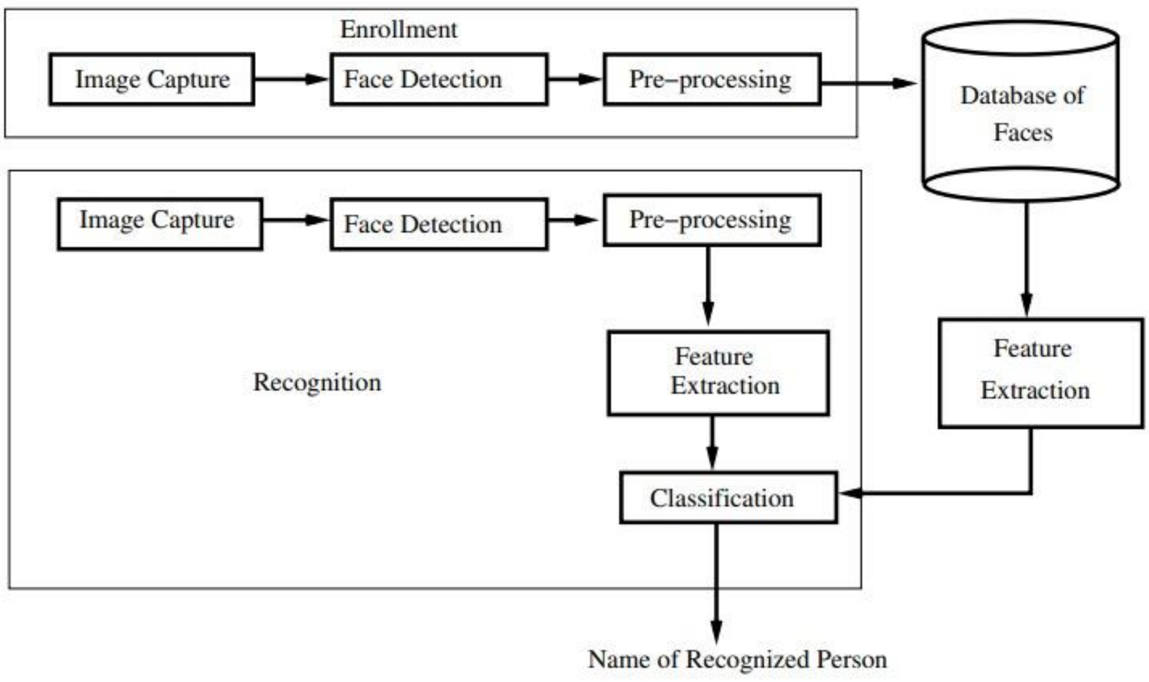


Fig. 1 System Architecture

Viola-Jones Method

Viola-Jones algorithm which was introduced by P. Viola, M. J. Jones is the most popular algorithm to localize the face segment from static images or video frames. Basically, the concept of the Viola-Jones algorithm consists of four parts. The first part is known as Haar feature, the second part is where the integral image is created, followed by the implementation of Adaboost on the third part, and lastly cascading process.

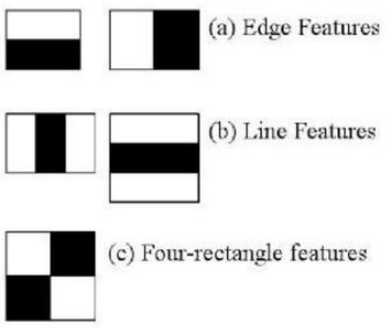
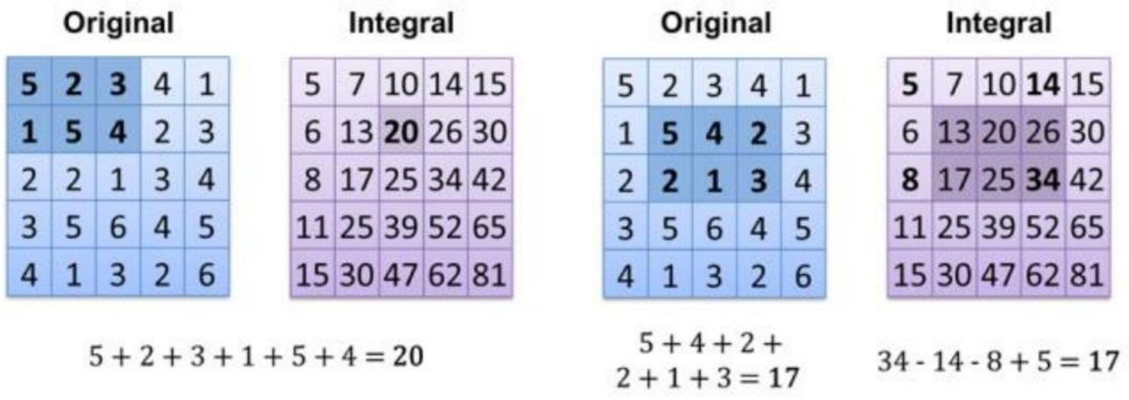


Fig: Haar Feature

The viola-Jones algorithm analyzes a given image using Haar features of multiple The fig shows several types of Haar features. The features perform as window function mapping onto the image. A single value result representing each feature can be computed by subtracting the sum of the white rectangle(s) from the sum of the black rectangle(s).



Original

5	**2**	**3**	4	1
1	**5**	**4**	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	14	15
6	13	**20**	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

Fig: Integral of Image

The value of integrating an image in a specific location is the sum of pixels on the left and the top of the respective location. In order to illustrate clearly, the value of the integral image at location 1 is the sum of the pixels in rectangle A. The values of integral image at the rest of the locations are cumulative. For instance, the value at location 2 is the summation of A and B, (A + B), at location 3 is the summation of A and C, (A + C), and at location 4 is the summation of all the regions (A + B + C + D). Therefore, the sum within the D region can be computed with only addition and subtraction of diagonal at location 4 + 1 − (2 + 3) to eliminate rectangles A, B, and C.

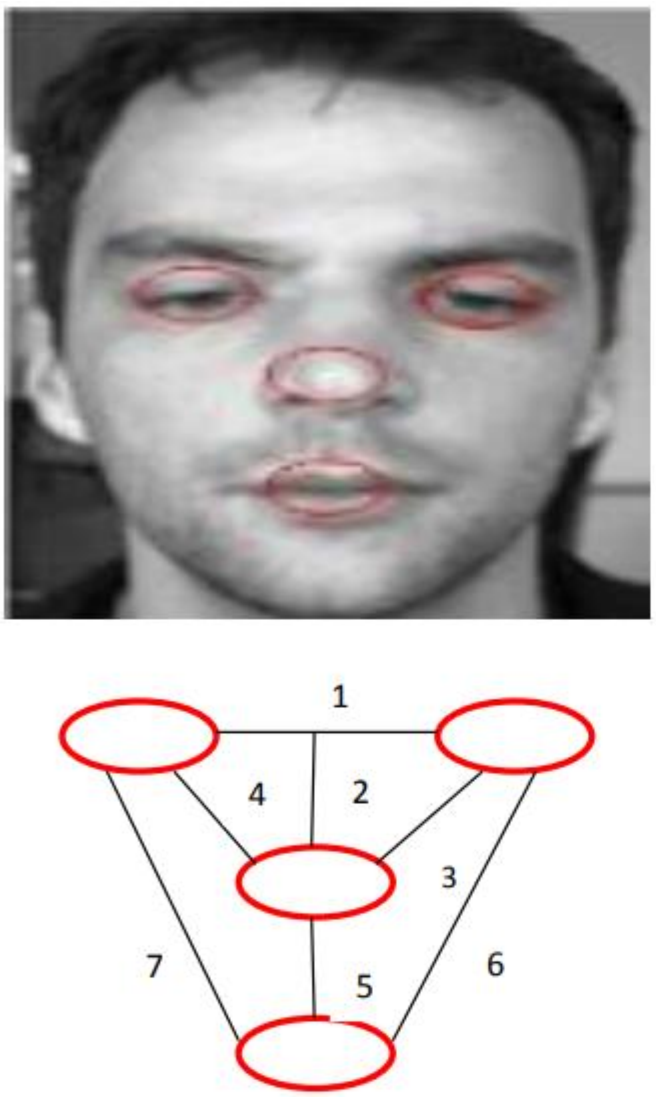


Figure 11: Face feature calculation

Local Binary Patterns

Local Binary Patterns HistogramLocal Binary Pattern (LBP) is simple yet very efficient texture operator labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

Using the LBP
combined with histograms, we can represent the face images with a simple data vector.
LBPH algorithm works step by step:

LBPH algorithm works in 5 steps.
Parameters: the LBPH uses four parameters:
• Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.

• Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
• Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, and the higher the dimensionality of the resulting feature vector. It is usually set to 8.
• Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, and the higher the dimensionality of the resulting feature vector. It is usually set to 8.

Training the Algorithm: First, we need to train the algorithm. We need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the person's name) for each image, so the algorithm will use this
information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.

Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image better by highlighting the
facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors

Principal Component Analysis (PCA)

The first step is to normalize all facets of the training set by removing any common features between these faces so that every face is left with only its unique features. This will be done by removing the average face (mean of pixels over the dataset) from each face.
Our vectors of images will include 64x64=4096 components for each image. These vectors will be created by converting the 2-dimensional image into one vector by aligning the pixels.
From a numerical point of view, this large number of components may be exaggerated to represent such images. To reduce the data size, we will apply the PCA method to select only the main components of our images.
As we have many dimensions in our human faces data, PCA enables us to resume or remove the most correlated components and looks for the directions capturing the maximum variance so that we can get only the most representative components.
The choice of a number of components m will be made according to the best-gotten accuracy when using classifiers on our data. The procedure involves looping over several components and constructing a PCA model for each specified number of principal components. Then we will build a classifier and compute accuracy from the confusion matrix to get the plot from which the best number of components can be chosen.
Once the m eigenfaces are chosen, we can reproduce any face from the training set using these eigenfaces as shown in the picture.

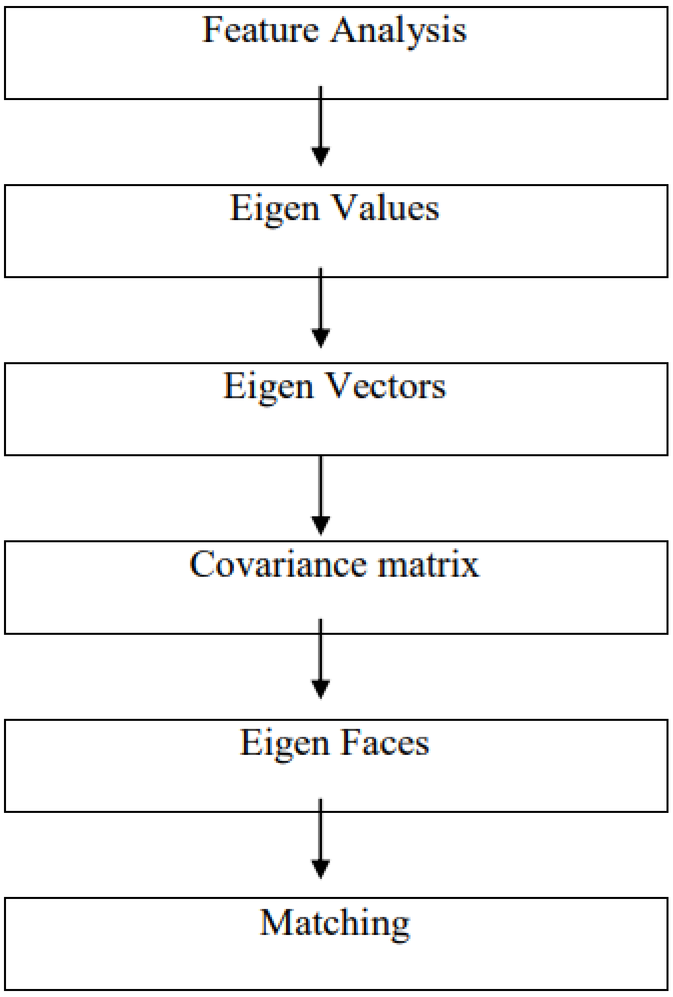
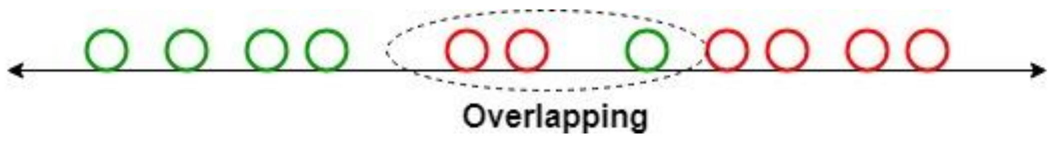


Figure 5: Flow chart of PCA Algorithm

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis or Normal Discriminant Analysis, or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modeling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.
For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification.



Linear Discriminant Analysis as its name suggests is a linear model for classification and dimensionality reduction. Most commonly used for feature extraction in pattern classification problems. This has been here for quite a long time. First, in 1936 Fisher formulated linear discriminant for two classes, and later on, in 1948 C.R Rao generalized it for multiple classes. LDA projects data from a D dimensional feature space down to a D' (D>D') dimensional space in a way to maximize the variability between the classes and reduce the variability within the classes.

The image below shows this procedure:

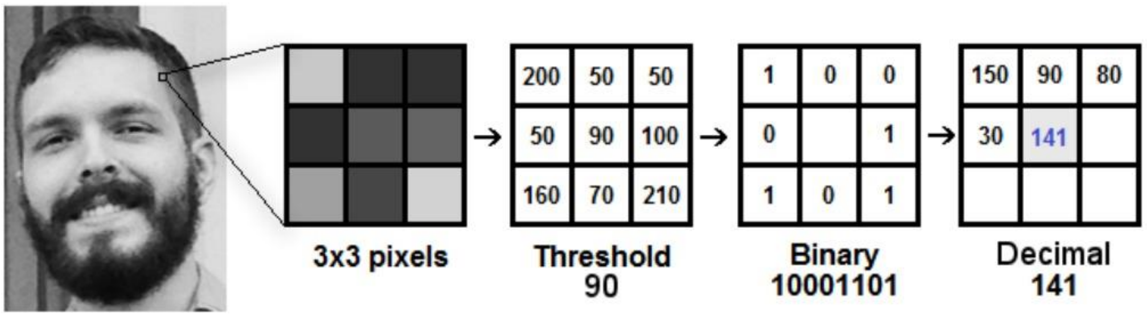
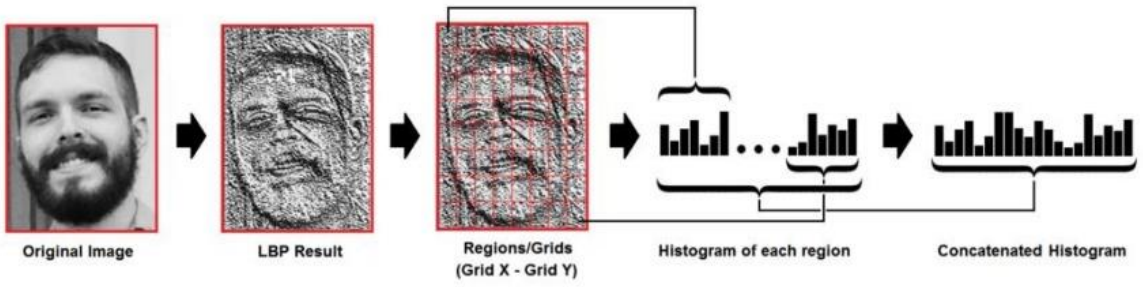


Fig: LBP Operation

Based on the image above, let's break it into several small steps so we can understand it easily:
• Suppose we have a facial image in grayscale.• We can get part of this image as a window of 3x3 pixels.
• It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
• Then, we need to take the central value of the matrix to be used as the threshold.
• This value will be used to define the new values from the 8 neighbors.
• For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
• Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
• Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
• At the end of this procedure (LBP procedure), we have a new image which better represents the characteristics of the original image.

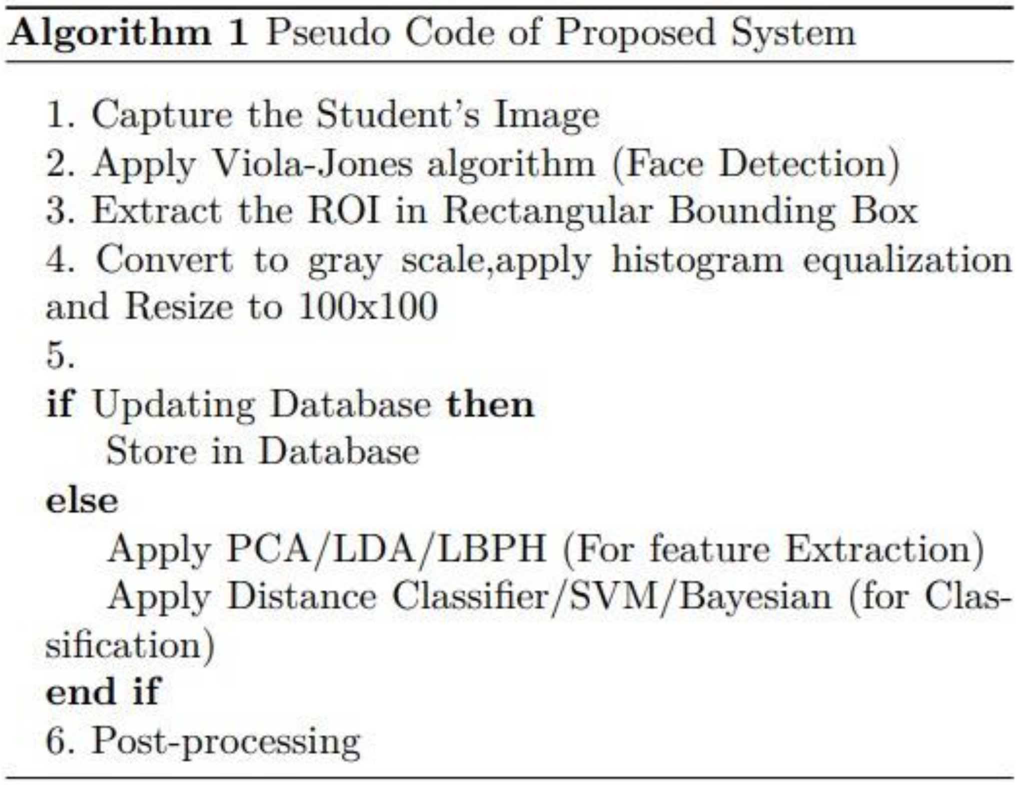
Extracting the Histograms: Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids,as can be seen in the following image:



Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the
Image.•
 So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
• We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance(which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

• So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement. Note: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.• We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower
than the threshold defined.



Algorithm 1 Pseudo Code of Proposed System

```
1. Capture the Student's Image
2. Apply Viola-Jones algorithm (Face Detection)
3. Extract the ROI in Rectangular Bounding Box
4. Convert to gray scale,apply histogram equalization
and Resize to 100x100
5.
if Updating Database then
    Store in Database
else
    Apply PCA/LDA/LBPH (For feature Extraction)
    Apply Distance Classifier/SVM/Bayesian (for Clas-
sification)
end if
6. Post-processing
```

Chapter 3

Post-processing

In the proposed system, after recognizing the faces of the students, the names are updated into an excel sheet. Then the excel sheet is mailed to the administration with the names and roll numbers of the students whose attendance has been marked through yagmail with the help of firebase.

YAGMAIL : Post-processing In the proposed system, after recognizing the faces of the students, the names are updated into an excel sheet. At the end of the class, a provision to announce the names of all students present in the class is also included. This is implemented using text to speech conversion. The system is also equipped with the facility of sending notification mail to the absentees when that facility is enabled.

FIREBASE : is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure.
Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

Module 1-

We have used two notebooks in order to experiment with the available techniques for face detection and face recognition.

Face detection:(Notebook 1)
This notebook is used to detect faces from a picture in a batch. It consists of three functions:
-Detect faces based on opencv haarcascade(viola jones method)
-Detect faces based on opencv deep learning.
-Detect faces combining haarcascade and deep learning.

First, we need to download, Deep neural network module and Caffe models
prototxt file(s)- which define the model architecture (i.e., the layers themselves)
caffemodel file -which contains the weights for the actual layers

Conclusion
Haarcascade finds many faces in the front and background but (dependent of the settings) makes many errors.
Deep learning finds less faces and makes few errors. But it has some trouble finding small faces in the background.
So our idea was to make use of the best of both worlds and use both models. First haarcascade is used to generate possible faces
by using settings that generate many faces but lead to many errors. The candidates are cropped and these cropped images are fed
to the deep learning net. So deep learning receives candidate faces of a large size and with this input can better confirm if the
candidates are real faces. This two steps face detection leads to better results: more faces detected and less errors.

Face Recognition(Notebook 2):
Face recognition concerns the identifying or verifying a person's identity from a visual input like an image or a video frame. It can be formulated as a classification problem, where the inputs are the images and the outputs are the identities or the names of the people
. This notebook analyzes three different techniques for feature extraction in the context of face recognition:
1. PCA (Principal Components Analysis)
2. LDA (Linear Discirminant Analysis)
3. LBP (Local Binary Patterns)
using three different datasets:
1. AT&T Faces dataset
2. Labeled faces of the wild

Conclusion
All three approaches show good performance: with LDA we obtain best results, followed by LBP, and then PCA.The results obtained with LBP are also satisfying and close to LDA ones.

Module 2-

This module consists of all the files which are part of our final working project . We have to run the file named "firstpage.py" in any code editor such as Visual Studio after importing all the necessary libraries.

Module 3-

We have moved on to use the pretrained models available for both face recognition and face detection in order to achieve even better accuracy .
For face detection we used the media pipe API available in python's dlib library-
Mediapipe is a cross-platform/open-source tool that allows you to run a variety of machine learning models in real-time. It's designed primarily for facilitating the use of ML in streaming media & It was built by Google
The mediapipe's face detection solution uses a very lightweight and highly accurate feature extraction network, that is inspired and modified from MobileNetV1/V2 and used a detection method similar to Single Shot MultiBox Detector (SSD). It is capable of running at a speed of 200-1000+ FPS on flagship devices.
They have utilized transfer learning and used both synthetic rendered and annotated real-world data to get a model capable of predicting 3D landmark coordinates. Another approach could be to train a model to

predict a 2D heatmap for each landmark but will increase the computational cost as there are so many points.

For face recognition we used the open cv face recognition library.

CHAPTER 4

Methodology:

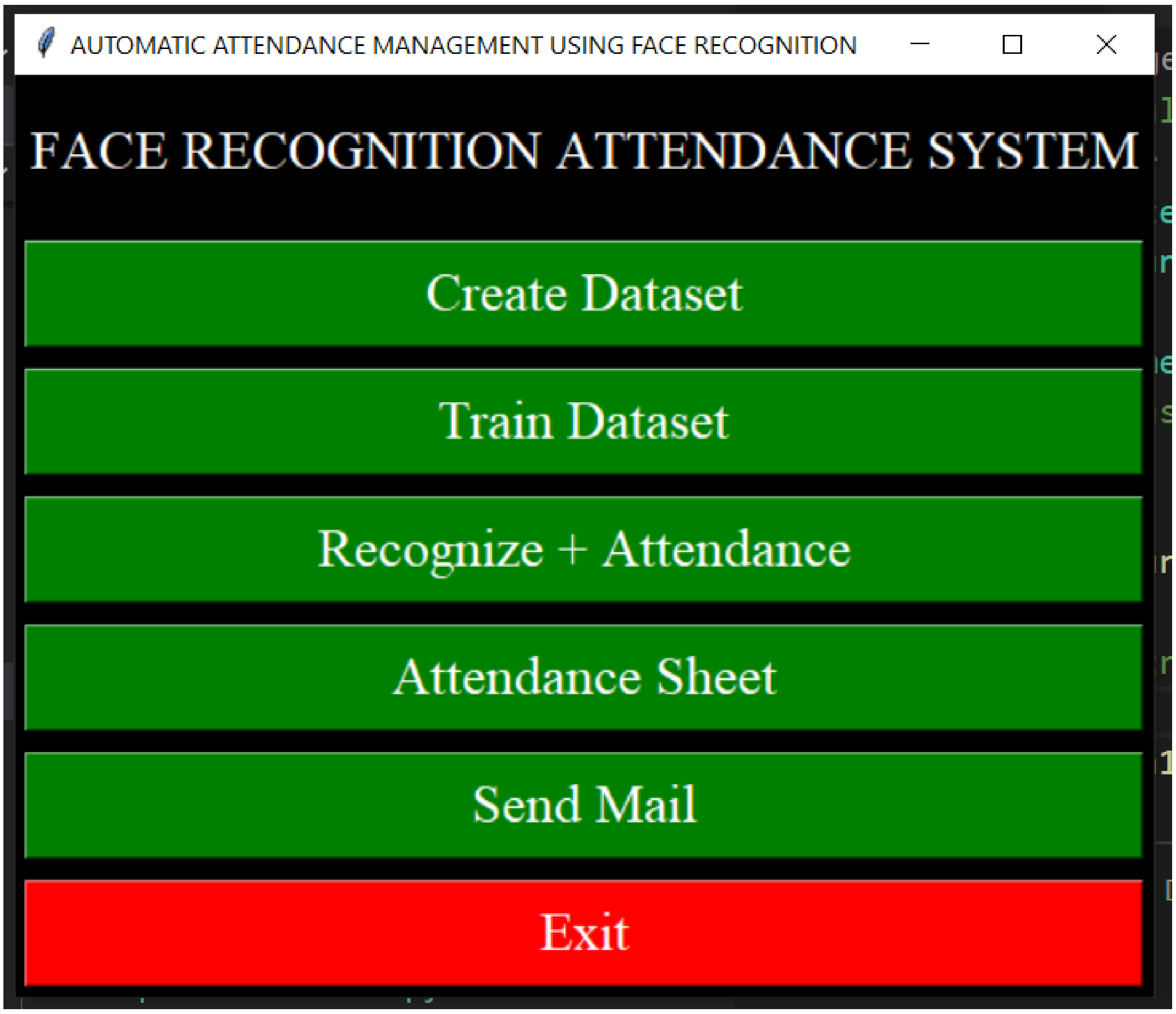
CODE:

Module 2-

firstpage.py-

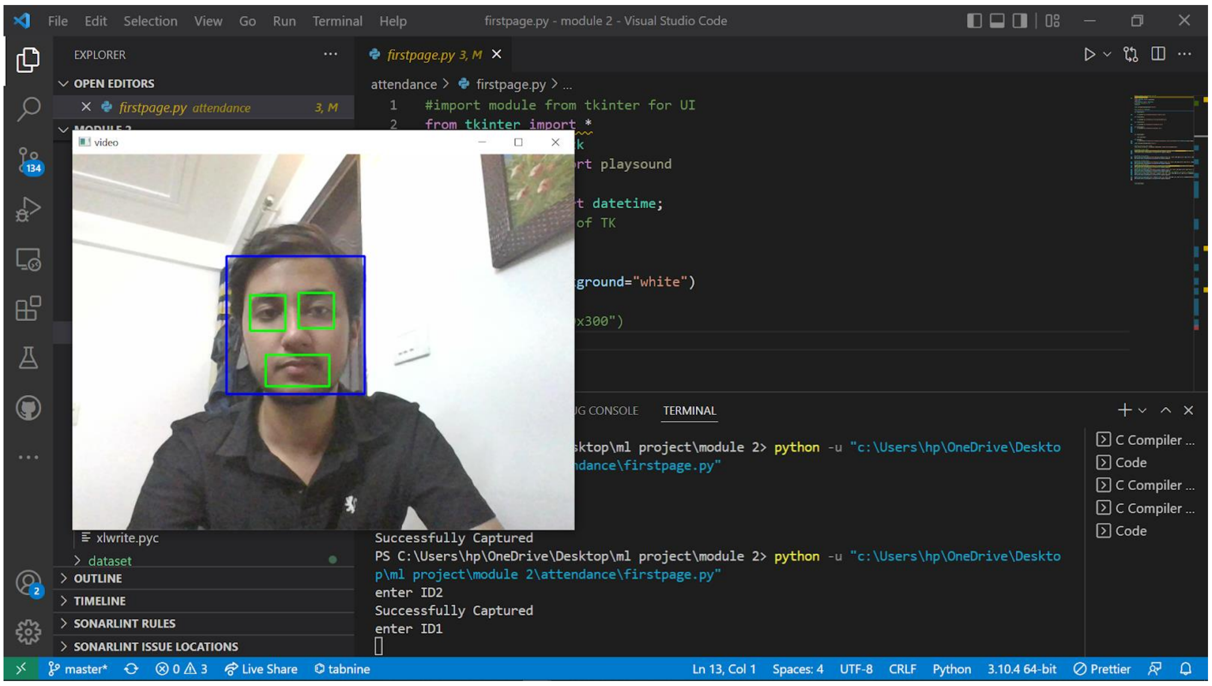
```
#import module from tkinter for UI
from tkinter import *
import tkinter as tk
from playsound import playsound
import os
from datetime import datetime;
#creating instance of TK
root=Tk()


root.configure(background="white")


#root.geometry("300x300")


def function1():

    os.system("py attendance\dataset_capture.py")

def function2():
    os.system("py attendance\\training_dataSet.py")


def function3():


    os.system("py attendance\\recognizer.py")


def function7():
    os.system("py attendance\\automail.py")




def function6():


    root.destroy()


def attend():
    os.startfile("attendance\\firebase\\attendance_files\\attendance"+str(
datetime.now().date())+'.xls')

root.configure(background='black')


#setting title for the window
root.title("AUTOMATIC ATTENDANCE MANAGEMENT USING FACE RECOGNITION")


#creating a text label
lbl2=Label(root, text="FACE RECOGNITION ATTENDANCE SYSTEM",font=("times new
roman",20),fg="white",bg="black",height=2)
lbl2.grid(row=1,columnspan=4,sticky=W+E+N+S,padx=5,pady=5)



#creating first button
bt1=tk.Button(root,text="Create         Dataset",font=("times          new
roman",20),bg="green",fg='white',command=function1)
bt1.grid(row=3,columnspan=4,sticky=W+E+N+S,padx=5,pady=5)



#creating second button
bt2=tk.Button(root,text="Train          Dataset",font=("times          new
roman",20),bg="green",fg='white',command=function2)
bt2.grid(row=4,columnspan=4,sticky=N+E+W+S,padx=5,pady=5)
#creating third button
bt3=tk.Button(root,text="Recognize    +    Attendance",font=('times    new
roman',20),bg="green",fg="white",command=function3)
bt3.grid(row=5,columnspan=4,sticky=N+E+W+S,padx=5,pady=5)
#creating attendance button
bt4=tk.Button(root,text="Attendance        Sheet",font=('times         new
roman',20),bg="green",fg="white",command=attend)
bt4.grid(row=6,columnspan=4,sticky=N+E+W+S,padx=5,pady=5)
```

```
#Button(root,text="Developers",font=('times                    new
roman',20),bg="#0D47A1",fg="white",command=function5).grid(row=8,columnspa
n=2,sticky=N+E+W+S,padx=5,pady=5)
bt6=tk.Button(root,text="Send            Mail",font=('times          new
roman',20),bg="green",fg="white",command=function7)
bt6.grid(row=8,columnspan=4,sticky=N+E+W+S,padx=5,pady=5)


bt5=tk.Button(root,text="Exit",font=('times                          new
roman',20),bg="red",fg="white",command=function6)
bt5.grid(row=9,columnspan=4,sticky=N+E+W+S,padx=5,pady=5)



root.mainloop()
```


AUTOMATIC ATTENDANCE MANAGEMENT USING FACE RECOGNITION
FACE RECOGNITION ATTENDANCE SYSTEM
Create Dataset
Train Dataset
Recognize + Attendance
Attendance Sheet
Send Mail
Exit

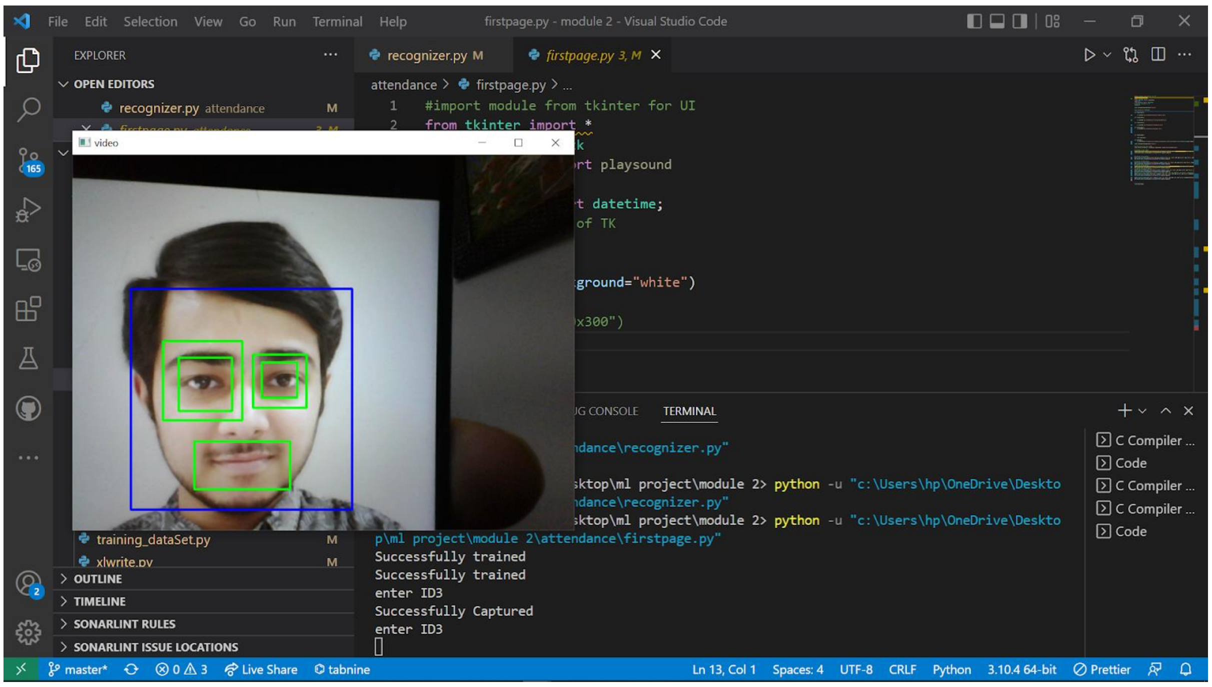
dataset_capture.py-

```
# Import OpenCV2 for image processing
import cv2
import os


def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)





face_id=input("enter ID")
# Start capturing video
vid_cam = cv2.VideoCapture(0,cv2.CAP_DSHOW)


# Detect object in video stream using Haarcascade Frontal Face
face_detector                                                              =
cv2.CascadeClassifier('attendance\haarcascade_frontalface_default.xml')
eyeCascade = cv2.CascadeClassifier('attendance/haarcascade_eye.xml')
smileCascade = cv2.CascadeClassifier('attendance/haarcascade_smile.xml')
# Initialize sample face image
count = 0
assure_path_exists("dataset/")


# Start looping
while(True):


    # Capture video frame
    _, image_frame = vid_cam.read()


    # Convert frame to grayscale
    gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)


    # Detect frames of different sizes, list of faces rectangles
    faces = face_detector.detectMultiScale(gray, 1.3, 5)


    # Loops for each faces
    for (x,y,w,h) in faces:


        # Crop the image frame into rectangle
        cv2.rectangle(image_frame, (x,y), (x+w,y+h), (255,0,0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image_frame[y:y+h, x:x+w]

        eyes = eyeCascade.detectMultiScale(
            roi_gray,
            scaleFactor= 1.5,
            minNeighbors=10,
            minSize=(5, 5),
        )

        smile = smileCascade.detectMultiScale(
            roi_gray,
            scaleFactor= 1.5,
            minNeighbors=15,
            minSize=(25, 25),
            )

        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255,
0), 2)

        for (xx, yy, ww, hh) in smile:
            cv2.rectangle(roi_color, (xx, yy), (xx + ww, yy + hh), (0, 255,
0), 2)

        #cv2.imshow('video', image_frame)

        cv2.imshow('video', image_frame)
        # Increment sample face image
        count += 1


        # Save the captured image into the datasets folder
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) +
".jpg", gray[y:y+h,x:x+w])


        # Display the video frame, with bounded rectangle on the person's
face
       # cv2.imshow('frame', image_frame)


    # To stop taking video, press 'q' for at least 100ms
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break


    # If image taken reach 100, stop taking video
    elif count>=30:
        print("Successfully Captured")
        break


# Stop video
vid_cam.release()


# Close all started windows
cv2.destroyAllWindows()
```


File Edit Selection View Go Run Terminal Help
firstpage.py - module 2 - Visual Studio Code
EXPLORER
OPEN EDITORS
firstpage.py attendance 3, M
video
xlwrite.pyc
dataset
OUTLINE
TIMELINE
SONARLINT RULES
SONARLINT ISSUE LOCATIONS
attendance > firstpage.py > ...
G CONSOLE
TERMINAL
C Compiler ...
Code
master* 0 3 Live Share tabnine
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit Prettier

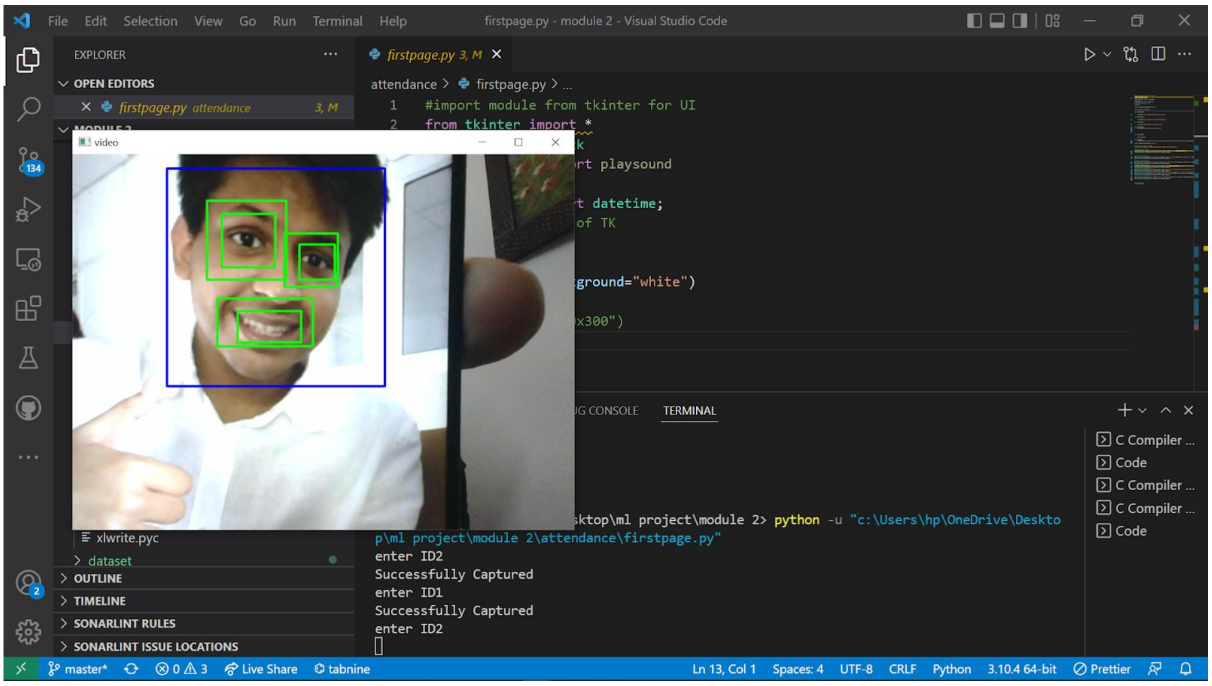
```
#import module from tkinter for UI
from tkinter import *
k
rt playsound

t datetime;
of TK

ground="white")

x300")
```

```
sktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
ndance\firstpage.py"



Successfully Captured
PS C:\Users\hp\OneDrive\Desktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\firstpage.py"
enter ID2
Successfully Captured
enter ID1
```


File Edit Selection View Go Run Terminal Help
firstpage.py - module 2 - Visual Studio Code
EXPLORER
OPEN EDITORS
recognizer.py attendance M
recognizer.py M
firstpage.py 3, M
video
training_dataSet.py M
xlwrite.py M
OUTLINE
TIMELINE
SONARLINT RULES
SONARLINT ISSUE LOCATIONS
attendance > firstpage.py > ...
G CONSOLE
TERMINAL
C Compiler ...
Code
master* 0 3 Live Share tabnine
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit Prettier

```
#import module from tkinter for UI
from tkinter import *
k
rt playsound

t datetime;
of TK

ground="white")

x300")
```

```
ndance\recognizer.py"

sktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
ndance\recognizer.py"
sktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\firstpage.py"
Successfully trained
Successfully trained
enter ID3
Successfully Captured
enter ID3
```



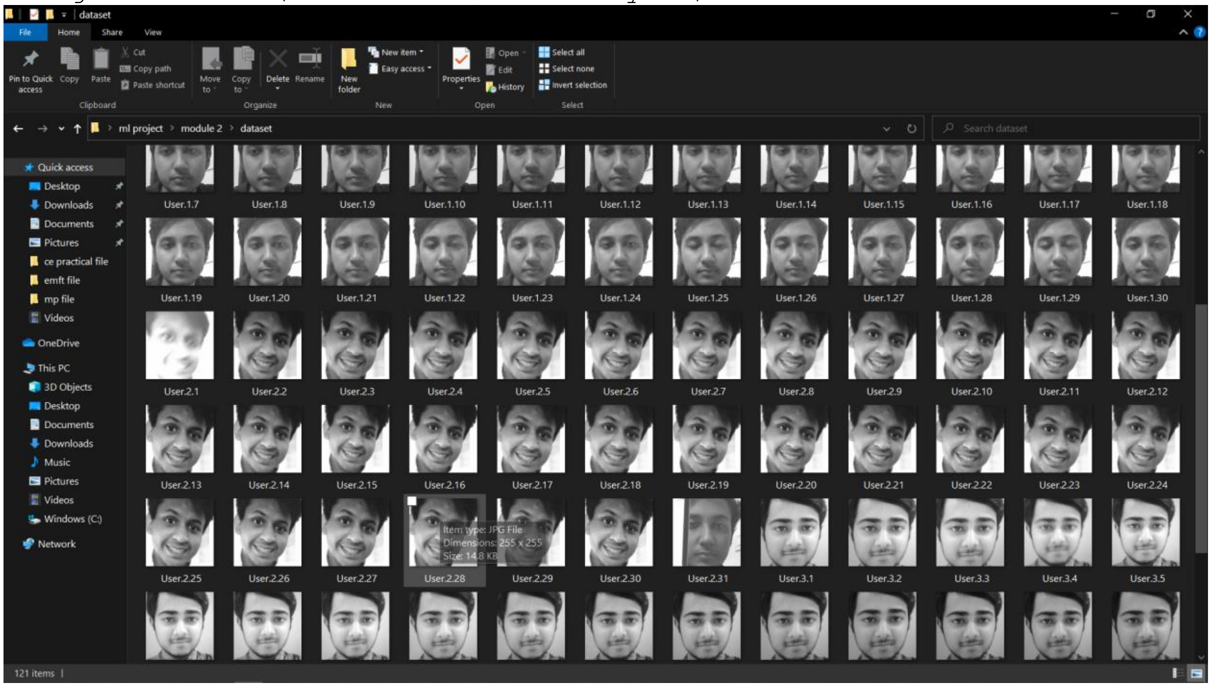
```
#import module from tkinter for UI
from tkinter import *
k
rt playsound

t datetime;
of TK

ground="white")

x300")
```

```
sktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\firstpage.py"
enter ID2
Successfully Captured
enter ID1
Successfully Captured
enter ID2
```

File Edit Selection View Go Run Terminal Help
firstpage.py - module 2 - Visual Studio Code
EXPLORER
OPEN EDITORS
firstpage.py attendance 3, M
video
xlwrite.pyc
dataset
OUTLINE
TIMELINE
SONARLINT RULES
SONARLINT ISSUE LOCATIONS
attendance > firstpage.py > ...
TERMINAL
C Compiler ...
Code
master*
Live Share
tabnine
Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit Prettier

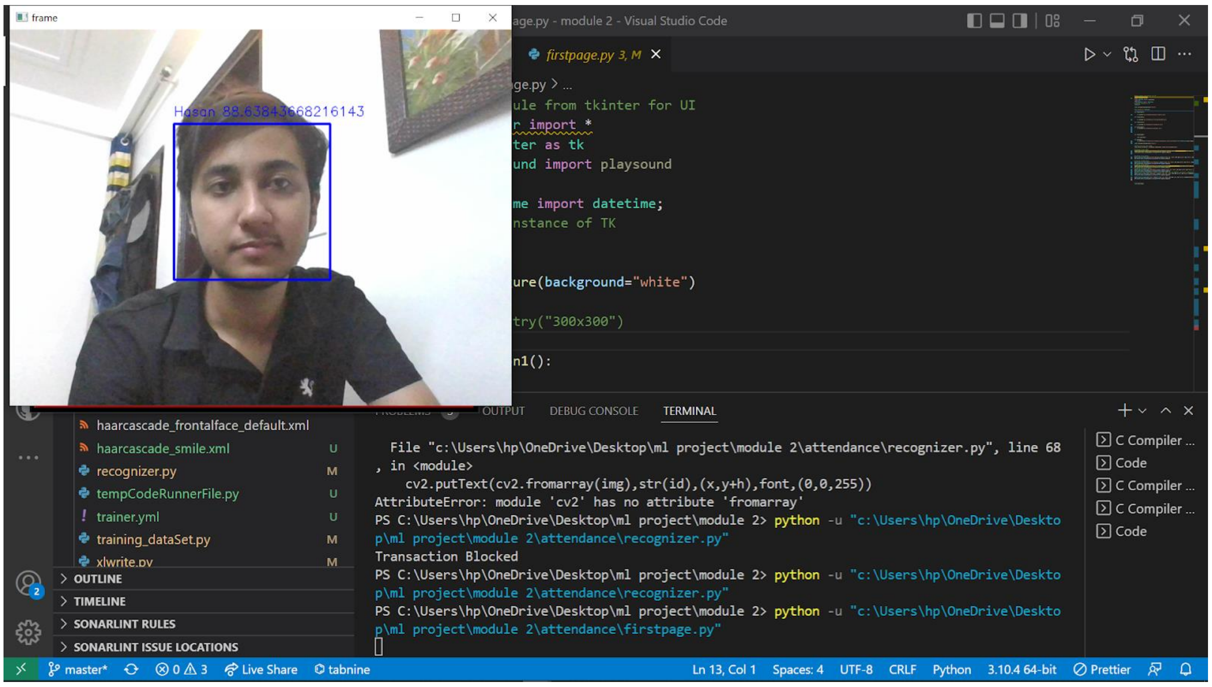
training_dataSet.py-

```
import os,cv2;
import numpy as np
from PIL import Image;


recognizer = cv2.face.LBPHFaceRecognizer_create()
detector=
cv2.CascadeClassifier("attendance/haarcascade_frontalface_default.xml")


def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #create empth face list
    faceSamples=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the
images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image

        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces=detector.detectMultiScale(imageNp)
        #If a face is there then append that in the list as well as Id of
it
        for (x,y,w,h) in faces:
            faceSamples.append(imageNp[y:y+h,x:x+w])
            Ids.append(Id)
    return faceSamples,Ids


faces,Ids = getImagesAndLabels('dataset')
s = recognizer.train(faces, np.array(Ids))
print("Successfully trained")
recognizer.write('attendance/trainer.yml')
```


dataset
File Home Share View
Pin to Quick access Copy Paste Cut Copy path Paste shortcut
Clipboard
Move to Copy to Delete Rename
Organize
New folder New item Easy access
New
Properties Open Edit History
Open
Select all Select none Invert selection
Select
ml project > module 2 > dataset
Search dataset
Quick access
Desktop
Downloads
Documents
Pictures
ce practical file
emft file
mp file
Videos
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Network
User.1.7 User.1.8 User.1.9 User.1.10 User.1.11 User.1.12 User.1.13 User.1.14 User.1.15 User.1.16 User.1.17 User.1.18
User.1.19 User.1.20 User.1.21 User.1.22 User.1.23 User.1.24 User.1.25 User.1.26 User.1.27 User.1.28 User.1.29 User.1.30
User.2.1 User.2.2 User.2.3 User.2.4 User.2.5 User.2.6 User.2.7 User.2.8 User.2.9 User.2.10 User.2.11 User.2.12
User.2.13 User.2.14 User.2.15 User.2.16 User.2.17 User.2.18 User.2.19 User.2.20 User.2.21 User.2.22 User.2.23 User.2.24
User.2.25 User.2.26 User.2.27 User.2.28 User.2.29 User.2.30 User.2.31 User.3.1 User.3.2 User.3.3 User.3.4 User.3.5
Item type: JPG File
Dimensions: 255 x 255
Size: 14.8 KB
121 items

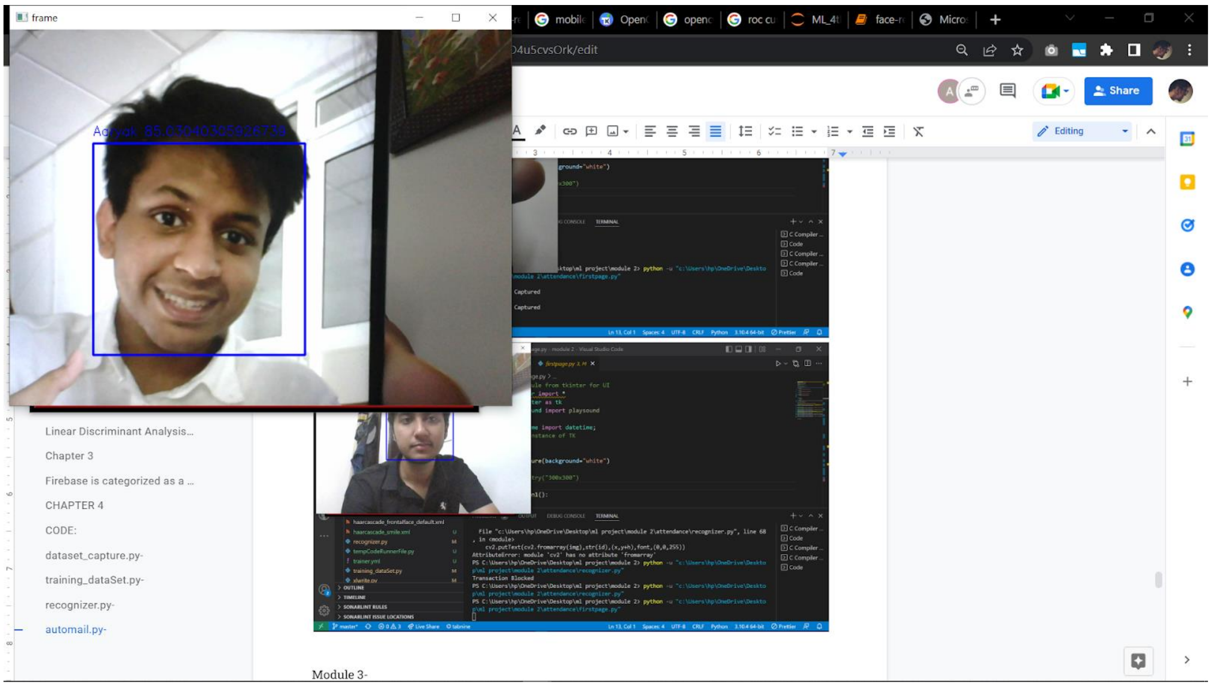
recognizer.py-

```
import cv2, numpy as np;
from firebase import firebase
import xlwrite;
import time
import sys
from playsound import playsound
start=time.time()
period=8


face_cas      =      cv2.CascadeClassifier(cv2.data.haarcascades      +
'haarcascade_frontalface_default.xml')
eye_cascade     =     cv2.CascadeClassifier(cv2.data.haarcascades      +
'haarcascade_eye.xml')
smile_cascade       =       cv2.CascadeClassifier(cv2.data.haarcascades
+'haarcascade_smile.xml')
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW);
recognizer = cv2.face.LBPHFaceRecognizer_create();
recognizer.read('attendance/trainer.yml');
flag = 0;
id=0;
filename='filename';
dict = {
            'item1': 1
}


font = cv2.FONT_HERSHEY_SIMPLEX
while True:
    ret, img = cap.read();
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
    faces = face_cas.detectMultiScale(gray, 1.3, 7)
    for (x,y,w,h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0),2);
        roi_color = img[y:y+h, x:x+w]


        id,conf=recognizer.predict(roi_gray)


        if(conf < 50):
         if(id==1):
            id='Hasan'
            if((str(id)) not in dict):
                filename=xlwrite.output('attendance','class1',1,id,'yes');
                dict[str(id)]=str(id);

         elif(id==2):
            id = 'Aaryak'
            if ((str(id)) not in dict):
                filename  =xlwrite.output('attendance',  'class1',  2,  id,
'yes');
                dict[str(id)] = str(id);


         elif(id==3):
            id = 'Yash'
            if ((str(id)) not in dict):
                filename  =xlwrite.output('attendance',  'class1',  3,  id,
'yes');
                dict[str(id)] = str(id);


         elif(id==4):
            id = 'Vaibhav'
            if ((str(id)) not in dict):
                filename  =xlwrite.output('attendance',  'class1',  4,  id,
'yes');
                dict[str(id)] = str(id);


        else:
             id = 'Unknown, can not recognize'
             flag=flag+1
             break

        cv2.putText(img,str(id)+"                        "+str(conf),(x,y-
10),font,0.55,(120,255,120),1)
        #cv2.cv.PutText(cv2.cv.fromarray(img),str(id),(x,y+h),font,(0,0,25
5));
    cv2.imshow('frame',img);
    #cv2.imshow('gray',gray);
    if flag == 10:
        print("Transaction Blocked")
        break;
    if time.time()>start+period:
        break;
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break;


cap.release();
cv2.destroyAllWindows();
```



```
ule from tkinter for UI
r import *
ter as tk
und import playsound

me import datetime;
nstance of TK


ure(background="white")

try("300x300")

n1():
```

```
  File "c:\Users\hp\OneDrive\Desktop\ml project\module 2\attendance\recognizer.py", line 68
, in <module>
    cv2.putText(cv2.fromarray(img),str(id),(x,y+h),font,(0,0,255))
AttributeError: module 'cv2' has no attribute 'fromarray'
PS C:\Users\hp\OneDrive\Desktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\recognizer.py"
Transaction Blocked
PS C:\Users\hp\OneDrive\Desktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\recognizer.py"
PS C:\Users\hp\OneDrive\Desktop\ml project\module 2> python -u "c:\Users\hp\OneDrive\Deskto
p\ml project\module 2\attendance\firstpage.py"
```



Linear Discriminant Analysis...

Chapter 3

Firebase is categorized as a ...

CHAPTER 4

CODE:

dataset_capture.py-

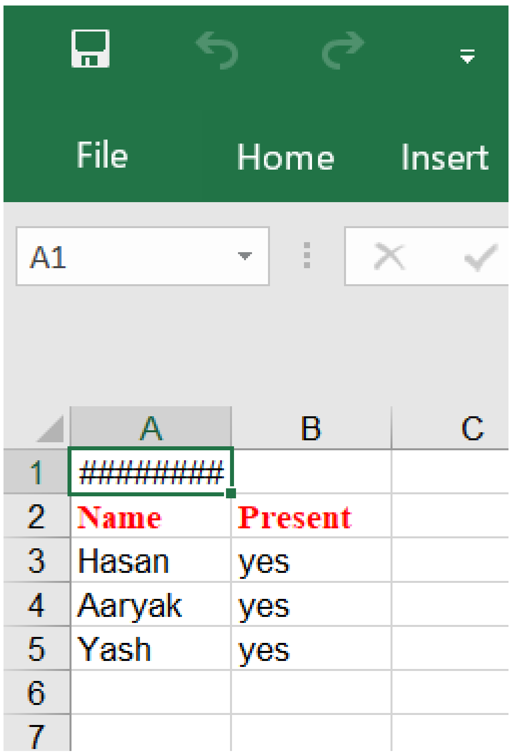
training_dataSet.py-

recognizer.py-

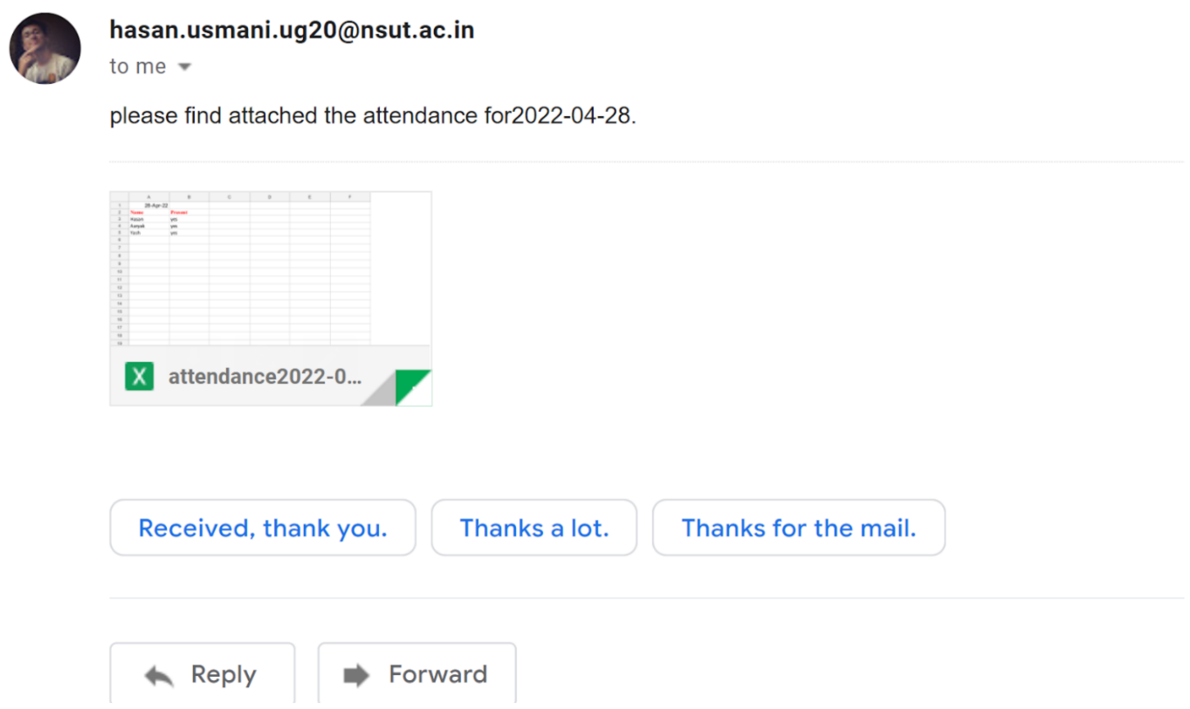
automail.py-

Module 3-

automail.py-
```
import yagmail
import os
import datetime
from datetime import datetime;
#date = datetime.date.today().strftime("%B %d, %Y")
path = 'attendance\\firebase\\attendance_files'
os.chdir(path)
files = sorted(os.listdir(os.getcwd()), key=os.path.getmtime)
newest = files[-1]
filename = newest
sub = "Attendance Report for " + str(datetime.now().date())
# mail information
yag = yagmail.SMTP("hasan.usmani.ug20@nsut.ac.in", "thus hlls vvwb pnrx")


# sent the mail
yag.send(
    to='zrusmani@gmail.com',
    subject='Attendance of ECAM_2', # email subject
    contents='please      find      attached      the      attendance
for'+str(datetime.now().date())+"." , # email body
    attachments= filename  # file attached
)
print("Email Sent!")
```



	A	B	C
1	#########		
2	**Name**	**Present**	
3	Hasan	yes	
4	Aaryak	yes	
5	Yash	yes	
6			
7			



hasan.usmani.ug20@nsut.ac.in
to me

please find attached the attendance for2022-04-28.

attendance2022-0...

Module 3-

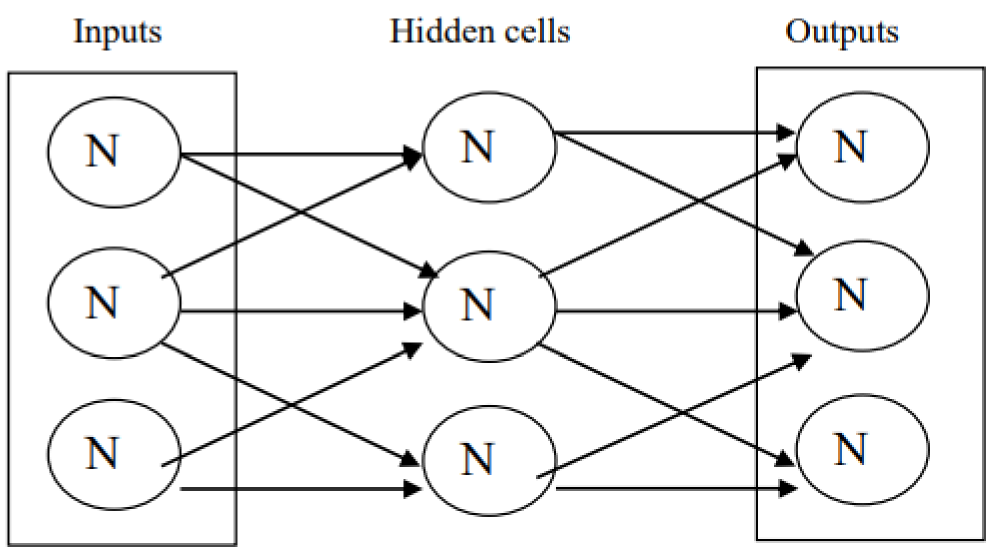
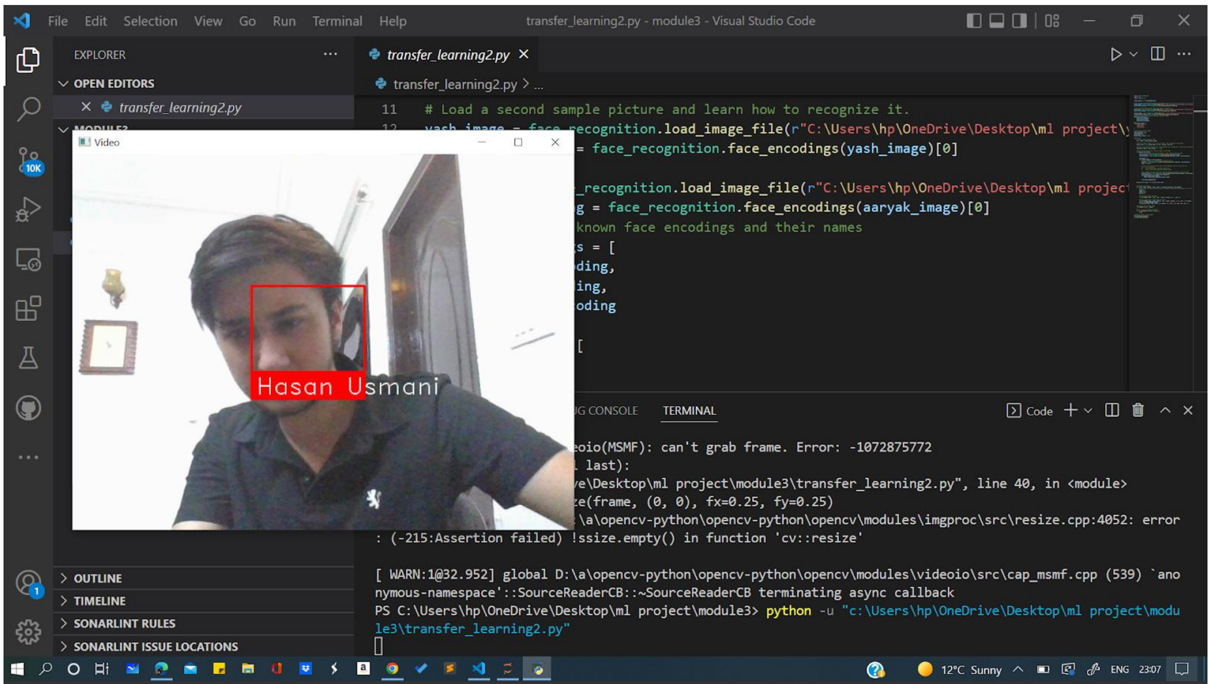


Figure 6: Artificial Neural Networks

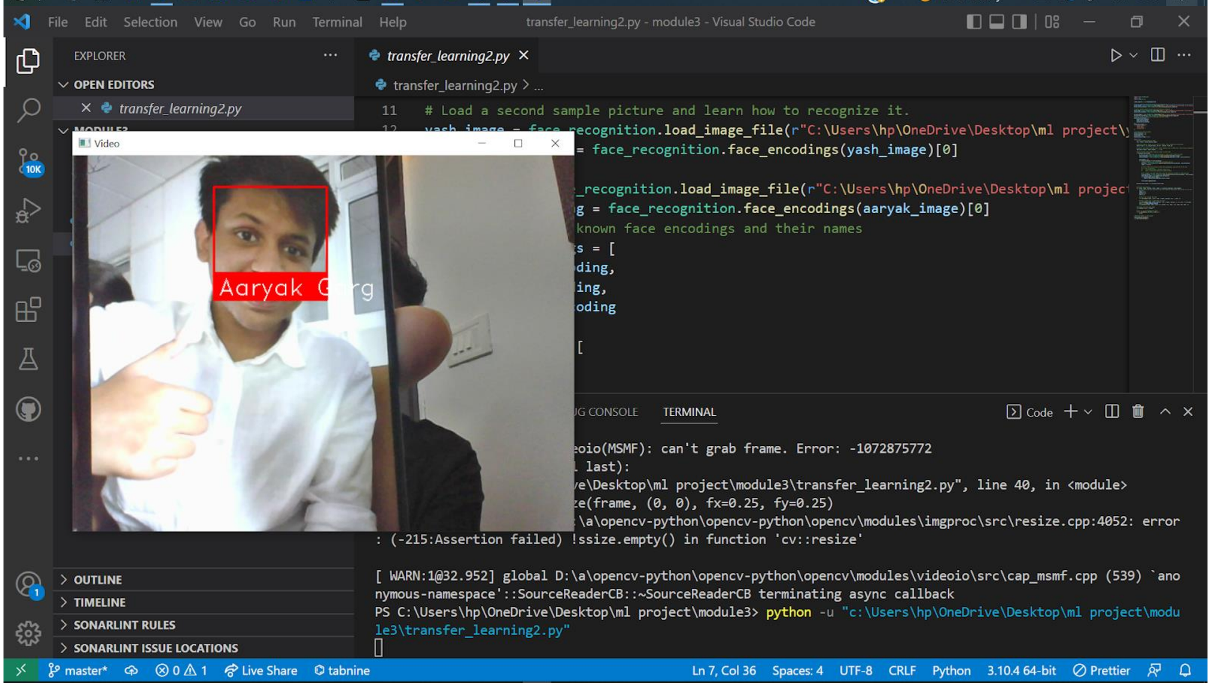
```
import face_recognition
import cv2
import numpy as np

video_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.
```

```
hasan_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\hasan.jpg")
hasan_face_encoding = face_recognition.face_encodings(hasan_image)[0]

# Load a second sample picture and learn how to recognize it.
yash_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\yash.jpg")
yash_face_encoding = face_recognition.face_encodings(yash_image)[0]

aaryak_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\aaryak.jpeg")
aaryak_face_encoding = face_recognition.face_encodings(aaryak_image)[0]
# Create arrays of known face encodings and their names
known_face_encodings = [
    hasan_face_encoding,
    yash_face_encoding,
    aaryak_face_encoding
]
known_face_names = [
    "Hasan Usmani",
    "Yash Joon",
    "Aaryak Garg "
]

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown "
            # # If a match was found in known_face_encodings, just use the first one.
            # if True in matches:
            #     first_match_index = matches.index(True)
            #     name = known_face_names[first_match_index]

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    process_this_frame = not process_this_frame

    # Display the results
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

    # Display the resulting image
    cv2.imshow('Video', frame)

    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```



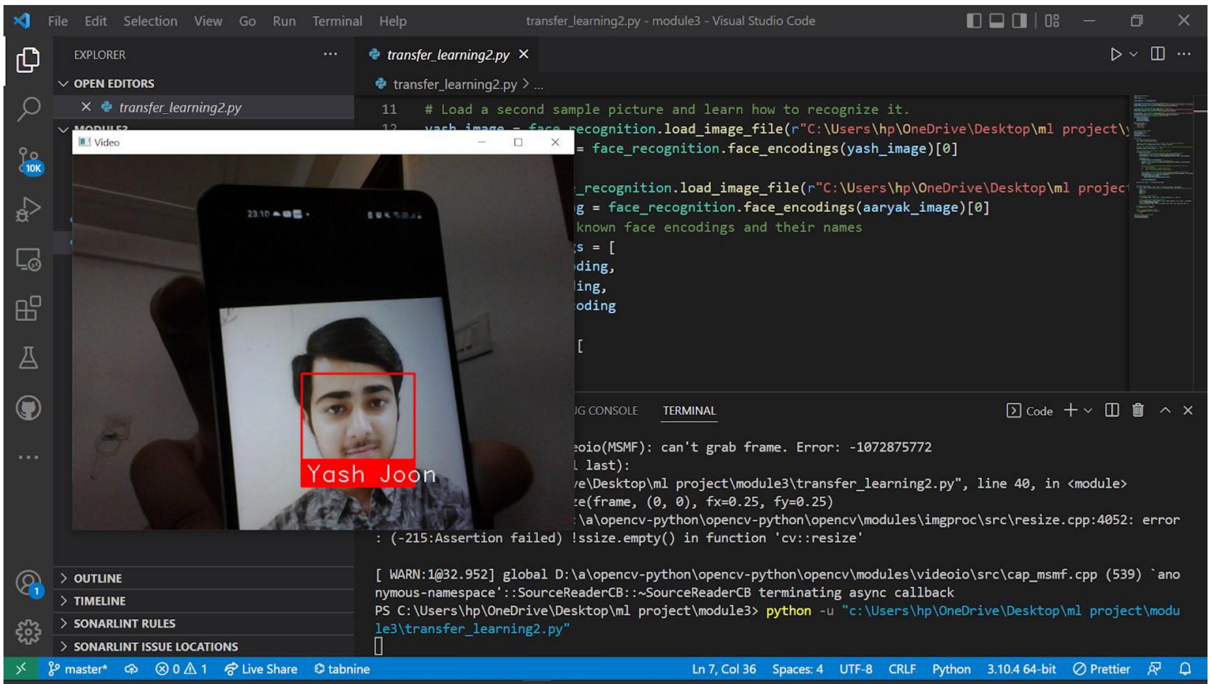
```
# Load a second sample picture and learn how to recognize it.
yash_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\y
= face_recognition.face_encodings(yash_image)[0]

_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml projec
g = face_recognition.face_encodings(aaryak_image)[0]
known face encodings and their names
s = [
ding,
ing,
oding

[
```

```
eoio(MSMF): can't grab frame. Error: -1072875772
 last):
ve\Desktop\ml project\module3\transfer_learning2.py", line 40, in <module>
ze(frame, (0, 0), fx=0.25, fy=0.25)
\a\opencv-python\opencv-python\opencv\modules\imgproc\src\resize.cpp:4052: error
: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'

[ WARN:1@32.952] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `ano
nymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\hp\OneDrive\Desktop\ml project\module3> python -u "c:\Users\hp\OneDrive\Desktop\ml project\modu
le3\transfer_learning2.py"
```


File Edit Selection View Go Run Terminal Help
transfer_learning2.py - module3 - Visual Studio Code
EXPLORER
OPEN EDITORS
transfer_learning2.py
Video
Aaryak G rg
OUTLINE
TIMELINE
SONARLINT RULES
SONARLINT ISSUE LOCATIONS
DEBUG CONSOLE
TERMINAL
master* Live Share tabnine
Ln 7, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit Prettier

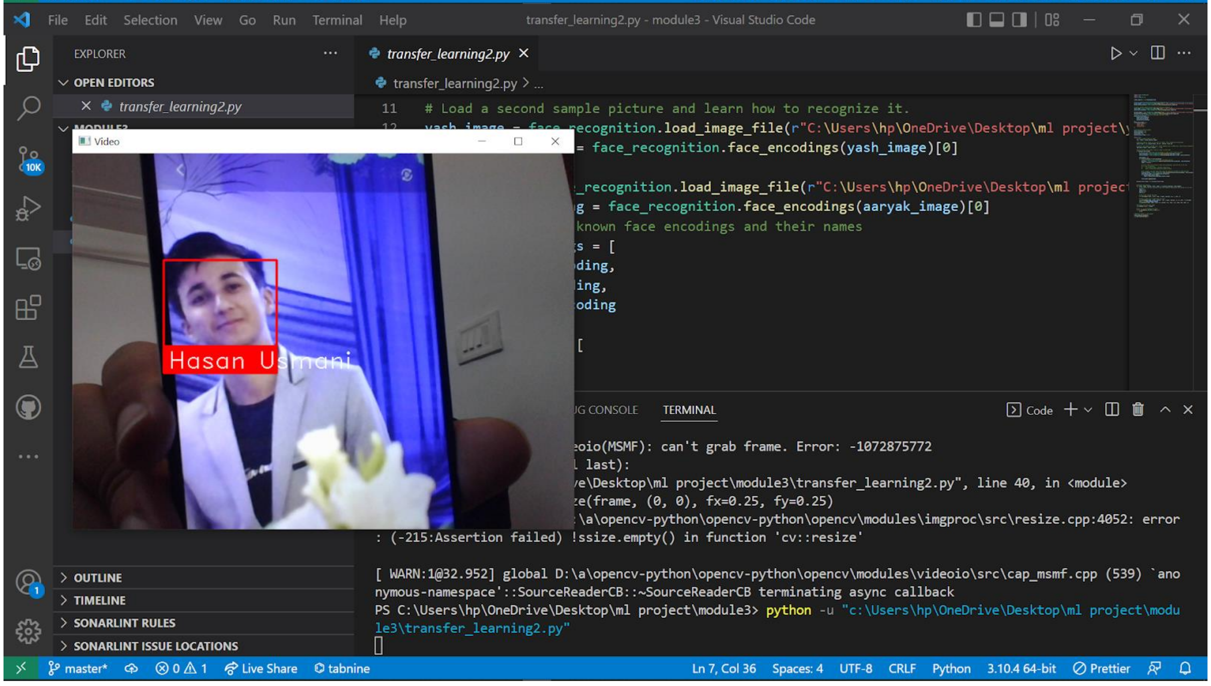
```
# Load a second sample picture and learn how to recognize it.
yash_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\y
= face_recognition.face_encodings(yash_image)[0]

_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml projec
g = face_recognition.face_encodings(aaryak_image)[0]
known face encodings and their names
s = [
ding,
ing,
oding

[
```

```
eoio(MSMF): can't grab frame. Error: -1072875772
 last):
ve\Desktop\ml project\module3\transfer_learning2.py", line 40, in <module>
ze(frame, (0, 0), fx=0.25, fy=0.25)
\a\opencv-python\opencv-python\opencv\modules\imgproc\src\resize.cpp:4052: error
: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'

[ WARN:1@32.952] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `ano
nymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\hp\OneDrive\Desktop\ml project\module3> python -u "c:\Users\hp\OneDrive\Desktop\ml project\modu
le3\transfer_learning2.py"
```



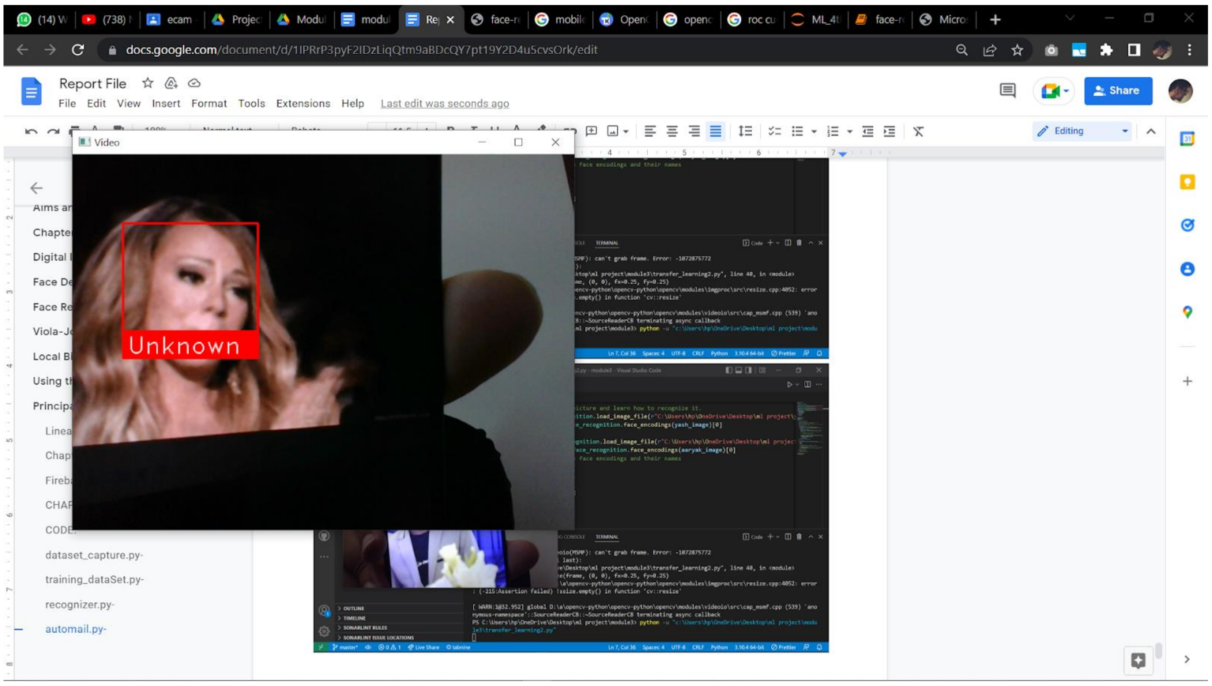
```
# Load a second sample picture and learn how to recognize it.
yash_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\y
= face_recognition.face_encodings(yash_image)[0]

_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project
g = face_recognition.face_encodings(aaryak_image)[0]
known face encodings and their names
s = [
ding,
ing,
oding

[
```

```
eoio(MSMF): can't grab frame. Error: -1072875772
l last):
/e\Desktop\ml project\module3\transfer_learning2.py", line 40, in <module>
ze(frame, (0, 0), fx=0.25, fy=0.25)
:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\resize.cpp:4052: error
: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'

[ WARN:1@32.952] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `ano
nymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\hp\OneDrive\Desktop\ml project\module3> python -u "c:\Users\hp\OneDrive\Desktop\ml project\modu
le3\transfer_learning2.py"
```



```
# Load a second sample picture and learn how to recognize it.
yash_image = face_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project\y
= face_recognition.face_encodings(yash_image)[0]

_recognition.load_image_file(r"C:\Users\hp\OneDrive\Desktop\ml project
g = face_recognition.face_encodings(aaryak_image)[0]
known face encodings and their names
s = [
ding,
ing,
oding

[
```

```
eoio(MSMF): can't grab frame. Error: -1072875772
l last):
/e\Desktop\ml project\module3\transfer_learning2.py", line 40, in <module>
ze(frame, (0, 0), fx=0.25, fy=0.25)
:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\resize.cpp:4052: error
: (-215:Assertion failed) !ssize.empty() in function 'cv::resize'

[ WARN:1@32.952] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `ano
nymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\hp\OneDrive\Desktop\ml project\module3> python -u "c:\Users\hp\OneDrive\Desktop\ml project\modu
le3\transfer_learning2.py"
```



CHAPTER 5-

CONCLUSION

The proposed method uses face detection and face recognition that helps to maintain the automated attendance system.

Module 1-

METHOD(Face Detection)	T.P	F.P	Total	Precision
Deep learning method	7	0	14	50.0%
Haar Cascade method	13	5	18	72.2%
Deep learning +Haar cascade	13	1	14	92.8%

Algorithm(Face Recognition)	Performance	Condition
PCA	Worse,Worst	Large Datset,Small Dataset
LDA	Best,Worse	Large Dataset,Small Dataset
LBP	Comparable to LDA,BEST	Large Dataset,Small Dataset

The LBPH algorithm can recognize the front face as well as side face with approximate accuracy of 90% This process is able to detect multiple faces with an accuracy of 91.67%. By using this method, we can recognize faces during day and night time and are also able to detect 15 degrees side facing faces.

In Module 2 for detection, Paul–Viola Jones algorithm is used and for face recognition Linear Binary Pattern Histogram (LPBH) algorithm is applied. In the result, the unique ID and name of the student is displayed along with the confidence percentage. Confidence percentage represents the distance between the histogram of the stored image and histogram of the real time image and is calculated by using Euclidean distance. Lower is the distance, higher is the recognition rate.

We tried to explore transfer learning and see the accuracy obtained on using pretrained models in module 3. MediaPipe in dlib library was used for face detection and the openCV Face_Recognition library was used for face recognition .It was observed that though the frame rate was considerably less but the accuracy increased immensely.

FUTURE SCOPE

To increase the accuracy further we can experiment with other classifiers such as KNN ,SVM apart from the distance classifier used in the project.Further we can also train our model to adjust as per different lightning conditions and detect side faces as well.

The future scope of the project is to integrate the software with the hardware components for example CCTV cameras and better Cam modules through which a monthly list of the defaulter students can be sent to the mentor. Additionally, an application can be developed to help students to maintain a track of their attendance which is capable of recognising the identity of each individuals and eventually record down the data into a database system.It can also be used in offices where a large group of employees sit in a hall and their attendance will be marked automatically by capturing a video .

References:

https://www.ijert.org/research/attendance-management-system-using-face-recognition-IJERTV10IS080085.pdf

https://www.researchgate.net/publication/341876647_Face_Recognition_based_Attendance_Management_System

https://www.researchgate.net/publication/220566092_Face_Recognition_A_Literature_Survey

https://ieeexplore.ieee.org/document/9134225

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9138372