BCA MINOR PROJECT

FACIAL RECOGNITION SYSTEM

FINAL YEAR

Graphical User Interface :

tKinter : It is a collection of many widgets and it is used to create a GUI, using Python.

Step 1 : At first we call a function called Tk()

Step 2 : We resize our root screen -

```
root = Tk()
root.geometry("675x500+120+120")
```

(675 = W, 500 = H, 120 = x, 120 = y)

W = width
H = Hight
x = x axis
y = y axis

Step 3 : We fix the root screen size using min size() and max size().

Step 4 : We set our root wallpaper. We downloaded a external picture from internet. After downloading the image we access it using follow module in python. It supports a range of image file formats such as PNG, JPEG, PPM, GIF, TIFF and BMP.

step5 : After we create a canvas and set the height and width of this canvas.

step 6 : We put the image as a background image of the canvas.

canvas.create_image (335, 195, image = bg image)

335 → x, 195 → y

bg image → A image file

x = x axis

y = y axis

step7 : After this step we set our GUI title, and we also set the icon of our GUI.

root.title ("FACIAL RECOGNITION SYSTEM")

root.iconbitmap ("icon.ico")

→ customized icon the format is = .ico

step 8 : After that we add all the necessary buttons.

b = Button (root, text = "button text", font = ("font family", "font size", "font style"), fg = "font colour", padx = "Horizontal Gap of button text", pady = "Vertical Gap of button text", relief = "type of the button", bg = "button background colour", command = lambda : [fn(), fn1(), fn2()....]

→ Multiple functions

Step-9 : Place all the buttons in our GUI at specific position of the canvas.

```
button_place = canvas.create_window(x axis, y axis,
                                    window = button)
```

Note:

For producing the clicking sound we used winsound module and create a lambda function -

```
clicked = lambda : PlaySound(random.choice(click list),
                             SND_FILENAME)
```

- PlaySound → Inbuilt function of winsound module
- click list → custom list of external clicking audio files names
- random.choice() → Type of the choice string which is randomly choose by the random.choice()
- SND_FILENAME

Step 10 : After create all GUI widgets we will call the main loop function. It is used to call the all pack widgets which we added in our previous steps.

GUI COMPLETE

FOLDER STRUCTURE

Total 3 Programs → Machine Learning (Superset of deep learning)

1. face_sample_collects.py
2. model_training.py
3. Recognizer.py

GUI Portion : 1 Program

1. GUI.py

DEEP LEARNING PORTION

1. Face sample collects :-

- Step 1 : At first we import the necessary modules open cv, numpy. open cv stands for open computer vision it is use to process image from webcam, live detection.
- Step-2 : Then we create the face classifier to detect our face from the web camera frame. We download the extenal extensible markup file (xml) named 'haarcascade_frontalface_default.xml'
- Step-3 : We create a function for extract our face and create our own dataset by collecting our images total 150 face sample images.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

↓

We convert the face image which is in Blue, Green, Red format to GRAY Scale.

We use cv2.videocapture() to stream our video by using webcam for saving total 150 face samples. We use os module in python, using this module at first we specified our path of single image file which is save in jpg format, then we use cv2.imwrite() to save the face sample at the image file location, after complete all the backend works then we use cv2.putText() to put the counter text in the face cropper window which is created by cv2, if no faces present in front of our webcam then face cropper will not open and print "face not found" in debugging console.

Note:

```
for (x,y,w,h) in faces:
cropped_face = img[y:y+h, x:x+w]
```

It is a logic for cropping our face using the face classifier which we have created in the previous portion in step 3.

After we done with all these things we wrap up all these things into a function named. fsc().

Then the function will call in the button of our GUI portion at the command property, with the clicked function simultaneously.

2. Model_training :

Step1 : At first we categorize our own created dataset which contains total 150 face samples.

Training_Data, Labels = [], []

↓ store 150 face samples as grey scale image [type is unsynchronized integer 8]

Every images store as numpy array using np.asarray() ← 150

Step-2 : We labeled our face samples according their filenames which is stored in the 'faces' folder at the project directory.

Step-3 : We create our model.

model = cv2.face.LBPHFaceRecognizer_create()

(*) LBPH = Local Binary Pattern Histogram.

It is a algorithm which is used to recognize the face of a person it is known for its performance and how its able to recognize the face of a person from both front and side face.

This is inbuilt in open cv2 module.

Step-4 : After creating our model we train it using train function.

```
model.train(np.asarray(Training-Data), np.asarray(Labels))
```

TRAINING COMPLETE

Recognizer :-

Step-1 : We create a function named face detector() which use to retake live faces from the frame of our webcam.

```
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
    roi = img[y:y+h, x:x+w]
    roi = cv2.resize(roi, (200,200))
```

→ This portion is used for fetching our face from live frame of our webcam.

Step-2 : After fetching our face from camera we predicted our face is authorized face or not using model.predict().

Step-3 : After predicting we create a variable result and store our prediction into this variable. Next we do

confidence = int(100 * (1-(result[1])/300))

↓

This variable assign the percentage of our model accuracy which is calculated by above simple logic. After that we can typecast this to integer.

Step-4 : If the confidence variable's value is more than 83 then we use cv2.putText() to print

'It is a authorized user'.

'UNLOCKED'

If no faces found then print 'face not found'.

If the confidence variables value's is not more than 83 then print '... 'Fake User'.

'LOCKED'

Step-5 : We can set our own title in the live detect screen. Using cv2.imshow('Face Recognition', image)

↓

image, face = face-detector (frame)

↓

Face detector function is already discussed in previous steps.

Recognizer.py

⊛ Note :

After we done all these things we wrap up it into a function named recog().

Model-training.py

⊛ Note :

After we done all these things we wrap up it into a function named mt().

After wrapping all these things these functions will call of our GUI portion at the command property with the clicked function simultaneously.

DEEP LEARNING COMPLETE

Reset :-

```
from plyer import notification
import os
def AllFilesDelete():
    dir = 'faces/'
    for f in os.listdir(dir):
        os.remove(os.path.join(dir, f))
    print("RESETING ALL IMAGES IN
     FACES FOLDER....")
notification.notify(
    title = "RESETING FINISHED",
    message = "ALL FACE SAMPLES ARE
                DELETED",
    # display time
    timeout = 2
)
```

os module is used to provide functions for interacting with Operating System.

plyer module is used to generate notifications.

This notify function is used for designing our notification which will generate when our all face samples are deleted.

This property is used to generate our notification after our given time. (here we set the time 2 seconds).

This AllFilesDelete() is called when we click the 'reset' button in our GUI.

RESET COMPLETE.

PROJECT COMPLETE