**Face Recognition Project:-**

#import module from tkinter for UI

from tkinter import \*

import os

from datetime import datetime;

import cv2

import sys

from PIL import Image

import numpy as np

import pickle

import csv

import pandas as pd

#creating instance of TK

root=Tk()

#sys.path.append('C:\python 3.7\lib\site-packages')

root.configure(background="white")

#path=r"C:\Users\HP"

#os.chdir(path)

#root.geometry("300x300")

def function1():

face\_id=input('Enter your Id')

face\_name=input('Enter Your Name')

if(\_\_name\_\_=="\_\_main\_\_"):

# Creating individual folders for individual users #

root="." # informing the computer about the present directory

folder="Python Train Images"

name=face\_name

path=f"{root}/{folder}/{name}"

os.makedirs(path)

# Folder of user Created #

root1=r"C:\Users\HP\Python Train Images"

# Creating The specified Directory for cv2.imwrite()

directory=f"{root1}\{name}"

dirc=str(directory)

# Creating Directory for .csv file

root2=r"C:\Users\HP"

# Loading Haar CascadeClassifier

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

capture=cv2.VideoCapture(0) # Opens Webcam

count =0 #Initialize sample face image

while(True):

# Capture video frame

ret, image\_frame = capture.read()

# Convert frame to grayscale

gray = cv2.cvtColor(image\_frame, cv2.COLOR\_BGR2GRAY)

# Detect frames of different sizes, list of faces rectangles

faces = face\_cascade.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)

# Increment sample face image

count += 1

roi\_gray= gray[y:y+h, x:x+w] # Region Of Interest

# Change the current working Directory for the image to be saved

os.chdir(dirc)

# Save the captured image into the training folder

cv2.imwrite(str(face\_name) + str(face\_id) + '.' + str(count) + '.jpg',roi\_gray)

# cv2.imwrite("Python Train Images/"+ str(face\_name) + str(face\_id) + '.' + str(count) + '.jpg',roi\_gray)

# cv2.imwrite("Folder Location with this /" + name + id + number + Region of interest)

# 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(20) & 0xFF == ord('q'):

break

# If image taken reach 100, stop taking video

elif count>=30:

print("Successfully Captured")

break

# Stop video

capture.release()

# Close all started windows

cv2.destroyAllWindows()

#########################

os.chdir(root2)

with open('StudentDetails.csv','a',newline='') as f: # The csv file is opened in append mode and [newline=''] is used to remove blank rows in between csv file...

df=pd.DataFrame({'ID' : face\_id,

'Name':pd.Categorical(face\_name)})

csvpath=r"C:\Users\HP\StudentDetails.csv"

if os.path.getsize(csvpath) == 0:

df.to\_csv(f, index=False)

else:

df.to\_csv(f, header=False, index=False)

def function2():

basedir=os.path.dirname(os.path.abspath('\_file\_')) # If I Put \_file\_ without invited commas then their will an errror

imagedir=os.path.join(basedir,"Python Train Images")

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

recognizer= cv2.face.LBPHFaceRecognizer\_create()

faceSamples=[]

y\_labels=[]

current\_id=0

label\_ids={}

for root,dirs,files in os.walk(imagedir):

for file in files:

if file.endswith("png") or file.endswith("jpg") or file.endswith("JPG") or file.endswith("jpeg") or file.endswith("JPEG"):

path=os.path.join(root,file)

# What is the use of Label??? Label is used to extract the folder name of the captured images...

# which will be further used in Recognizer.py file

label=os.path.basename(root).replace(" ","-").lower() # We can also write label=os.path.basename(os.path.dirname(path))

#print(label, path)

if not label in label\_ids: #

label\_ids[label]=current\_id # creating a label id for motherchod labels

current\_id += 1 #

id\_=label\_ids[label] # 'id' is a built in function in python so use 'id\_'

#print(label\_ids)

pil\_image= Image.open(path).convert("L") # Grayscale

size=(550, 550)

final\_image= pil\_image.resize(size, Image.ANTIALIAS)

#image=pil\_image.rotate(45).show() Thois function is used to rotate image by 45 degrees

#print(pil\_image)

image\_array=np.array(pil\_image,'uint8') # Turning this into numpy array "uint" stands for unsigned int

#print(image\_array)

faces=face\_cascade.detectMultiScale(image\_array,1.32,5)

for (x,y,w,h) in faces:

faceSamples.append(image\_array[y:y+h,x:x+w])

y\_labels.append(id\_)

# Using Pickle to save Label Ids

with open("labels.pickle",'wb') as f:

pickle.dump(label\_ids,f)

# Train the OpenCV Recognizer

recognizer.train(faceSamples,np.array(y\_labels))

recognizer.save("trainer.yml")

print("Successfully trained")

############################### PERFORMING FACE RECOGNITION ################################

labels={}

with open("labels.pickle",'rb') as f:

og\_labels = pickle.load(f)

labels = {v:k for k,v in og\_labels.items()}

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

recognizer= cv2.face.LBPHFaceRecognizer\_create()

recognizer.read("trainer.yml")

capture=cv2.VideoCapture(0)

font=cv2.FONT\_HERSHEY\_SIMPLEX

name=labels[id\_]

color=(255,255,255)

stroke=2

while(True):

# Capture video frame

ret, image\_frame = capture.read()

# Convert frame to grayscale

gray = cv2.cvtColor(image\_frame, cv2.COLOR\_BGR2GRAY)

# Detect frames of different sizes, list of faces rectangles

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=7)

# Loops for each faces

for (x,y,w,h) in faces:

roi\_gray = gray[y:y+h, x:x+w]

# Crop the image frame into rectangle

cv2.rectangle(image\_frame, (x,y), (x+w,y+h), (255,0,0), 2)

id\_,conf=recognizer.predict(roi\_gray)

if conf>= 45 and conf<=85:

cv2.putText(image\_frame, (labels[id\_]) , (x,y) , font , 1 , color , stroke , cv2.LINE\_AA )

#print(labels[id\_])

else:

str\_="Intruder!!!Cannot Recognize"

cv2.putText(image\_frame, str\_ , (x,y) , font , 1 , color , stroke , cv2.LINE\_AA )

cv2.imshow('frame',image\_frame)

if cv2.waitKey(100) & 0xFF == ord('q'):

break

capture.release()

cv2.destroyAllWindows()

#def function5():

#os.startfile(os.getcwd()+"/developers/diet1frame1first.html")

def function3():

root.destroy()

#stting title for the window

root.title("AUTOMATIC ATTENDANCE MANAGEMENT USING FACE RECOGNITION")

#creating a text label

Label(root, text="FACE RECOGNITION ATTENDANCE SYSTEM",font=("times new roman",20),fg="white",bg="maroon",height=2).grid(row=0,rowspan=2,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

#creating first button

Button(root,text="Take Images For Creating Dataset",font=("times new roman",20),bg="#0D47A1",fg='white',command=function1).grid(row=3,columnspan=2,sticky=W+E+N+S,padx=5,pady=5)

#creating second button

Button(root,text="Train + Recognize Dataset",font=("times new roman",20),bg="#0D47A1",fg='white',command=function2).grid(row=4,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

# creating third button

# Button(root,text="Recognize + Attendance",font=('times new roman',20),bg="#0D47A1",fg="white",command=function3).grid(row=5,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

# creating attendance button

# Button(root,text="Attendance Sheet",font=('times new roman',20),bg="#0D47A1",fg="white",command=attend).grid(row=6,columnspan=2,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,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

#creating EXIT button

Button(root,text="Exit",font=('times new roman',20),bg="maroon",fg="white",command=function3).grid(row=9,columnspan=2,sticky=N+E+W+S,padx=5,pady=5)

root.mainloop()