Emotion based smart music player

def display(emotion):

condition= True

print("inside function",emotion)

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

color=np.asarray((255, 255, 0))

color = color.astype(int)

color = color.tolist()

def draw\_text(coordinates, image\_array, text, color, x\_offset=0, y\_offset=0,

font\_scale=2, thickness=2):

x, y = coordinates[:2]

cv2.putText(image\_array, text, (x + x\_offset, y + y\_offset),

cv2.FONT\_HERSHEY\_SIMPLEX,

font\_scale, color, thickness, cv2.LINE\_AA)

cv2.putText(image\_array, emotion\_probability, (x + x\_offset, y + y\_offset),

cv2.FONT\_HERSHEY\_SIMPLEX,

font\_scale, color, thickness, cv2.LINE\_AA)"""

**capture frames from a camera**

cap = cv2.VideoCapture(0)

**loop runs if capturing has been initialized.**

while condition:

**reads frames from a camera**

ret, img = cap.read()

**convert to gray scale of each frames**

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

**Detects faces of different sizes in the input image**

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

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

**To draw a rectangle in a face**

cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)

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

roi\_color = img[y:y+h, x:x+w]

draw\_text((x,y), img, emotion,

color , 0, -45, 1, 1)

**Detects eyes of different sizes in the input image**

#eyes = eye\_cascade.detectMultiScale(roi\_gray)

#To draw a rectangle in eyes

'''for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2) ''

**Display an image in a window**

cv2.imshow('img',img)

**Wait for Esc key to stop**

#k = cv2.waitKey(2000)

if cv2.waitKey(1):

break

**Close the window**

cap.release()

# De-allocate any associated memory usage

cv2.destroyAllWindows()

condition=False

return

if \*\*name\*\* == "\*\*main\*\*":

display()

import cv2

import argparse

import time

import os

import Update\_Model

import glob

import random

import eel

import light

import winsound

frequency=2500

duration=1000

eel.init('WD')

emotions=["angry", "happy", "sad", "neutral"]

fishface = cv2.face.FisherFaceRecognizer\_create()

font = cv2.FONT\_HERSHEY\_SIMPLEX

'''try:

fishface.load("model.xml")

except:

print("No trained model found... --update will create one.")'''

parser=argparse.ArgumentParser(description="Options for emotions based music

player(Updating the model)")

parser.add\_argument("--update", help="Call for taking new images and retraining the

model.", action="store\_true")

args=parser.parse\_args()

facedict={}

video\_capture=cv2.VideoCapture(0)

facecascade=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

def crop(clahe\_image, face):

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

faceslice=clahe\_image[y:y+h, x:x+w]

faceslice=cv2.resize(faceslice, (350, 350))

facedict["face%s" %(len(facedict)+1)]=faceslice

return faceslice

def grab\_face():

#ret, frame=video\_capture.read()#1

#gray\_image = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)#

#rgb\_image = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)#

#bgr\_image = cv2.cvtColor(rgb\_image, cv2.COLOR\_RGB2BGR)#

#cv2.imshow('window\_frame', bgr\_image)#

ret, frame=light.nolight()

cv2.imwrite('test.jpg', frame)

cv2.imwrite("images/main%s.jpg" %count, frame)

gray=cv2.imread('test.jpg',0)

#gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)#3

clahe=cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

clahe\_image=clahe.apply(gray)

return clahe\_image

def detect\_face():

clahe\_image=grab\_face()

face=facecascade.detectMultiScale(clahe\_image, scaleFactor=1.1, minNeighbors=15,

minSize=(10, 10), flags=cv2.CASCADE\_SCALE\_IMAGE)

if len(face)>=1:

faceslice=crop(clahe\_image, face)

#return faceslice

else:

print("No/Multiple faces detected!!, passing over the frame")

def save\_face(emotion):

print("\n\nLook "+emotion+" untill the timer expires and keep the same emotion for

some time.")

winsound.Beep(frequency, duration)

print('\a')

for i in range(0, 5):

print(5-i)

time.sleep(2)

while len(facedict.keys())<16:

detect\_face()

for i in facedict.keys():

path, dirs, files = next(os.walk("dataset/%s" %emotion))

file\_count = len(files)+1

cv2.imwrite("dataset/%s/%s.jpg" %(emotion, (file\_count)), facedict[i])

facedict.clear()

def update\_model(emotions):

print("Update mode for model is ready")

checkForFolders(emotions)

for i in range(0, len(emotions)):

save\_face(emotions[i])

print("Collected the images, looking nice! Now updating the model...")

Update\_Model.update(emotions)

print("Model train successful!!")

def checkForFolders(emotions):

if os.path.exists("dataset/%s" %emotion):

pass

else:

os.makedirs("dataset/%s" %emotion)

def identify\_emotions():

prediction=[]

confidence=[]

for i in facedict.keys():

pred, conf=fishface.predict(facedict[i])

cv2.imwrite("images/%s.jpg" %i, facedict[i])

prediction.append(pred)

confidence.append(conf)

output=emotions[max(set(prediction), key=prediction.count)

print("You seem to be %s" % output)

print("confidence %s" %conf)

facedict.clear()

return output;

#songlist=[]

#songlist=sorted(glob.glob("songs/%s/\*" %output))

#random.shuffle(songlist)

#os.startfile(songlist[0])

count=0

@eel.expose

def getEmotion():

count=0

while True:

count=count+1

detect\_face()

if args.update:

update\_model(emotions)

break

elif count==10:

fishface.read("model2.xml")

return identify\_emotions()

break

eel.start('main.html')