THE   
EMOTIONAL   
PROGRAM [TEP]

An Emotion Detecting program that plays songs according to your Facial Expressions.

Created By:

CID: 103782

M.Hanif Hasan 9003

Sehar Dedar Ali 9008

Rafia Masood 9213

Process Of Developing Program:

The program was Developed on TensorFlow backend engine!!

The program has to .py files one is,

1. Emotion\_Recognition.py
2. videoTester.py

in file no-1 “Emotion\_Recognition.py” the model was trained using CNN more then 5 times with given variables,

**1st attempt to train:**

With model activation=softmax and with 3 convolution layers

Accuracy:61%

**5th attempt to train:**

With model activation=sigmod and with 3 convolution layers

Accuracy:87%

Finally started using “training with 3rd layer.h5” trained model in file no-2 “videoTester.py” to get live video/images from webcam and then converting them grayscale using “OpenCV” and after detecting the facial expression using “pygame” plays and stops songs

**PROGRAM GUIDE:**

**Run “**videoTester.py” it will require you to make a facial expression watching your webcam and then press “K” or just “Enter” and it will play the song according to your emotion/expression. To stop the song press “S” or “.” To stop the song.

The Emotional Program [TEP] only detects these Emotions/Facial Expressions,

1. Neutral
2. Happy
3. Sad
4. Angry
5. Fear

**CODE:**

File Name=”Emotion\_Recognition.py”:

import sys, os

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Conv2D, MaxPooling2D, BatchNormalization,AveragePooling2D

from keras.losses import categorical\_crossentropy

from keras.optimizers import Adam

from keras.regularizers import l2

from keras.utils import np\_utils

# pd.set\_option('display.max\_rows', 500)

# pd.set\_option('display.max\_columns', 500)

# pd.set\_option('display.width', 1000)

df=pd.read\_csv('fer2013.csv')

# print(df.info())

# print(df["Usage"].value\_counts())

# print(df.head())

X\_train,train\_y,X\_test,test\_y=[],[],[],[]

for index, row in df.iterrows():

val=row['pixels'].split(" ")

try:

if 'Training' in row['Usage']:

X\_train.append(np.array(val,'float32'))

train\_y.append(row['emotion'])

elif 'PublicTest' in row['Usage']:

X\_test.append(np.array(val,'float32'))

test\_y.append(row['emotion'])

except:

print(f"error occured at index :{index} and row:{row}")

num\_features = 64

num\_labels = 7

batch\_size = 64

epochs = 55

width, height = 48, 48

X\_train = np.array(X\_train,'float32')

train\_y = np.array(train\_y,'float32')

X\_test = np.array(X\_test,'float32')

test\_y = np.array(test\_y,'float32')

train\_y=np\_utils.to\_categorical(train\_y, num\_classes=num\_labels)

test\_y=np\_utils.to\_categorical(test\_y, num\_classes=num\_labels)

#cannot produce

#normalizing data between oand 1

X\_train -= np.mean(X\_train, axis=0)

X\_train /= np.std(X\_train, axis=0)

X\_test -= np.mean(X\_test, axis=0)

X\_test /= np.std(X\_test, axis=0)

X\_train = X\_train.reshape(X\_train.shape[0], 48, 48, 1)

X\_test = X\_test.reshape(X\_test.shape[0], 48, 48, 1)

# print(f"shape:{X\_train.shape}")

##designing the cnn

#1st convolution layer

model = Sequential()

model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu', input\_shape=(X\_train.shape[1:])))

model.add(Conv2D(64,kernel\_size= (3, 3), activation='relu'))

# model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2, 2)))

model.add(Dropout(0.5))

#2nd convolution layer

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

# model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2, 2)))

model.add(Dropout(0.5))

#3rd convolution layer

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2, 2)))

model.add(Flatten())

#fully connected neural networks

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(num\_labels, activation='sigmoid'))

# model.summary()

#Compliling the model

model.compile(loss=categorical\_crossentropy,

optimizer=Adam(),

metrics=['accuracy'])

#Training the model

model.fit(X\_train, train\_y,

batch\_size=batch\_size,

epochs=epochs,

verbose=1,

validation\_data=(X\_test, test\_y),

shuffle=True)

#Saving the model to use it later on

fer\_json = model.to\_json()

with open("fer.json", "w") as json\_file:

json\_file.write(fer\_json)

model.save\_weights("training with 3rd layer.h5")

**CODE:**

File Name=”videoTester.py”:

import os

import cv2

import numpy as np

import threading

import winsound

import pygame

#from getkey import getkey, keys

from keras.models import model\_from\_json

from keras.preprocessing import image

winsound.PlaySound("filename", winsound.SND\_ASYNC | winsound.SND\_ALIAS )

print("\n\t\t Welcome to THE EMOTIONAL PROGRAM [TEP]\n")

predicted\_emotion = []

def Key\_Grabber():

pygame.mixer.init()

while(True):

key = input("Press Enter to play song according to your expression \nOnce songs are playing Press S or . to stop the songs\n")

if((key =='s' or key == 'S') or (key =='.')):

key='s'

else:

key='k'

if(key == 'K' or key == 'k'):

if(predicted\_emotion == 'happy'):

print("Happy Song")

pygame.mixer.music.load("Bruno Mars - 24K Magic (Happy Music).mp3")

pygame.mixer.music.play()

if(predicted\_emotion == 'sad'):

print("Sad Song")

pygame.mixer.music.load("Serhat Durmus - Yalan (Sad Music).mp3")

pygame.mixer.music.play()

if(predicted\_emotion == 'angry'):

print("Angry Song")

pygame.mixer.music.load("DOOM (2016) - BFG Division (Angry Music).mp3")

pygame.mixer.music.play()

if(predicted\_emotion == 'fear'):

print("Fear Music")

pygame.mixer.music.load("Scary horror music (Fear Music) .mp3")

pygame.mixer.music.play()

if(predicted\_emotion == 'neutral'):

print("Neutral Song")

pygame.mixer.music.load("My Love Is Winter (Natural Music).mp3")

pygame.mixer.music.play()

if(key == 's' or key == 'S'):

print("Music Stopped!")

pygame.mixer.music.stop()

thread1 = threading.Thread(target = Key\_Grabber, args = ())

thread1.start()

#load model

model = model\_from\_json(open("fer.json", "r").read())

#load weights

model.load\_weights('training with 3rd layer.h5')

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

cap=cv2.VideoCapture(0)

while True:

ret,test\_img=cap.read()# captures frame and returns boolean value and captured image

if not ret:

continue

gray\_img= cv2.cvtColor(test\_img, cv2.COLOR\_BGR2GRAY)

faces\_detected = face\_haar\_cascade.detectMultiScale(gray\_img, 1.32, 5)

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

cv2.rectangle(test\_img,(x,y),(x+w,y+h),(255,0,0),thickness=7)

roi\_gray=gray\_img[y:y+w,x:x+h]#cropping region of interest i.e. face area from image

roi\_gray=cv2.resize(roi\_gray,(48,48))

img\_pixels = image.img\_to\_array(roi\_gray)

img\_pixels = np.expand\_dims(img\_pixels, axis = 0)

img\_pixels /= 255

predictions = model.predict(img\_pixels)

#find max indexed array

max\_index = np.argmax(predictions[0])

emotions = ('angry', '', 'fear', 'happy', 'sad', '', 'neutral')

predicted\_emotion = emotions[max\_index]

#print(predicted\_emotion)

cv2.putText(test\_img, predicted\_emotion, (int(x), int(y)), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

resized\_img = cv2.resize(test\_img, (1000, 700))

cv2.imshow('Facial emotion analysis ',resized\_img)

if cv2.waitKey(10) == ord('q'):#wait until 'q' key is pressed

break

cap.release()

cv2.destroyAllWindows