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="Emotion_Recognition.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
```