Emojify – Create your own emoji with Deep Learning

Emojis or avatars are ways in which to point nonverbal cues. These cues became an important a part of on-line chatting, product review, complete emotion, and lots of more. It conjointly cause increasing knowledge science analysis dedicated to emoji-driven storytelling. With advancements in pc vision and deep learning, it's currently doable to discover human emotions from images. during this deep learning project, we'll classify human facial expressions to filter and map corresponding emojis or avatars.

About Data Set The FER2013 (Facial Expression Recognition) data set consists of 48 \* 48 pixels grayscale facial images. The image is centered and occupies the same amount of space. This dataset consists of facial emotions from the following categories: 0: angry 1: disgusted 2: achievement 3: happy 4: sad 5: surprised 6: natural

import tkinter as tk

from tkinter import \*

import cv2

from PIL import Image, ImageTk

import os

import numpy as np

import cv2

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D

from keras.optimizers import Adam

from keras.layers import MaxPooling2D

from keras.preprocessing.image import ImageDataGenerator

emotion\_model = Sequential()

emotion\_model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48,48,1)))

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

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Flatten())

emotion\_model.add(Dense(1024, activation='relu'))

emotion\_model.add(Dropout(0.5))

emotion\_model.add(Dense(7, activation='softmax'))

emotion\_model.load\_weights('model.h5')

cv2.ocl.setUseOpenCL(False)

emotion\_dict = {0: " Angry ", 1: "Disgusted", 2: " Fearful ", 3: " Happy ", 4: " Neutral ", 5: " Sad ", 6: "Surprised"}

emoji\_dist={0:"./emojis/angry.png",2:"./emojis/disgusted.png",2:"./emojis/fearful.png",3:"./emojis/happy.png",4:"./emojis/neutral.png",5:"./emojis/sad.png",6:"./emojis/surpriced.png"}

global last\_frame1

last\_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)

global cap1

show\_text=[0]

def show\_vid():

cap1 = cv2.VideoCapture(0)

if not cap1.isOpened():

print("cant open the camera1")

flag1, frame1 = cap1.read()

frame1 = cv2.resize(frame1,(600,500))

bounding\_box = cv2.CascadeClassifier('/home/shivam/.local/lib/python3.6/site-packages/cv2/data/haarcascade\_frontalface\_default.xml')

gray\_frame = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

num\_faces = bounding\_box.detectMultiScale(gray\_frame,scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in num\_faces:

cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

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

cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray\_frame, (48, 48)), -1), 0)

prediction = emotion\_model.predict(cropped\_img)

maxindex = int(np.argmax(prediction))

cv2.putText(frame1, emotion\_dict[maxindex], (x+20, y-60), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE\_AA)

show\_text[0]=maxindex

if flag1 is None:

print ("Major error!")

elif flag1:

global last\_frame1

last\_frame1 = frame1.copy()

pic = cv2.cvtColor(last\_frame1, cv2.COLOR\_BGR2RGB)

img = Image.fromarray(pic)

imgtk = ImageTk.PhotoImage(image=img)

lmain.imgtk = imgtk

lmain.configure(image=imgtk)

lmain.after(10, show\_vid)

if cv2.waitKey(1) & 0xFF == ord('q'):

exit()

def show\_vid2():

frame2=cv2.imread(emoji\_dist[show\_text[0]])

pic2=cv2.cvtColor(frame2,cv2.COLOR\_BGR2RGB)

img2=Image.fromarray(frame2)

imgtk2=ImageTk.PhotoImage(image=img2)

lmain2.imgtk2=imgtk2

lmain3.configure(text=emotion\_dict[show\_text[0]],font=('arial',45,'bold'))

lmain2.configure(image=imgtk2)

lmain2.after(10, show\_vid2)

if \_\_name\_\_ == '\_\_main\_\_':

root=tk.Tk()

img = ImageTk.PhotoImage(Image.open("logo.png"))

heading = Label(root,image=img,bg='black')

heading.pack()

heading2=Label(root,text="Photo to Emoji",pady=20, font=('arial',45,'bold'),bg='black',fg='#CDCDCD')

heading2.pack()

lmain = tk.Label(master=root,padx=50,bd=10)

lmain2 = tk.Label(master=root,bd=10)

lmain3=tk.Label(master=root,bd=10,fg="#CDCDCD",bg='black')

lmain.pack(side=LEFT)

lmain.place(x=50,y=250)

lmain3.pack()

lmain3.place(x=960,y=250)

lmain2.pack(side=RIGHT)

lmain2.place(x=900,y=350)

root.title("Photo To Emoji")

root.geometry("1400x900+100+10")

root['bg']='black'

exitbutton = Button(root, text='Quit',fg="red",command=root.destroy,font=('arial',25,'bold')).pack(side = BOTTOM)

show\_vid()

show\_vid2()

root.mainloop()

SUMMARY

In this beginner’s deep learning project, we built a convolutional neural network to recognize facial emotions. We trained our model on the FER2013 dataset. We therefore put these emotions in correspondence with the corresponding emojis or avatars.

Using OpenCV’s haar cascade xml, we get the bounding box of the faces in the webcam. Then, we provide these boxes to the trained model for classification.