Avatars For Your Expressions

05.07.2021

Group members :

Labdhi Sharad Gandhi (190110041)

Shirshika (190040111)

Swastika Agarwal (190110095)

Overview

Deep learning can build remarkably accurate emojis and avatars. Emojis and avatars are examples of nonverbal signals. These indications have been ingrained in internet talking, product reviews, brand sentiment, and a variety of other activities. It also resulted in an increase in data science research on emoji-driven storytelling.

Many approaches for detecting human faces in photos and videos have been proposed, and they may be classified into four categories: feature-based methods, knowledge-based methods, appearance-based methods, and template-based methods. When these approaches are employed in isolation, they are unable to handle all of the difficulties associated with face identification, such as posture, emotion, and orientation.

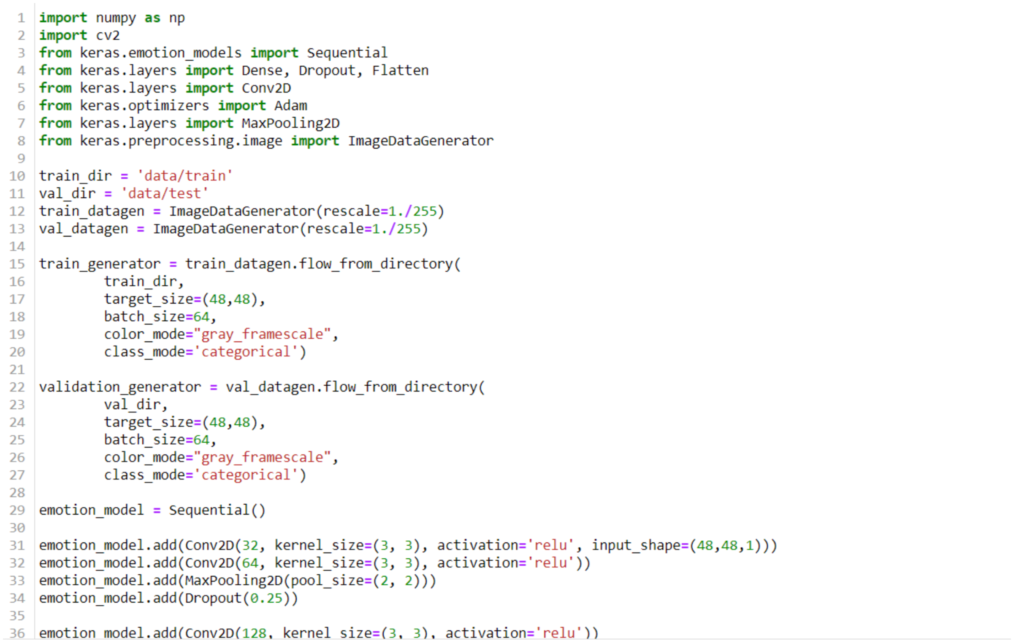
As a result, it's best to use many techniques in succession or simultaneously. The majority of currently available facial expression recognition algorithms are focused on recognising five basic emotion categories: pleasure, sorrow, fear, rage, and disgust. For face detection, we use the HAAR classifier and the CNN algorithm and for expression detection two functions are used: relu and soft max(these are activity functions).

Code :

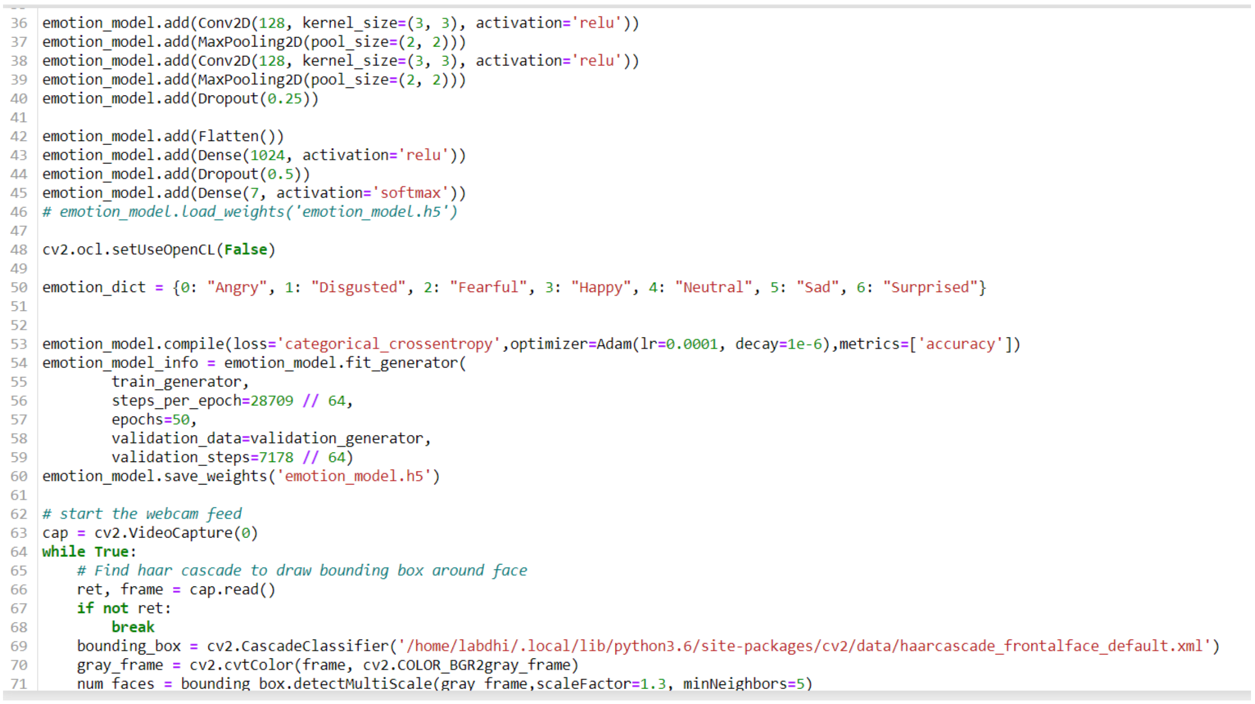
```
import tkinter as tk
from tkinter import *
import cv2
from PIL import Image, ImageTk
import os
import numpy as np

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

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)
```

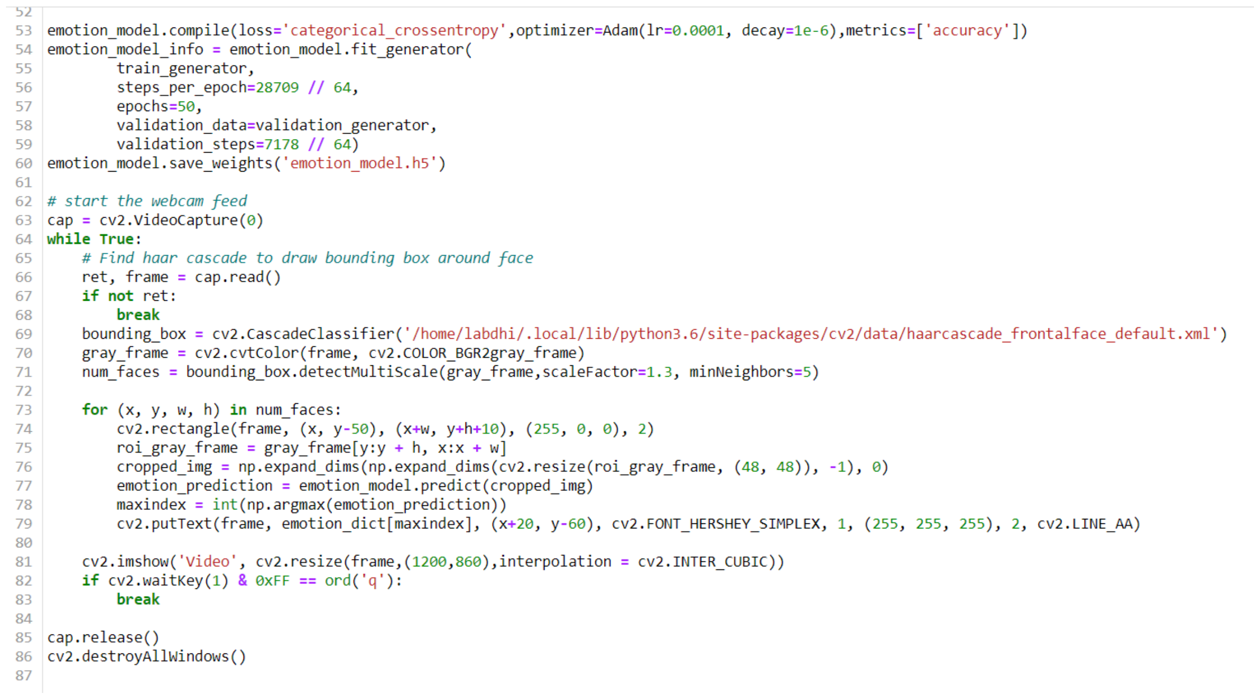
```
if __name__ == '__main__':
    root=tk.Tk()
    img = ImageTk.PhotoImage(Image.open("logo.png"))
    heading = Label(root,image=img,bg='black')

    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()
```



```python
import numpy as np
import cv2
from keras.emotion_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

train_dir = 'data/train'
val_dir = 'data/test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="gray_framescale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="gray_framescale",
        class_mode='categorical')

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'))
```



```python
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('emotion_model.h5')

cv2.ocl.setUseOpenCL(False)

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


emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-6),metrics=['accuracy'])
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
emotion_model.save_weights('emotion_model.h5')

# start the webcam feed
cap = cv2.VideoCapture(0)
while True:
    # Find haar cascade to draw bounding box around face
    ret, frame = cap.read()
    if not ret:
        break
    bounding_box = cv2.CascadeClassifier('/home/labdhi/.local/lib/python3.6/site-packages/cv2/data/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2gray_frame)
    num_faces = bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3, minNeighbors=5)
```



```python
emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-6),metrics=['accuracy'])
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
emotion_model.save_weights('emotion_model.h5')

# start the webcam feed
cap = cv2.VideoCapture(0)
while True:
    # Find haar cascade to draw bounding box around face
    ret, frame = cap.read()
    if not ret:
        break
    bounding_box = cv2.CascadeClassifier('/home/labdhi/.local/lib/python3.6/site-packages/cv2/data/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2gray_frame)
    num_faces = bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3, minNeighbors=5)

    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame, (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)
        emotion_prediction = emotion_model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.imshow('Video', cv2.resize(frame,(1200,860),interpolation = cv2.INTER_CUBIC))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Goals

The goal of this project is to analyze and map human face emotions to emojis. To identify facial emotions, we'll use a convolutional neural network. Then we'll match those feelings to the appropriate emojis or avatars.

Motivation

Snapchat and TikTok, for example, offer some really unique features like lenses and emojis (bitmoji), and Apple has recently released its animated characters Animoji and Memoji.

The animated form of popular emoji characters is known as real-time emoji. Your facial expressions are effectively mirrored by them. As a result, as you move your face and speak, they will respond in real time.

Facial emotion identification and analysis has received a lot of interest in the development of human-machine interfaces because it provides a natural and efficient way for humans to interact. Person identification and access control, video call and teleconferencing, forensic applications, human-computer interaction, automated surveillance, cosmetology, and other applications are all connected to the face and its expressions.

However, the performance of facial expression detection has a significant impact on the performance of all applications.

Interested in creating a real-time emoji.

Details

Facial emoji or avatar recognizer is a user-facing programme that recognises the expression of the person in the video being recorded by the camera. On the screen, a smiley that corresponds to the person's expression in the video is displayed, which varies as the emotions change. Using OpenCV's haar cascade xml we are getting the bounding box of the faces in the webcam. For face detection, we utilise the HAAR classifier, the CNN algorithm for expression detection, and two functions: relu and soft max (these are activity functions)

Methodology

In simple terms, facial expressions are the arrangement of facial muscles to transmit a certain emotional state to the viewer. Anger, disgust, fear, happiness, sadness, surprise, and neutral are the six major types of emotions. Train a model to distinguish between them by utilising the FER2013 dataset to train a convolutional neural network and fine-tuning the model with various hyper-parameters.

Decomposing an image

Pixels are the primary building blocks of images, yet they are nothing more than numbers. Colored pictures are frequently thought to be split into three colour channels: red, green, and blue, with each channel represented by a grid (2-dimensional array). The intensity of each cell in the grid is represented by a value between 0 and 255 in each cell.

Importing necessary libraries

- Keras
- numpy
- tkinter
- opencv

- pillow

Establish a Data Loading Mechanism

Now we'll define the load_data() method, which will quickly read the data file and extract the information we need before converting it to an image format.

The pictures in our collection are all 48x48 pixels in size. There is just one channel in these photos because they are grayscale. The picture data will be extracted and rearranged into a 48x48 array. Then, to normalise the data, convert it to unsigned integers and divide by 255. Because 255 is the highest number a single cell may have, we divide each element by 255 to ensure that all of our values fall between 0 and 1. We'll look at the Usage column and divide the data into two lists, one for training and one for testing the network.

Defining the model

We'll construct a Sequential Convolutional Network with Keras. As a result, our neural network will consist of a linear stack of layers.

Convolutional Layers: These are the network's building pieces, and they compute the dot product of their weights and the tiny areas to which they are attached. This is the technique through which layers learn specific characteristics from these pictures.

Activation functions are functions that are applied to all levels of the network's outputs. In this project, we'll be relying on two functions: Relu and Soft max.

Layers of Pooling: These layers sample the process along the dimensions. This reduces the amount of spatial data and the amount of processing power required.

Dense layers: At the end of a CNN These layers take in all of the feature data created by the convolution layers and make the decisions.

Dropout Layers: prevents overfitting by turning off a few neurons in the network at random.

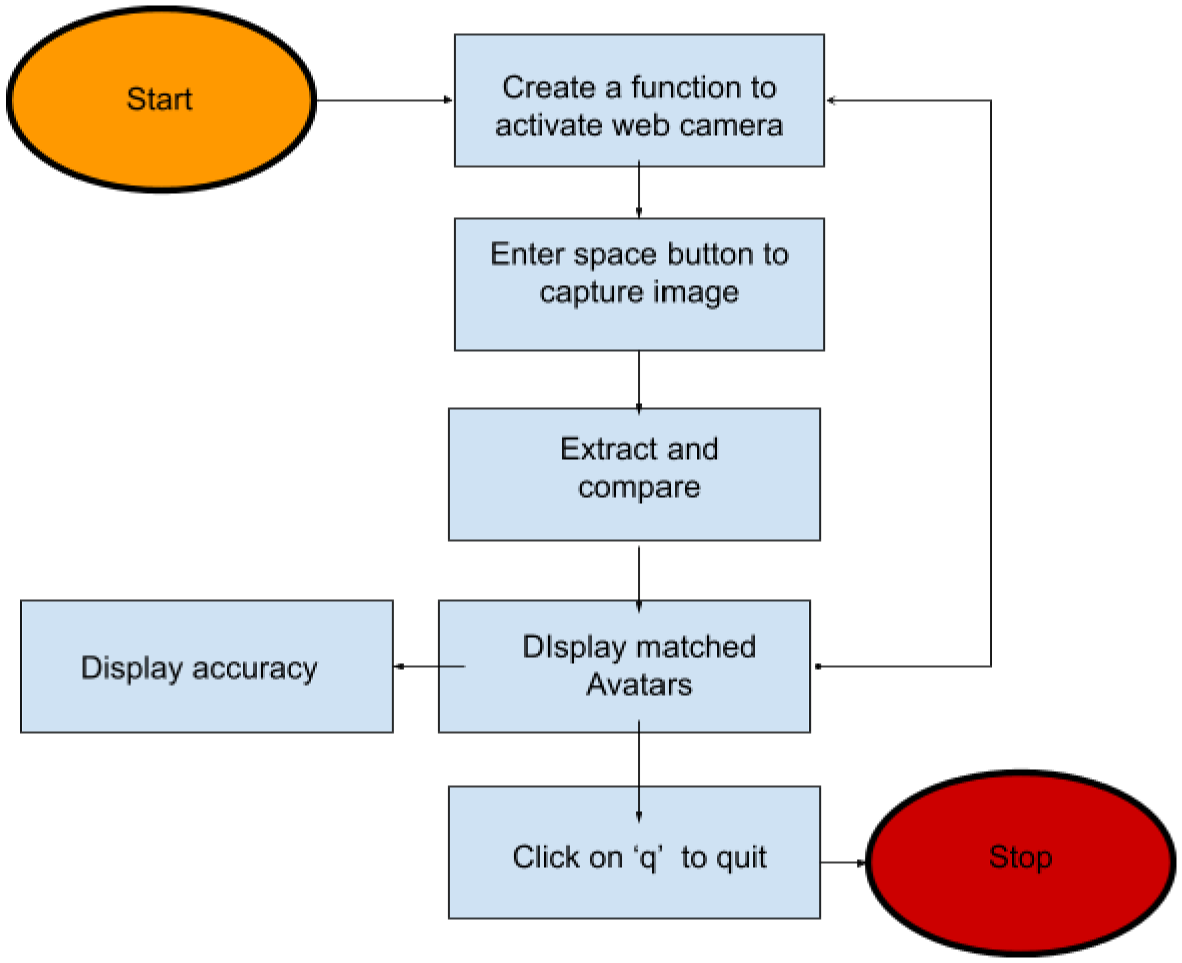
Train and Test the model

The project begins with the definition of a loading mechanism and the loading of pictures. Following that, a training set and a testing set are produced. Following that, a good model and a few callback functions are defined. The basic components of a convolutional neural network are examined, and then the network is trained.

Output

When an emotion is detected, the appropriate emoticon appears on the left side of the screen. This emoji or avatar changes in response to the person in front of the webcam's emotion. As a result, the expression of the individual in front of the webcam changes. As a result, this real-time application is extremely useful in subjects such as psychology, computer science, linguistics, neurology, and other related disciplines.

Flowchart



Conclusion

A human emotion detector utilising emoticons is proposed, which uses machine learning and Python to predict people's emotions and express them using emoticons. These steps involve picture capture, image preprocessing, face identification, feature extraction, and classification, after which the system gives the user specific music based on his emotion. Our technology is designed to focus on live webcam videos.

The primary aim of this project is to create an automatic face expression detection system that uses an emoticon to provide output for individuals, assigning them various therapies or remedies to relieve stress. Happiness, Sadness, Surprise, Fear, Disgust, and Anger were among the widely acknowledged emotions utilised in the tests.