Emojify – Create your own emoji with Deep Learning

Payas Doshi[1], Priyanshi Sethi[2], Pulkit Sharma[3], Mahaveer Jain[4]

Abstract:- Emojis are small images that are commonly included in social media text messages. The combination of visual and textual content in the same message builds up a modern way of communication. Emojis or avatars are ways to indicate nonverbal cues. These cues have become an essential part of online chatting, product review, brand emotion, and many more. It also led to increasing data science research dedicated to emoji-driven storytelling. With advancements in computer vision and deep learning, it is now possible to detect human emotions from images. In this deep learning project, we will classify human facial expressions to filter and map corresponding emojis or avatars. This project is not intended to solve a real-world problem, instead it allows us to see things more colourful in the chatting world. Emoji-Fy is a software which deals with the creation of Emoji's or Avatars.

In Today's Generation people usually tend to communicate with each other using Emoticons. So, we thought of making our own customized emojis.

Key-words:- Convolutional Neural Network(CNN), Deep Learning, Fer2013 Dataset.

Introduction:-

People use emojis every day? Emojis have become a new language that can more effectively express an idea or emotion. This visual language is now a standard for online communication, available not only in Twitter, but also in another large online platform such as Facebook and Instagram. In Today's Generation people usually tend to communicate with each other using Emoticons. So, we thought of making our own customized emojis. Emoji-Fy is a software which deals with the creation of Emoji's or Avatars.

The neural network has been an emerging application in numerous and diverse areas as example of end to end learning This paper is based on a system which implements Convolutional Neural Network and Fer2013 Dataset to detect emotions from facial expressions and converting them to personalised emojis.

We are building a convolution neural network to recognize facial emotions. We will be training our model on the FER2013 dataset. Then we'll map those emotions with the corresponding emojis or avatars. Fer2013 contains approximately 30,000 facial RGB images of different expressions with size restricted to 48×48, and the main labels of it can be divided into 7 types: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral. The Disgust expression has the minimal number of images – 600, while other labels have nearly 5,000 samples each.

Methodology:-

Proposed System

A. Facial Emotion Recognition Using CNN

1. Build a CNN architecture.

Here we are importing all the required libraries needed for our model and then we are initialising the training and validation generators i.e., we are first rescaling all our images needed to train our model and then converting them to grayscale images.

- Imports:

```
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
```

- Initializing the training and validation generators

```
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')
```

- Build the CNN architecture

```
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'))
```

2. Train the model on Fer2013 dataset.

Here we are training our network on all the images we have i.e., FER2013 dataset and then saving the weights in model for the future predictions. Then using OpenCV harassed xml to detect the bounding boxes of face in webcam and predict the emotions.

- Training the model

```
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)
```

- Predicting emotions

```
cv2.ocl.setUseOpenCL(False)

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

cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    bounding_box = cv2.CascadeClassifier('/home/shivam/.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'):
cap.release()
cv2.destroyAllWindows()
    break
```

B. Developing GUI and Mapping with emojis

Create a folder named emojis and save the emojis corresponding to each of the seven emotions in the dataset. The trained model is tested on a set of images. Random images are introduced to the network and the output label is compared to the original known label of the image. Parameters used for evaluation are F1 score, precision and recall. Precision is the proportion of predicted positives that are truly positives.

- Testing

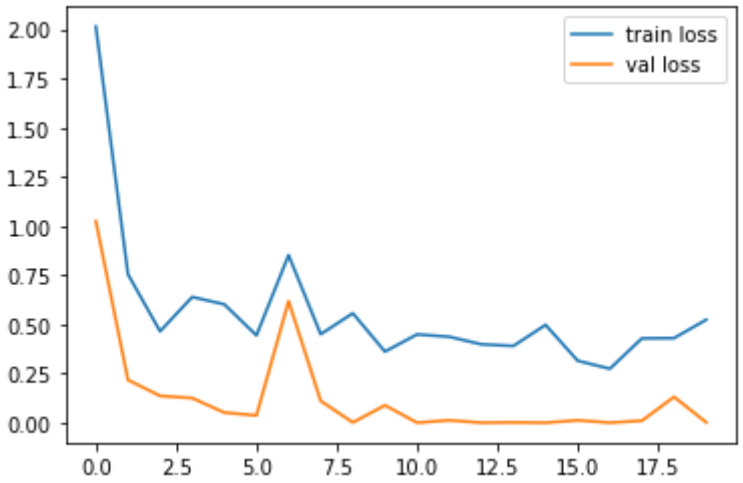


Fig 6: Loss Plot

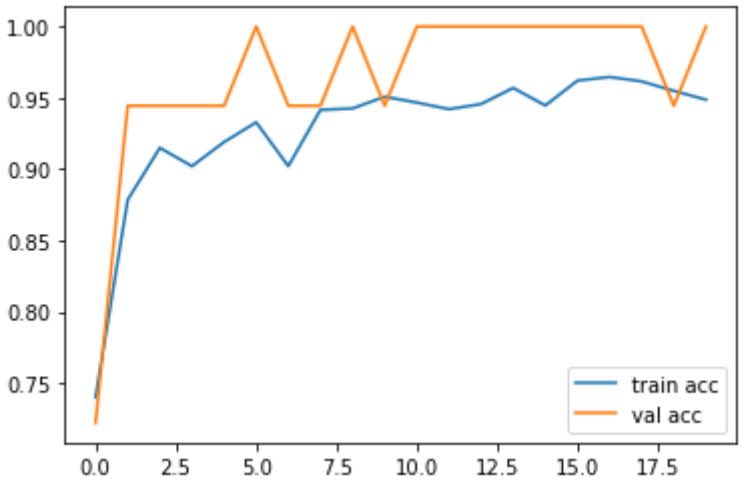
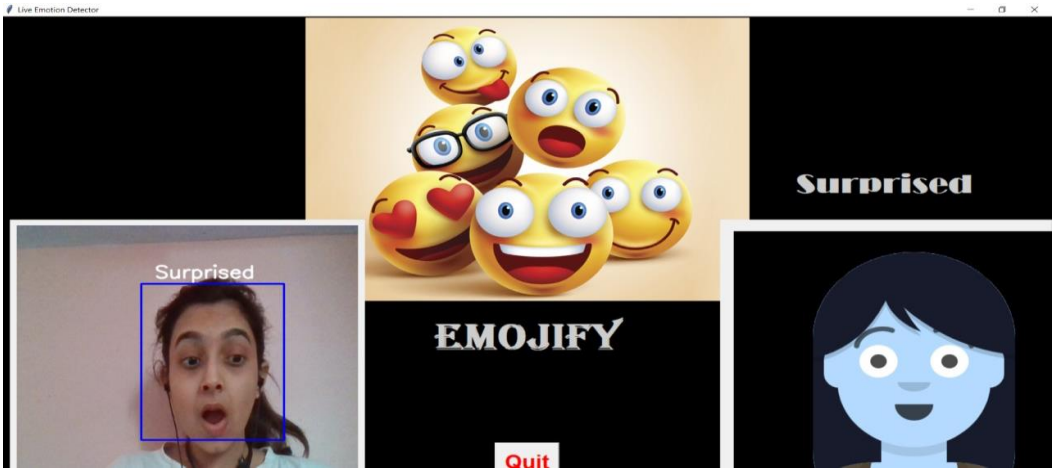


Fig 7: Accuracy Plot

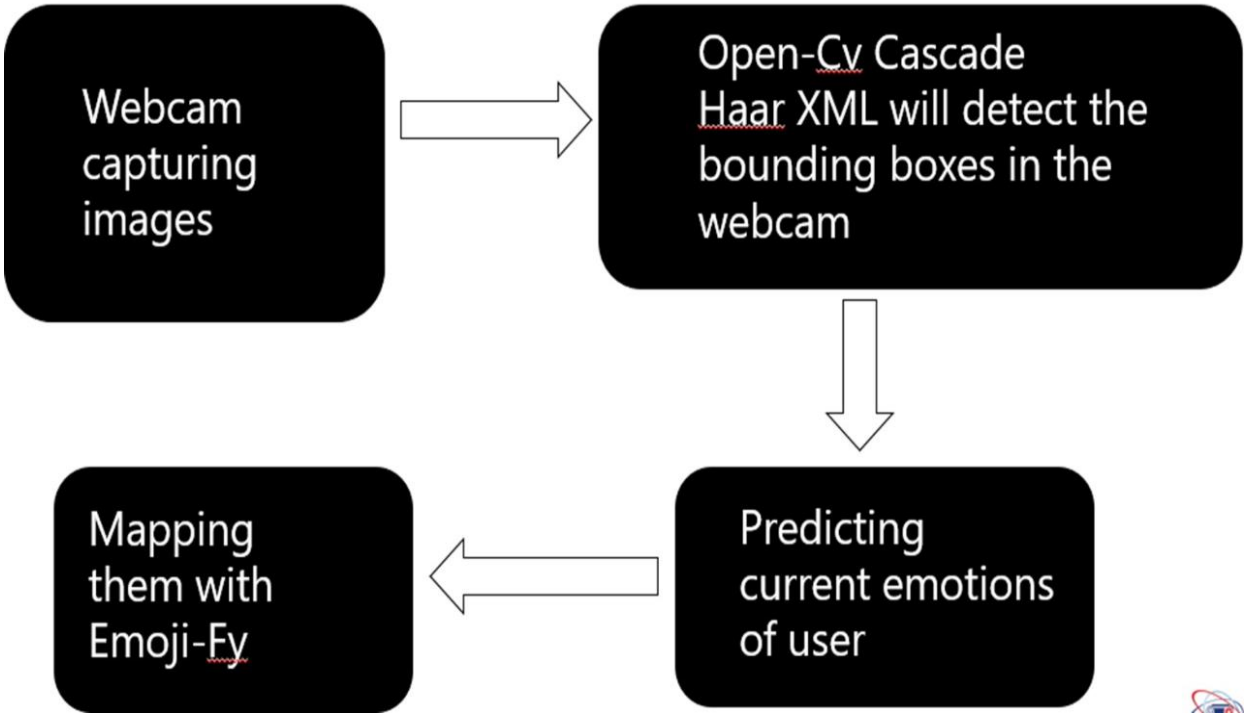
- Final Output

Here are some images of how will this project look. This dataset consists of facial emotions of following categories: 0:angry 1:disgust 2:feat 3:happy 4:sad 5:surprise 6:natural.



Project Description

Block Diagram



Tools Used:

- Google Collab supports both GPU and TPU instances, which makes it a perfect tool to train the model on large datasets
- We have also used various data science related libraries like koras, TensorFlow, OpenCV, matplotlib, NumPy etc. For the purpose of building the keras model we have used sequential modelling technique.
- VS Code is used for overall development as a standard platform.

Conclusions:-

As Today's generation people is loving the trend of communicating with non-verbal cues like emoticons so we thought why not bring out our own emojis.

With advancements in computer vision and deep learning, we will now able to detect human emotions from images. In this deep learning project, we will classify human facial expressions to filter and map corresponding emojis or avatars.

The result we are expected is the use of emojify in chatting world. We want people to communicate with their own customisable emoticon. The project will recognize one's current emotion and convert that emotion's emoji so that the customer gets emoji of their face and use it in chatting.

References:-

I. R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018.

II. D. Meena and R. Sharan, "An approach to face detection and recognition", 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp. 1-6, 2016

III. The Extended Cohn–Kanade Database. Available online: **http://www.consortium.ri.cmu.edu/ckagree/**

IV. The Japanese Female Facial Expression Database. Available online: **http://www.kasrl.org/jaffe.html**.

V. Binghamton University 3D Facial Expression Database. Available online: **http://www.cs.binghamton.edu/~lijun/Research/3DFE/3DFE_Analysis.html**.

VI. Facial Expression Recognition 2013 Database. Available online: **https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data**

VII. Zafeiriou, S.; Zhang, C.; Zhang, Z. A survey on face detection in the wild: Past, present and future. Comput. Vis. Image Underst. **2015**, 138, 1–24.

VIII. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. arXiv **2017**, arXiv:1712.04621

IX. ACM Digital Library. Available online: **https://dl.acm.org/**

X. IEEE Xplore Digital Library. Available online: **https://ieeexplore.ieee.org/Xplore/home.jsp**

XI. Bielefeld Academic Search Engine. Available online: **https://www.base-search.net/**