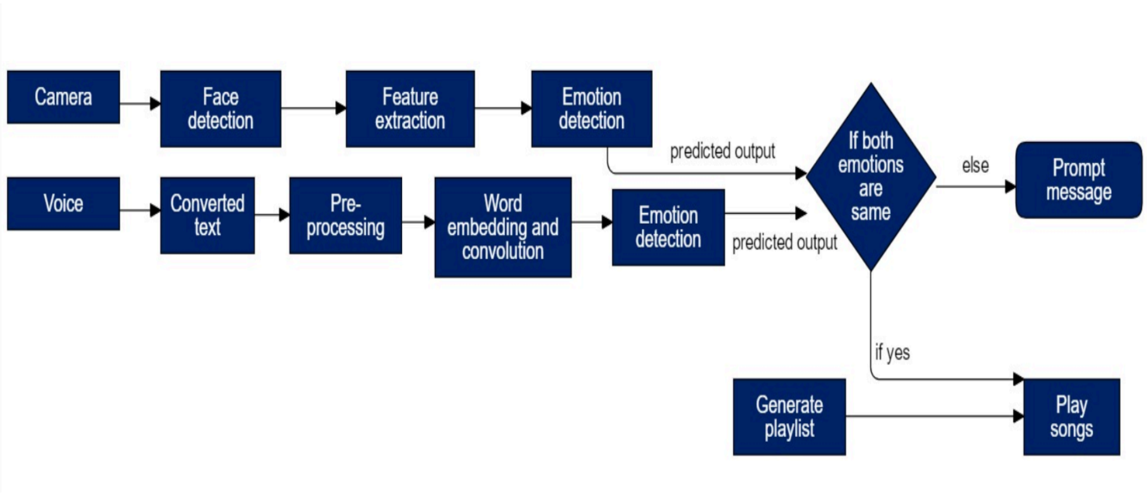
CS6301 – Machine Learning Laboratory
PROJECT REPORT

EMOTION BASED MUSIC PLAYER



Department of Computer Science, Anna University

Name: Juanita J
E-mail: juanitaraja0@gmail.com

CONTENTS

ABSTRACT

Music plays a very important role in human's daily life and in modern advanced technologies. Usually, the user has to face the task of manually browsing through the playlist of songs to select.If these playlists are generated according to one’s emotion at that moment then that would fascinate the music lovers even more.Speech and facial expressions are the most ancient and natural way of expressing feelings, emotions and mood. Here we are proposing an efficient and accurate model that would generate a playlist based on current emotional state and behaviour of the user.Though there exist a lot of algorithms for emotion detection, the computation is not as expected.

The proposed system constitutes the implementation of Convolutional Neural Networks (CNN) and Natural Language Processing (NLP) for the emotion detection using facial expressions and speech respectively and thereby playing a song accordingly. In order to obtain maximum accuracy, we are fusing the emotion detection from facial expressions and speech.This proposed system to detect emotion based on fusing facial expressions and speech as well as extracting audio features from songs to classify into a specific emotion that will generate a playlist automatically such that the computation cost is relatively low.

1. INTRODUCTION

Music has always been the expression of human emotions and the most popular choice to depict and understand human emotions. People undergo a lot of problems and the relief to stress of daily life is encountered in majority through music in today's world. In the old time system [2], user had to browse through songs and add the desired songs to the music list so he can listen anytime later.The most common means of access to personalized music today is by creating playlist of desired songs on music platforms is by creating playlist manually under categories like genre of music (hip-hop, jazz, classical, pop, rock, waltz) moods and basic emotions,climate based songs,movie,actors or the music artists etc [7].

Finding the right song for one's current mood is troublesome, especially, when a person is in a bad mood and has to search through the entire playlist to find that one song that calms him down. Also, generating a playlist and updating it constantly is no piece of cake. It requires time, patience and a lot of effort.Thus we propose a new advancement in this old time system that will be very efficient and desired, playing music by predicting the user's emotions.

The foremost concept of this project is to automatically play songs based on the emotions of the user. Emotion based music player is a novel approach that helps the user to automatically play songs according to the emotions of the user. It recognizes the facial emotions and the speech emotion [6] of the user and plays the songs according to their emotion.

In the existing system[2] the user has to manually select the songs, randomly played songs may not match to the mood of the user, the user has to classify the songs into multiple emotions and then for playing the songs user has to manually select a particular emotion. And so,the vital part of hearing the song has to be in a facilitated way, that is the player is able to play the song in accordance to the person's mood. The paper proposes such a player and hence named Emotion based music player.

Reliable emotion based classification systems can go a long way in helping us parse their meaning. However, research [5] in the field of emotion-based music classification has not yielded optimal results. In this paper, we present an effective cross-platform music player [9] which shows effective implementation of a system that recommends music based on the real-time mood of the user.

And we choose the two prominent ways to detect a human emotion in our system:Facial detection and voice detection.[13]

Facial expressions [4] are a great indicator of the state of a mind for a person. Indeed, the most natural way to express emotions is through facial expressions. Humans tend to link the music they listen to; to the emotion they are feeling. Facial emotion recognition[1] takes part as a key role in human interaction and communication processes because the expression contains a lot of information and the state of mind of a person. Facial expressions and other gestures convey nonverbal communication cues that play an important role in interpersonal relations. Facial expression recognition system technology uses algorithms to instantaneously detect faces, code facial expressions, and recognize emotional states. It does this by analyzing faces in images or video through computer powered cameras embedded in laptops, mobile phones, and digital signage systems, or cameras that are mounted onto computer screens using ML,which is explained below in this document.

As human beings, speech is also amongst the most natural ways to express ourselves. We depend so much on it that we recognize its importance when resorting to other communication forms like emails and text messages where we often use emojis to express the emotions associated with the messages.Speech Emotion Recognition [10] can extract the emotional state of the speaker from his or her speech signals. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication.

Thus,the song playlists are, at times, too large to sort out automatically. It can be a great relief and desirable if the music player can automatically sort out the music based on the current state of emotion the person is feeling. The project sets out to use various techniques for an emotion recognition system.The implementation and architecture is explained below in the document.

2. LITERATURE SURVEY

There are a considerable number of papers and applications that render papers and services for music playlist generation through emotion detection.Here we have cited a couple of papers and run comparison checks on the papers and mentioned the innovation and enhancement we have employed in our project.

1. Ninad Mehendale "Facial emotion recognition using convolutional neural networks (FERC)",Springer Nature Switzerland AG,2020.

   In this paper, the authors remove the background of the face image and to that image EV (expressional vector) is applied which is a basic perceptron unit. Through this EV the facial emotions from images with various orientations can be extracted. These are the basic pre-processing units. For final predictions CNN is used.

   Our Enhancement of a kind on this paper in comparison

   Though this paper has various pre-processing units for better accuracy, this model is used for only 5 classes of emotion whereas ours can be used to predict 7 classes of emotions. Moreover, to improve the accuracy of the model, we fuse facial emotions and voice emotions which enhances the system in a much better way. In addition to this, we extend the system by playing songs according to the predicted emotion.

2. S. Deebika, K. A. Indira, Dr. Jesline "A Machine Learning Based Music Player by Detecting Emotions", 2019, Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), IEEE.

   In this paper, the authors use Point detection algorithm to extract the features i.e. ROI. These extracted features are fed to CNN for emotion prediction . Finally, the song according to the emotion is played from the generated playlist.

   Our Enhancement on this paper in comparison

   Our system also works on a similar way, but we use haarcascade model for face detection and mini_XCEPTION (CNN model) for emotion detection rather than Point detection and SVM methodologies. In addition to this, we detect the emotions from speech also and the song is played only when both the emotions match i.e. only when the emotions are pretty much stable.

3. Xusheng Wang , Xing Chen, Congjun Cao “Human emotion recognition by optimally fusing facial expression and speech feature”, Signal Processing: Image Communication Volume 84, May 2020, 115831, Elsevier.

   In this paper, the authors fuse the emotions detected from facial expressions and speech. For detecting from face they combine CNN and RNN and for detecting from speech they convert the speech signals into images and then use weighted decision fusion method to fuse facial expression and speech signal to achieve speech emotion recognition.

   Our Enhancement

   Though we use CNN for emotion detection from face and weighted decision fusion method for fusing the emotions, we predict the emotion from speech by converting it into speech and hence use RNN and NLP techniques to predict the emotions rather than converting into images, such as the Mel Frequency Cepstrum Coefficient (MFCC). Further we extend our system for music playing with the predicted fused emotions.

4. Saeed Turabzadeh, Hongying Meng , Rafiq M. Swash, Matus Pleva and Jozef Juhar “Facial Expression Emotion Detection for Real-Time Embedded Systems”, Technologies 2018, 6, 17; doi:10.3390/technologies6010017.

   In this paper,the authors have designed and tested the emotion detection model on a MATLAB environment followed by a MATLAB Simulink environment that is capable of recognizing continuous facial expressions in real-time with a rate of 1 frame per second and that is implemented on a desktop PC. Secondly, in order to implement in real-time at a faster frame rate, the facial expression recognition system was built on the field-programmable gate array (FPGA).

   Our Enhancement

   We use tensorflow for emotion detection from face rather than MATLAB and FPGA. Tensorflow is easier to set up compared to MATLAB. Hence, our model would be more compatible than the model discussed.

5. Maruthi Raja S K, Kumaran V, Keerthi Vasan A and Kavitha N, “Real Time Intelligent Emotional Music Player using Android”, Journal for Research | Volume 03| Issue 01 | March 2017.

In this paper, the authors extract facial points using Beizer Curves which are more suitable for use in mobile devices along with music player editing features like audio trimming and precise voice recording. Later the customized audio file can be set as alarm, ringtone, notification tones.

Our Enhancement

We detect the face using haarcascasde model and hence the emotion from the detected ROI using mini_XCEPTION model which is a CNN model. We make our system more efficient for PC than mobile devices. Hence, for PC our model would be more efficient than the discussed one.

6. Eduard FRANT, Ioan ISPAS , Voichita DRAGOMIR , Monica DASCALU, Elteto ZOLTAN and Ioan Cristian STOICA, “ Voice Based Emotion Recognition with Convolutional Neural Networks for Companion Robots”, ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY Volume 20, Number 3, 2017.

   In this paper, the authors detect the voice emotion from the voice signals to detect the Pitch Average, Pitch Range, Intensity ,Voice Quality, Pitch Changes and Articulation as different emotions have different ranges of these timbres.

   Our Enhancement

   We detect the voice emotion by converting the voice into text and further detect the emotion from the converted text using CNN model. In the model discussed, they detect the emotion from speech signals using CNN model. The accuracy obtained in the discussed model is 71.33% whereas the accuracy of our model is 75.66%. Hence, our model performs better than the discussed model.

7. Karan Mistry, Prince Pathak and Prof. Suvarna Aranjo, “ Mood based Music Player”,International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 3 | Mar -2017.

   In this paper, the author detect the face using the Viola Jones algorithm and the required features are extracted using the JavaCv library that is imported in the Android Studio java folder.

   Our Enhancement

We detect the face using haarcascasde model and hence the emotion from the detected ROI using mini_XCEPTION model which is a CNN model.

8. Sathya Bursic , Giuseppe Boccignone , Alfio Ferrara , Alessandro D'Amelio and Raffaella Lanzarotti,“ Improving the Accuracy of Automatic Facial Expression Recognition in Speaking Subjects with Deep Learning” ,Appl. Sci. 2020, 10, 4002; Doi: 10.3390/app10114002.

   In this paper, the authors detect the facial emotions on facial features only, and afterwards both on facial features and articulation related cues extracted from a model trained for lip reading, while varying the number of consecutive frames provided in input as well. By this, the accuracy improved by 12% than the previous one.

   Our Enhancement

   In order to improve the accuracy we also fuse face and speech emotions but rather we convert the voice to text and detect emotion from that than from the lip which is used in the discussed model.

9. Qingmei Yao,“ Multi-Sensory Emotion Recognition with Speech and Facial Expression” ,The University of Southern Mississippi The Aquila Digital Community.

   In this paper, the authors detect the emotion from face using the Distinctive and dimensional emotion model (Hamann, 2012) and from speech using the various timbres in the speech signals. Finally, they fuse the 2 emotions using a weighted sum approach.

   Our Enhancement

   In our approach, we detect the face and voice emotions using CNN models (mini_XCEPTION and word embedding respectively) and then finally fuse the 2 using weighted sum.

10. Vladimir Chernykh and Pavel Prikhodko, “Emotion Recognition From Speech With Recurrent Neural Networks”, arXiv:1701.08071v2 [cs.CL] 5 Jul 2018.

    In this paper, the authors detect the emotion from the speech from various acoustic features (as different emotions have different acoustics). The effectiveness of such an approach is shown in two ways. Firstly, the comparison with recent advances in this field is carried out. Secondly, human performance on the same task is measured. Both criteria show the high quality of the proposed method.

Our Enhancement

We develop a system which converts the voice to text and detect the emotion from the converted text using word embedding and hence convoluding the embedded layer.

11. Renuka R. Londhe, Dr. Vrushshen P. Pawar, ―Analysis of Facial Expression and Recognition Based On Statistical Approach, International Journal of Soft Computing and Engineering (IJSCE) Volume-2, May 2012.

    In this paper,an accurate and efficient statistical based approach for analyzing extracted facial expression features was proposed by Renuka R. Londhe et al. The paper was majorly focused on the study of the changes in curvatures on the face and intensities of corresponding pixels of images. Artificial Neural Networks (ANN) was used in the classification extracted features into 6 major universal emotions like anger, disgust, fear, happy, sad, and surprise. A Scaled Conjugate Gradient back-propagation algorithm in correlation with two-layered feed forward neural network was used and was successful in obtaining a 92.2 % recognition rate.

    Our Enhancement

    In order to reduce the human effort and time needed for manual creation of playlist or segregation of songs from a playlist, in correlation with different classes of emotions and moods, various approaches, we use this system of detecting emotions from facial expressions but not as done in above paper using ANN,rather with Neural networks.

12. Z. Zeng ―A survey of affect recognition methods: Audio, visual, and spontaneous expressions,IEEE. Transaction Pattern Analysis, vol 31, January 2009

    In this paper,for the purpose of face feature recognition, facial features have been categorized into two major categories such as Appearance-based feature extraction and Geometric based feature extraction.Geometric based feature extraction technique considered only the shape or major prominent points of some important facial features such as mouth and eyes.

Our Enhancement

We have employed a predefined Mini-Xception model in our project to detect emotions of the face which might have been an advancement of the above paper to detect face features using appearance and Geometry based detections.

13. "Emotion based music player app" paper by Sri Charan Nimmagadda,Report-2017.

    This project focuses on an android application that would capture an image of the user and detect 4 emotions and develops a simple algorithm for generating a playlist and detecting emotion further also lets user add a song and skip a song.

    Our Enhancement

    In out project,we simply propose and create a model system that can play songs based on the user's emotions (6 emotions unlike in the above paper that detects 4 emotions onlu) and it can be further advanced to an app or adding options such as add song or skip songs.

14. Patent-Facial emotion recognition Paper by Mutasem K. Alsmadi,IJER,February 2015.

    This paper provides an automatic geometric method for analyzing and recognizing human facial expression based on extracted features using a neural network and then using genetic and back propagation algorithm for detecting face emotion.

    Our Enhancement

    In our paper we don't employ genetic or back propagation algorithm of any sort, we have used CNN model Mini xception for facial emotion detection.

15. Smart player research paper by Hafeez Kabani, Sharik Khan , Omar Khan, Shabana Tadvi,University of Washington,2012

This paper uses the intensity of speech of the user in order to detect emotion of the user and further uses artificial neural networks for generation of the playlist.

Our Enhancement

In our paper,we use both facial emotion and voice/speech emotion integrated, for playing the suitable playlist, and we don't use artificial networks for generating music.We employ the CNN pipeline.

3. PROBLEM DESCRIPTION

In old-style music players, a user had to manually browse through playlists and select songs that would soothe his mood and play it manually. The next advancement led to pre-created playlists by the system and the users can browse through them and add them to their music list to play them manually later when needed.The pre created playlists are classified based on several attributes like the genre of music as in hip-hop,jazz,classical,pop,rock etc,, moods and emotions,climate based songs, artists,movies,actors etc.
In today's world, with ever increasing advancements in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed, local playback, streaming playback with multicast streams and including volume modulation, genre classification etc, designing a system where the user can have a playlist of songs played, just by reading the emotions in his face and voice would be highly efficient.
While the old time music playing may satisfy the user's basic requirements,the user has to face the task of manually browsing through the playlist of songs and select songs based on the current mood and behavior. This new system would save the user time and effort of manual work which would be very desirable in the fast running world today.

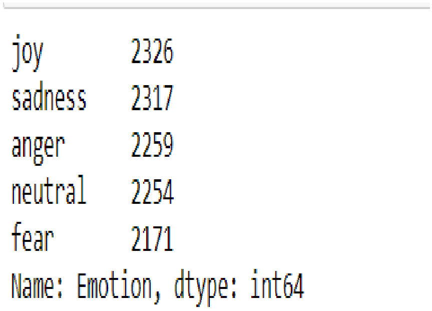
4. PROBLEM SOLUTION

4.1 DATASET

- ★ The dataset used for emotion detection from face is fer2013 which is available in kaggle.
  url-https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data?select=fer2013.tar.gz
- ★ The dataset used for emotion detection from text are:

- ➔ The dataset is collected from various sources having 7936 records as train data and 3394 records as the test data with 5 emotions being anger,joy,neutral,sad and fear.

Emotion value count:

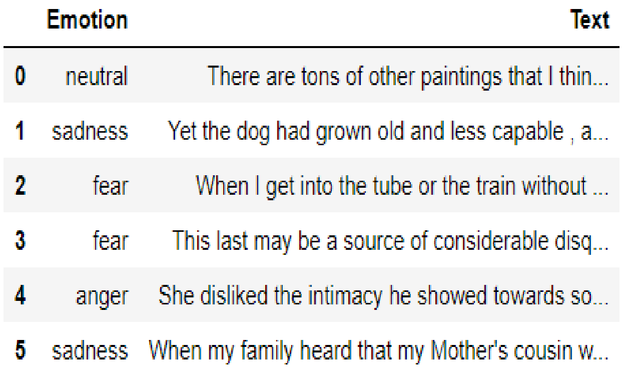


```
joy        2326
sadness    2317
anger      2259
neutral    2254
fear       2171
Name: Emotion, dtype: int64
```

- ➔ The dataset used for embedding matrix is wiki-news-300d-1m.vec which is a dataset on 1 million word vectors trained on wikipedia 2017.

url- https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip

Glimpse of data (head):



	Emotion	Text
0	neutral	There are tons of other paintings that I thin...
1	sadness	Yet the dog had grown old and less capable , a...
2	fear	When I get into the tube or the train without ...
3	fear	This last may be a source of considerable disq...
4	anger	She disliked the intimacy he showed towards so...
5	sadness	When my family heard that my Mother's cousin w...

4.2 INPUT

The inputs for the emotion detection from face and voice are, as the name suggests, human face and voice for which the emotion has to be predicted.

The face input is given through the device camera and the voice input is given to the device microphone which is converted to text using Google API.

4.3 APPROACH

- ★ The human face for which the emotion has to be detected is projected through the device camera and the face is converted into a grayscale image. From this grayscale image the

ROI (region of interest) is extracted in the form of a bounding box and the emotion for the extracted ROI is predicted using the mini_XCEPTION model.

★ The voice for which the emotion has to be detected is given to the device microphone which is converted into text Google API. This text is tokenized and hence padded. This padded text is passed as the input to the RNN model from which the emotion is predicted.

★ The 2 predicted emotions are checked and the system is directed to music playing only when both the emotions match, else the user needs to provide the inputs from first as the emotion wasn't stable.

5. NOVELTY

Every person as a music lover would love to hear songs according to their current mood rather than a playlist already in their hearing device (for example- mobile phone, computer, walkman etc). Hence we come out with an idea of playing music with the person's current mood. To do so, we detect the emotion of the person from their facial expressions and voice and match the two to play the music. If both the emotions match then music (from the person's playlist) according to the person's predicted emotion is played. In case both the emotions do not match then the system stops and urges the person to start from the first as this depicts that the emotion of the person is not stable.

Though there are previous works on emotion based music players, there is no such work on a music player which takes the emotion from the fusion of facial expressions and voice.

Hence, we have proposed this new system to detect the emotion through the fusion of emotions and play music accordingly.

6. ARCHITECTURE:

6.1 ABSTRACT ARCHITECTURE

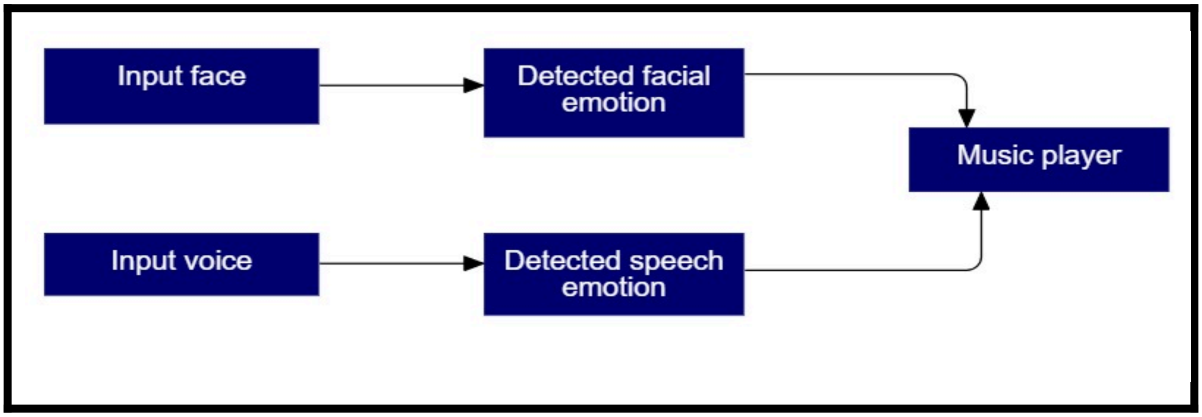


Fig.1 Abstract architecture of the system

This is the abstract view of our project. The system first requests for the human face for which the emotion has to be detected. This face has to be projected in the device camera. Once, the face is given, the emotion from the facial expressions is detected using the trained models. The next step is to provide the voice for which the emotion has to be detected. Once the voice is given, the emotion from the voice is detected using the trained models. Finally, these emotions are fused and the system is directed to the music player to play songs according to the predicted emotion.

6.2 DETAILED ARCHITECTURE

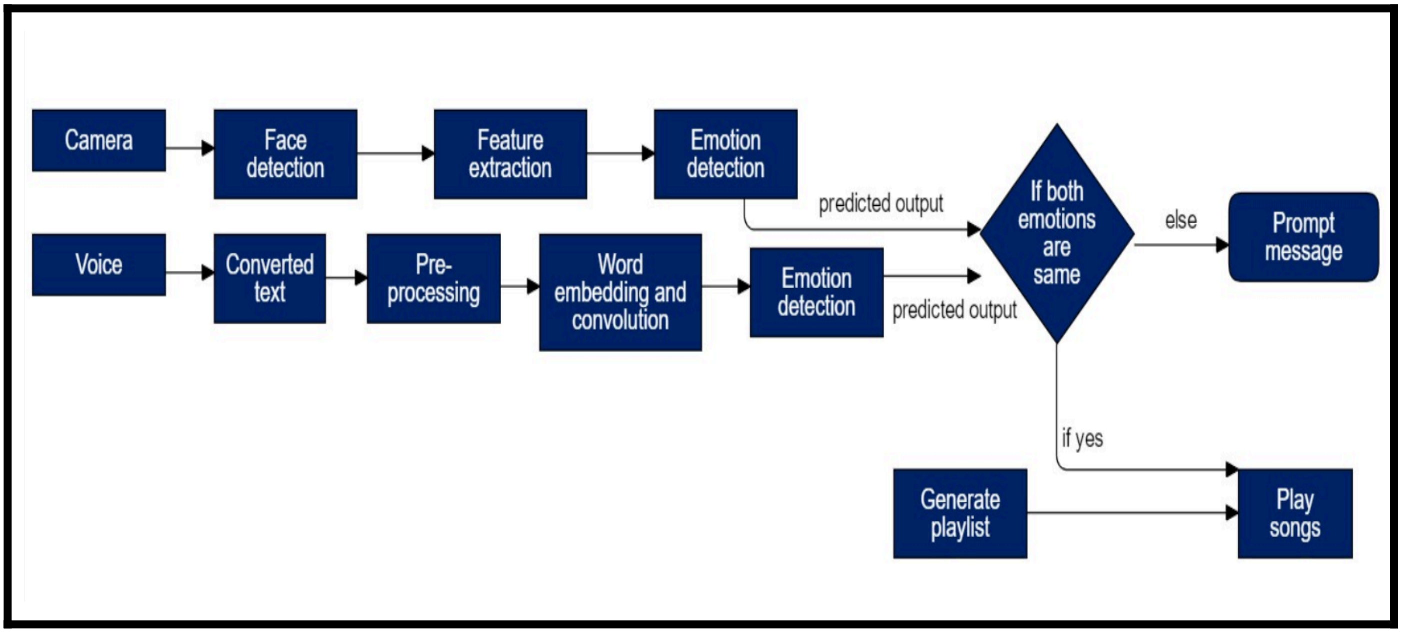


Fig.2 Architecture of the system

6.3 EXPLANATION OF THE ARCHITECTURE

Camera

The user provides the face for which emotion has to be detected through the device camera.

Face detection

Once the face is projected, the face has to be detected as the screen captures the whole background and we need only the face for our emotion detection.

For face detection **haarcascade model** is used.

This model accepts only grayscale images hence, the image is converted into grayscale and it detects the emotion of face with 5 features as given below.

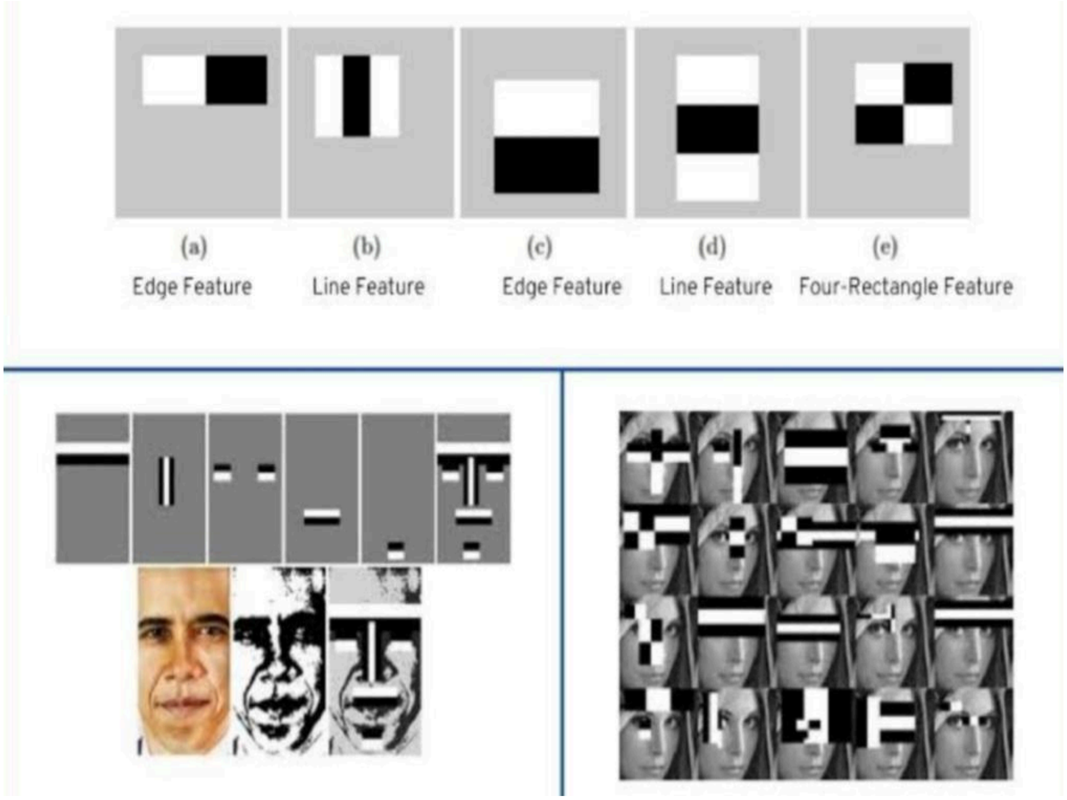


Fig.3- Haarcascade model-Overview

Feature extraction

Once the face is detected, the facial features such as eyes, nose, mouth needs to be extracted which are called as ROI (region of interest) for emotion detection from the converted grayscale image.

Emotion detection

With the extracted ROI the emotion is predicted using the trained **mini_XCEPTION model,** a CNN model whose working is represented below:

This predicted emotion (label) of the face is displayed once ‘q’ is pressed by which the image is captured.

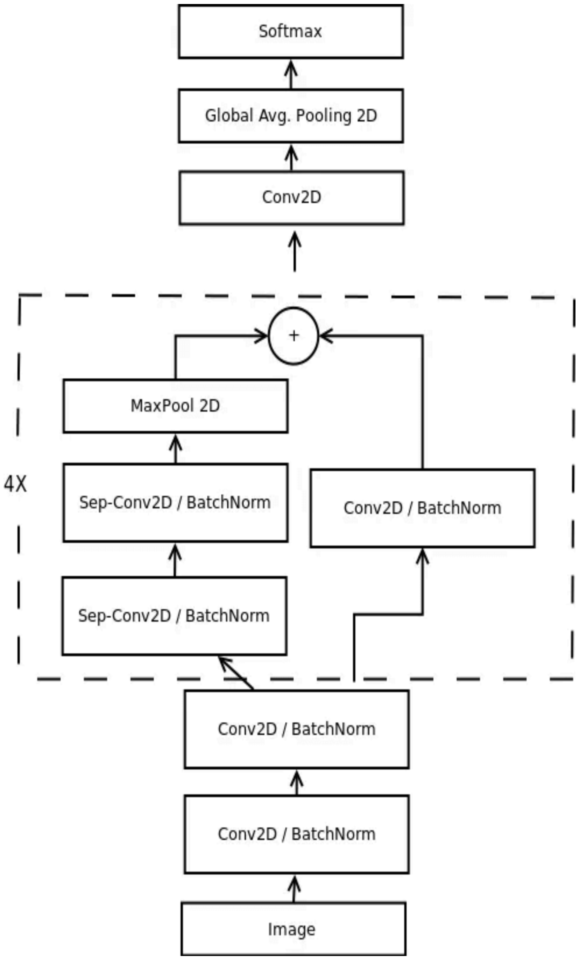


Fig.4- mini_XCEPTION model architecture

.

Voice- The voice for which the emotion has to be detected is given to the device microphone.

Converted text- The given input voice is converted into text using Google API from which the motion can be detected.

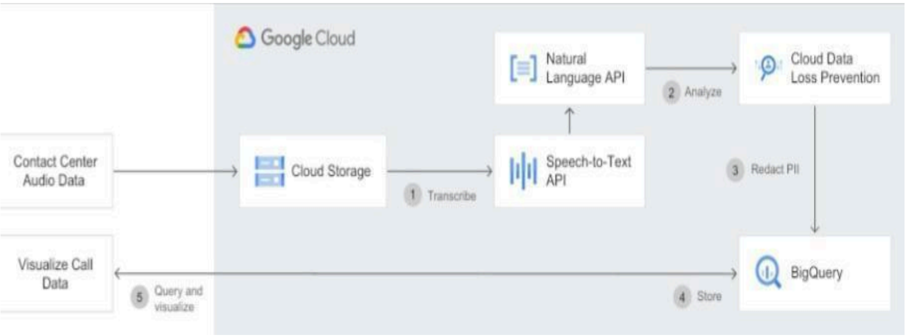


Fig.5- Google API:Speech to text conversion

Pre-processing

Once the text is obtained, we use NLP techniques such as tokenization, padding of the text before it is fed to the model. Tokenization refers to tokenizing the text into individual words i.e. from a sentence to word and padding of the text refers to equalising the input text to the same length. These 2 comprise the pre-processing of the text.

Word embedding and convolution

The padded text is fed to the embedding matrix. Embedding is a technique of mapping the words and their corresponding vectors as a matrix.With this embedding matrix the embed layer is created which is fed as the input to the CNN model. With the final activation of softmax, the model is used to predict the emotion from the text.

Emotion detection

With the CNN model, the emotion from text is predicted.

Decision checker

The system is directed to the music player only when the facial and voice emotions match. For this, the classes of emotions are classified into 2 primary classes- happy and sad. If both the emotions fall into happy or sad then the system is directed to the music player.

Generate playlist

A playlist for the list of happy songs and for the list of sad songs is generated from a list of songs. The song is played from this generated playlist randomly.

Play songs

If both the emotions are same, then a song at random is played from the list of happy or sad songs according to the detected emotion.

Prompt message

In case both the emotions do not match i.e one is happy and the other is sad, then this indicates that the emotion is not stable and an error message showing - “Face emotion and Voice emotion does not match”.

7. DETAILED MODULE DESIGN

Modules split

1. Emotion detection from facial expressions
2. Emotion detection from voice
3. Fusing and music generation

7.1. Module 1- Emotion detection from facial expressions

In this module, we detect the emotion of the face projected in the device camera using haarcascade model and mini_XCEPTION model. To do so,

Algorithm

#Training models

1. Load the dataset (fer2013) and normalise the images to (48,48) pixel size.
2. Load both the models and train the model.

#Predictions

3. Start the video streaming with the device camera.
4. Convert the RGB image to grayscale image.
5. With the grayscale image extract facial features and ROI.
6. With the extracted ROI predict the emotion.
7. The emotion predicted for the maximum number of times is the detected emotion.

Pseudo code

```
# loading dataset
    SET image_size as (48,48)
    define load_fer() {
    SET data as dataset (fer_2013)
    ADD pixels from data to list
    SET width and height to 48
    CREATE an empty array faces
    for (pixel_sequence in pixels) {
        RESHAPE faces to width,height
        RESIZE faces to image_size
        APPEND faces as type float
    }
    CONVERT emotions to dummy data and STORE as a matrix
    RETURN faces,emotions
    }
    # normalising
define preprocess_input(x, SET v2 as true) {
```

```
SET the datatype of x as float
SET x as x/255.0
if (v2) {
        SET x as x-0.5
        SET x as x*2.0
}
RETURN x
#model train
SET batch_size as 32
SET num_epochs as 10000
SET input_shape as (48, 48, 1)
SETvalidation_split as 0.2
SET verbose to 1
SET num_classes as 7
SET patience to 50
SET base_path as 'models/'
##load the model and compile it with the above parameters and pre-processed inputs
#prediction
##load the models and start the video streaming
while(true) {
CONVERT RGB image to grayscale image
SET Total_no_of_faces as len(faces)
if(len(faces) >0) {
        for( face in faces) {
                ##predict the emotion with the extracted ROI using the loaded model
                SET emotion_probability as the maximum of preds
                SET label as the maximum predicted emotion
                if( ‘q’ is pressed)
                then face is captured
                else
                continue
        }
else
continue
print(predicted emotion)
```

7.2. Module 2- Emotion detection from voice

In this module, we detect the emotion from the person's voice by converting voice to text using the Google API. From the converted text, the emotion is predicted in the following manner.

Algorithm

#Training models

1. Collect dataset from various sources and load them.
2. Clean the data.
3. Tokenize the words and hence pad them.
4. Convert the data into sequence of integers.
5. Create the embedding matrix with wiki-news-300d-1M.vec as the dataset
6. With these, create the embed layer which is the input to the model and train the model.

#Predictions

7. Convert speech to text using Google API.
8. With the loaded model, predict the emotion of the input text.

Pseudo code

```
#pre processing
define clean_text (data) {
REMOVE hashtags from the data
REMOVE username from the data
TOKENIZE data
RETURN data
}
##tokenize the data and convert the text to sequence of integers
SET word_index to index_of_words
SET vocab_size as len( index_of_words)+1
##pad the input data
#word embedding
```

```
define create_embedding_matrix(filepath, word_index, embedding_dim) {
    SET vocab_size as len(word_index) + 1  # Adding again 1 because of reserved 0 index
    SET embedding_matrix to zeros with the size of vocab_size, embedding_dim
    with open(filepath) as f:
        for line in f:
            SPLIT word and store it as *vector
            if word in word_index:
                SET idx as word_index[word]
                SET embedding_matrix[idx] as the array of vector, with the datatype float with   the dimension of [:embedding_dim]
    RETURN embedding_matrix
}
# Inspect unseen words
SET new_words to 0
for word in index_of_words:
    SET entry as embedd_matrix[index_of_words[word]]
    if all(v == 0 for v in entry):
        SET new_words = new_words + 1
#model train
SET kernel_size as 3
SET filters as 256
SET batch_size as 256
SET epochs as 6
##load the model and compile it with the above parameters and pre-processed inputs
#speech to text
with speech as source:
            adjust_for_ambient_noise(source)
            listen(source)
try:
    recognize_google(source, language = 'en-US')
if (UnknownValueError)
    print("Google Speech Recognition could not understand audio")
if(RequestError)
    print("Could not request results from Google Speech Recognition service”)
```

7.3. Module 3- Fusing and music generation

In this module we integrate the 2 modules to get the final emotion and play the music from the playlist according to the predicted emotion.

Algorithm

1. Classify the emotion classes into 2 primary classes - happy and sad.
2. If both the face emotion and voice emotion match, then play a song from the predicted emotion directory.
3. In case, if the emotions don’t match then give an error message.

Pseudo code

```
# label is the detected emotion from face and predictions is the detected
emotion from voice.
if ( label = happy || neutral || surprised)
        SET face_emotion as happy
else if ( label = sad || angry || disgust || scared)
        SET voice_emotion as sad

if (face_emotion == voice_emotion)
        if emotion is happy:
                Play song(randomly) from list of happy songs directory
        if emotion is sad:
                Play song(randomly) from list of sad songs directory
else:
   print("Face Emotion and Voice Emotion Does not Match")
```

8. IMPLEMENTATION

8.1. Initial setup requirements:

- FER library Facial Expression Recognition with a deep neural network using Tensorflow
- Keras libraries implemented in python.
- Dataset used is from Kaggle: Facial Expression Recognition.

- **Dependencies**: Tensorflow 1.7+ and Python 3.6+
  To install the required packages, run pip install -r requirements.txt .
- Installation of Command/Anaconda Prompt and necessary libraries for all the given modules using pip install
  A few example libraries used for face and voice emotion detection here are:
  Face emotion detection:
  pip install opencv- python
  pip install tensorflow
  pip install keras
  pip install adam
  pip install kwargs
  pip install cinit
  Voice recognition:
  import librosa
  import soundfile
  import os, glob,pickle
  import numpy as npfrom sklearn
- Jupiter notebook to test run and train model
- Webcam to detect face
- Google API to recognize and convert voice
- Python Speech Recognition module -pyaudio
- Visual Communication C++ (before pyaudio)

8.2. Tools Used

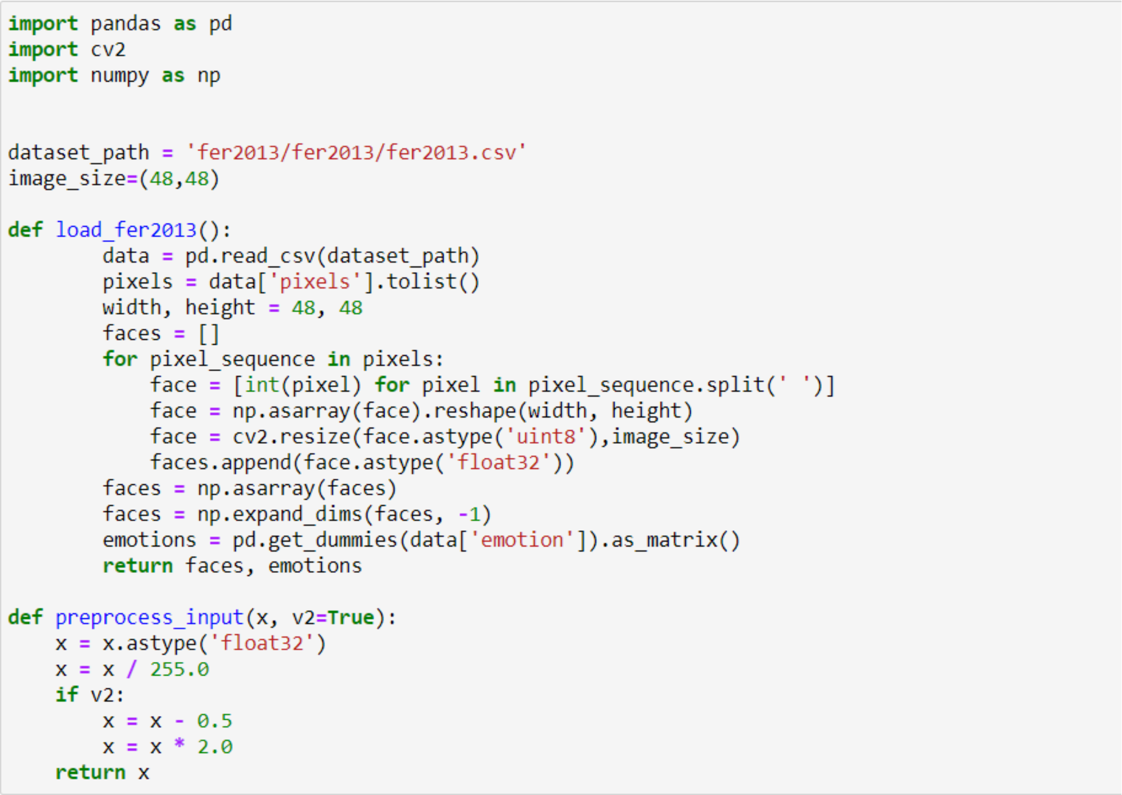
The tools used to implement this project are:

1. Python 3
2. TensorFlow
3. Keras
4. Matplotlib
5. OpenCV
6. Pyaudio

8.3. Code snippets of implementation:

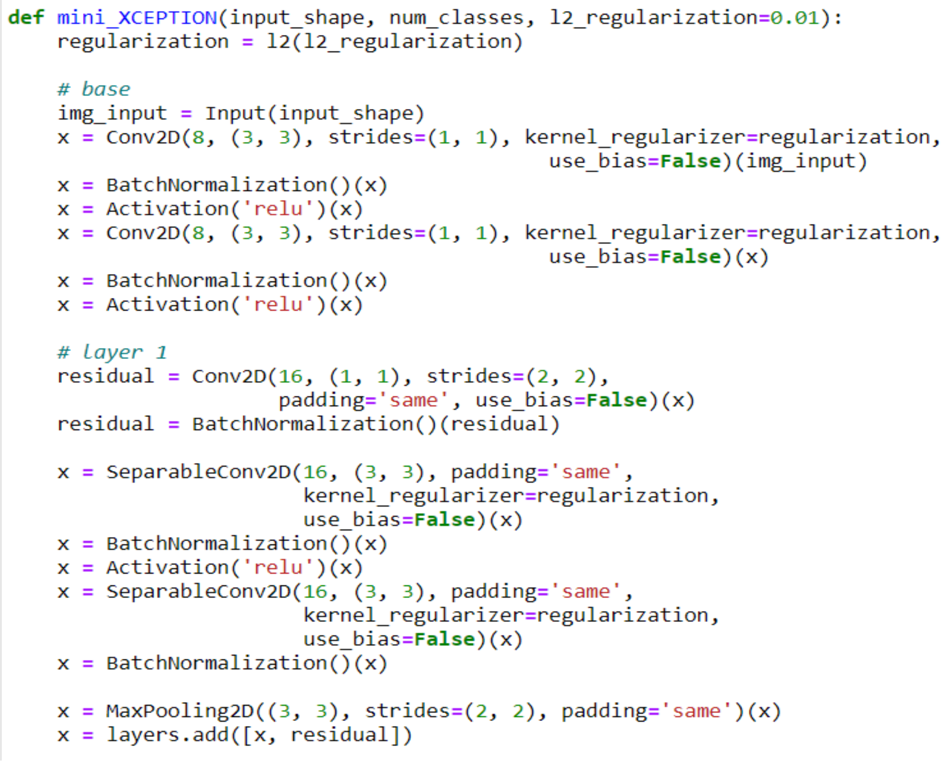
Emotion detection from face

- We load the fer_2013 dataset and resize the images to 48x48 and further normalise the images to load to the model.

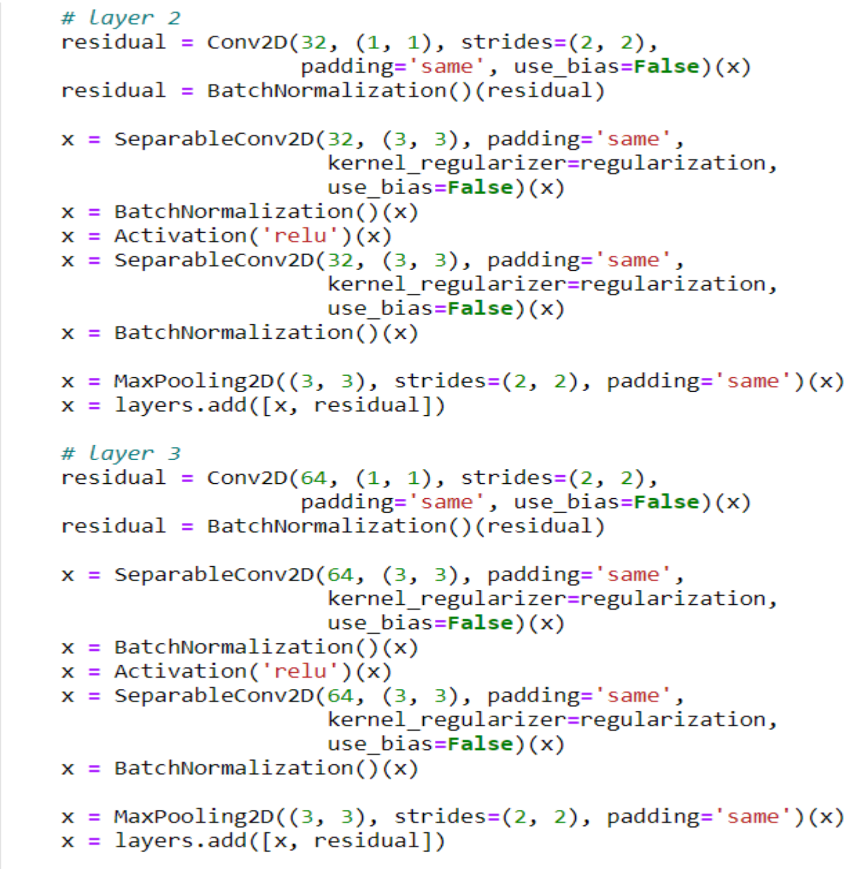


```
import pandas as pd
import cv2
import numpy as np


dataset_path = 'fer2013/fer2013/fer2013.csv'
image_size=(48,48)

def load_fer2013():
        data = pd.read_csv(dataset_path)
        pixels = data['pixels'].tolist()
        width, height = 48, 48
        faces = []
        for pixel_sequence in pixels:
            face = [int(pixel) for pixel in pixel_sequence.split(' ')]
            face = np.asarray(face).reshape(width, height)
            face = cv2.resize(face.astype('uint8'),image_size)
            faces.append(face.astype('float32'))
        faces = np.asarray(faces)
        faces = np.expand_dims(faces, -1)
        emotions = pd.get_dummies(data['emotion']).as_matrix()
        return faces, emotions

def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
    return x
```

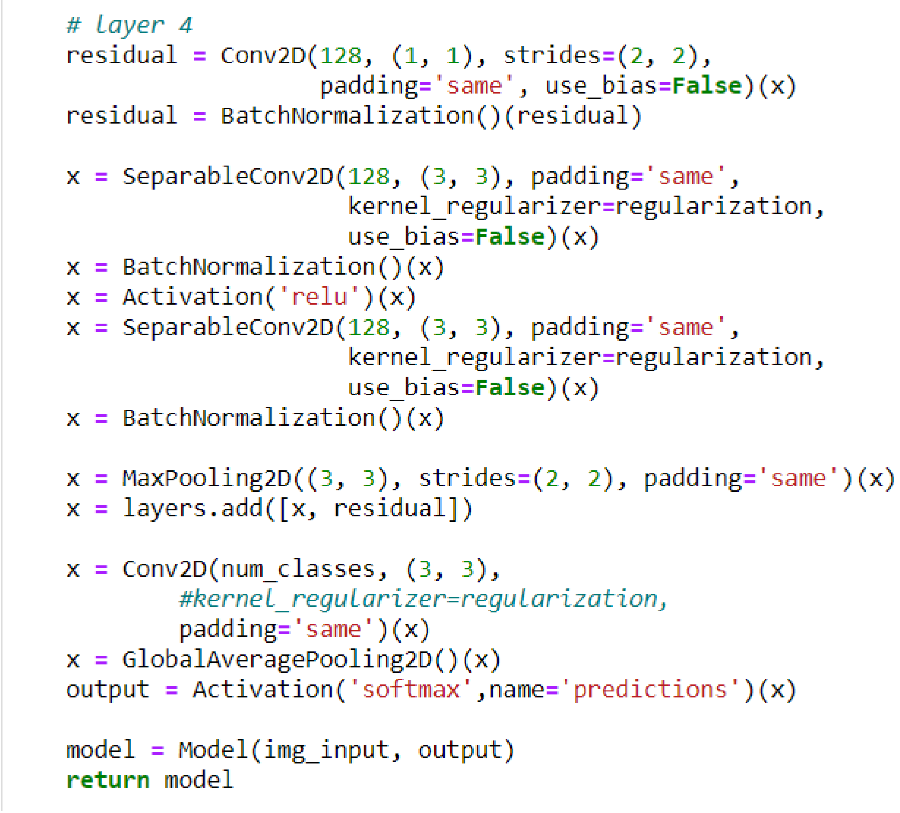
- Building the mini_XCEPTION model:



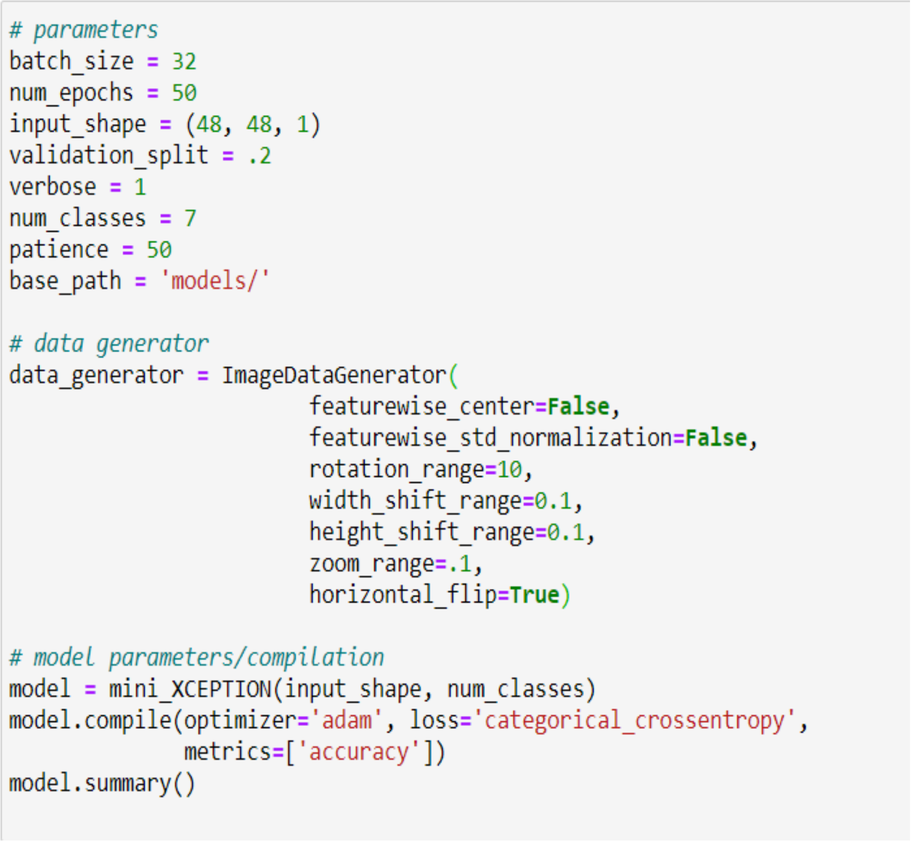
```
def mini_XCEPTION(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
                                            use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
                                            use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # layer 1
    residual = Conv2D(16, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])
```



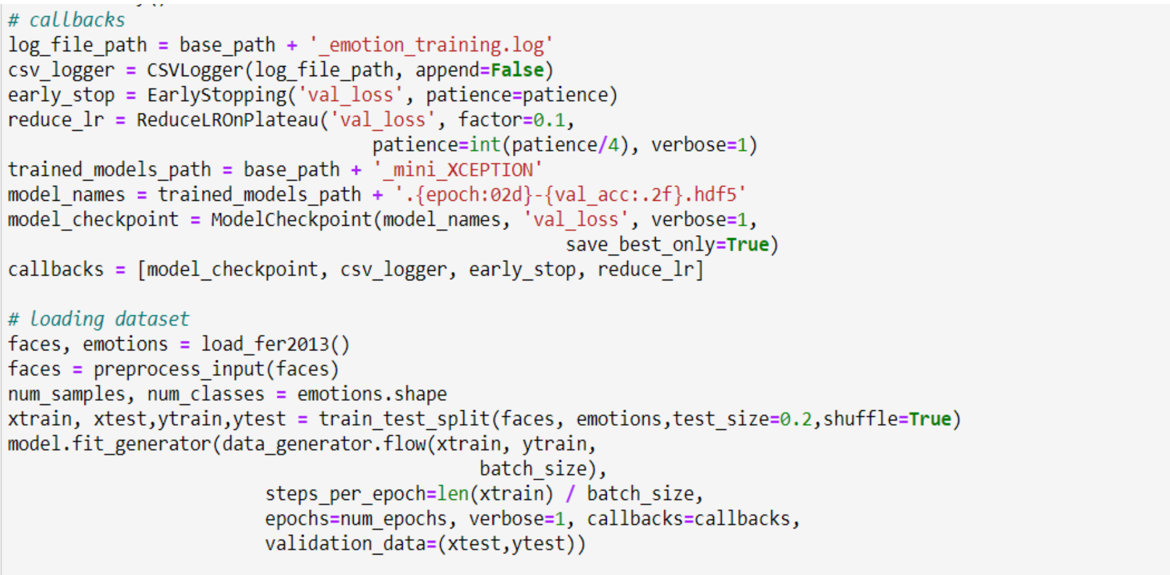
```python
# layer 2
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# layer 3
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
```



```python
# layer 4
residual = Conv2D(128, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(128, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(128, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

x = Conv2D(num_classes, (3, 3),
        #kernel_regularizer=regularization,
        padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax',name='predictions')(x)

model = Model(img_input, output)
return model
```

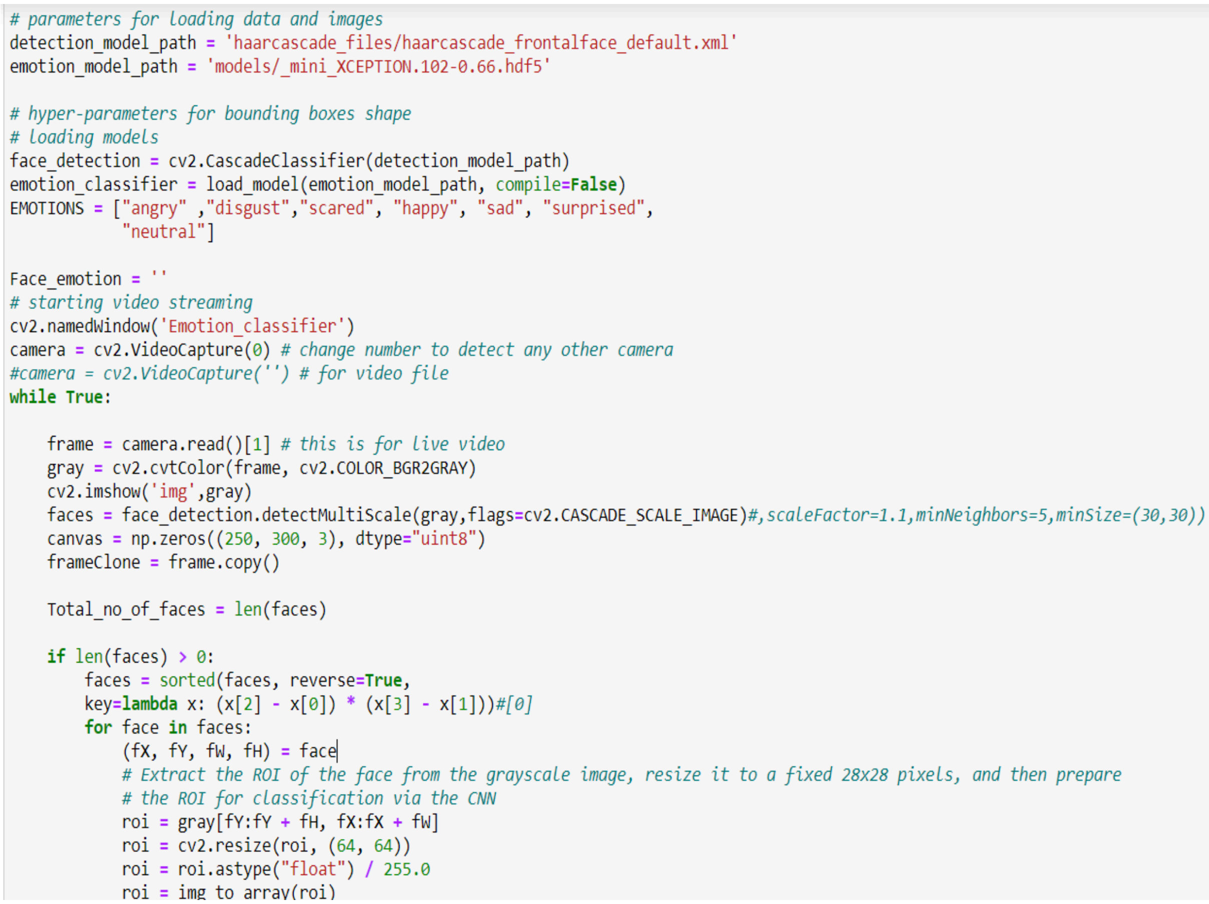


```
# parameters
batch_size = 32
num_epochs = 50
input_shape = (48, 48, 1)
validation_split = .2
verbose = 1
num_classes = 7
patience = 50
base_path = 'models/'

# data generator
data_generator = ImageDataGenerator(
                        featurewise_center=False,
                        featurewise_std_normalization=False,
                        rotation_range=10,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        zoom_range=.1,
                        horizontal_flip=True)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

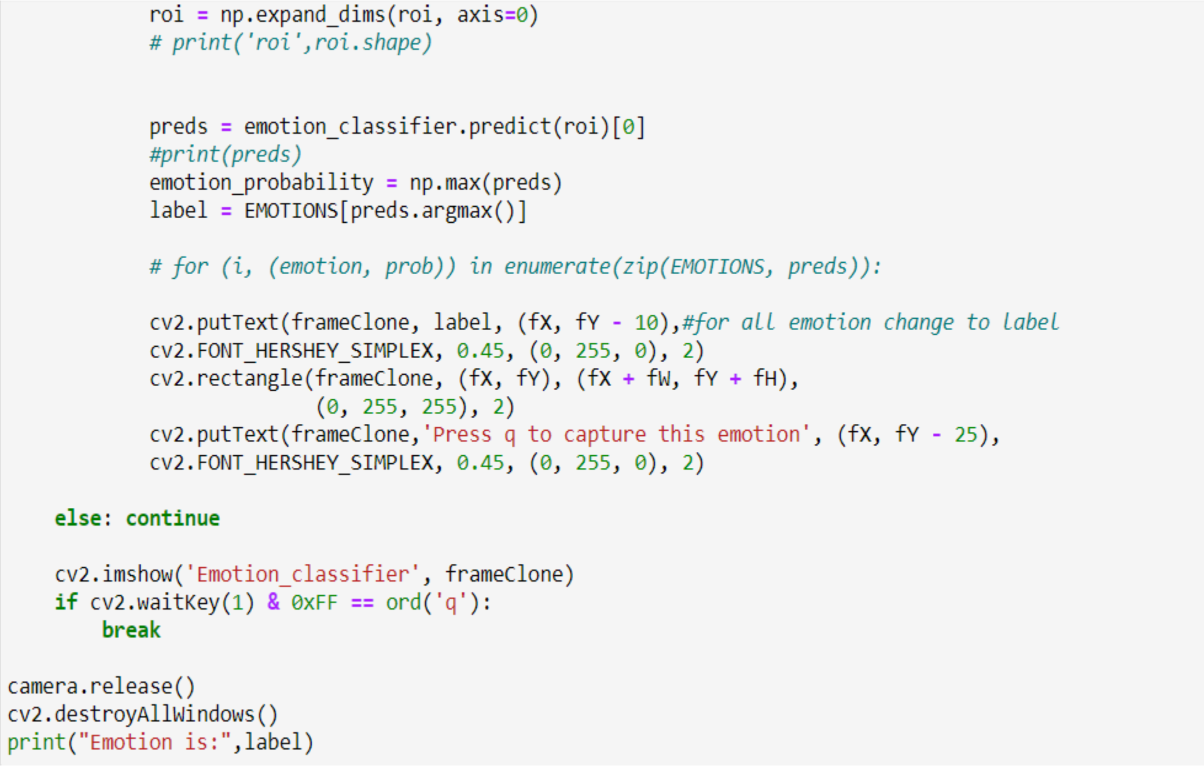


```
# callbacks
log_file_path = base_path + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                                  patience=int(patience/4), verbose=1)
trained_models_path = base_path + '_mini_XCEPTION'
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
                                                    save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

# loading dataset
faces, emotions = load_fer2013()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
xtrain, xtest,ytrain,ytest = train_test_split(faces, emotions,test_size=0.2,shuffle=True)
model.fit_generator(data_generator.flow(xtrain, ytrain,
                                            batch_size),
                        steps_per_epoch=len(xtrain) / batch_size,
                        epochs=num_epochs, verbose=1, callbacks=callbacks,
                        validation_data=(xtest,ytest))
```

- Loading the haarcascade model (pretrained model for face detection) and the mini_XCEPTION model and then start the video streaming to detect the emotion of the projected face.

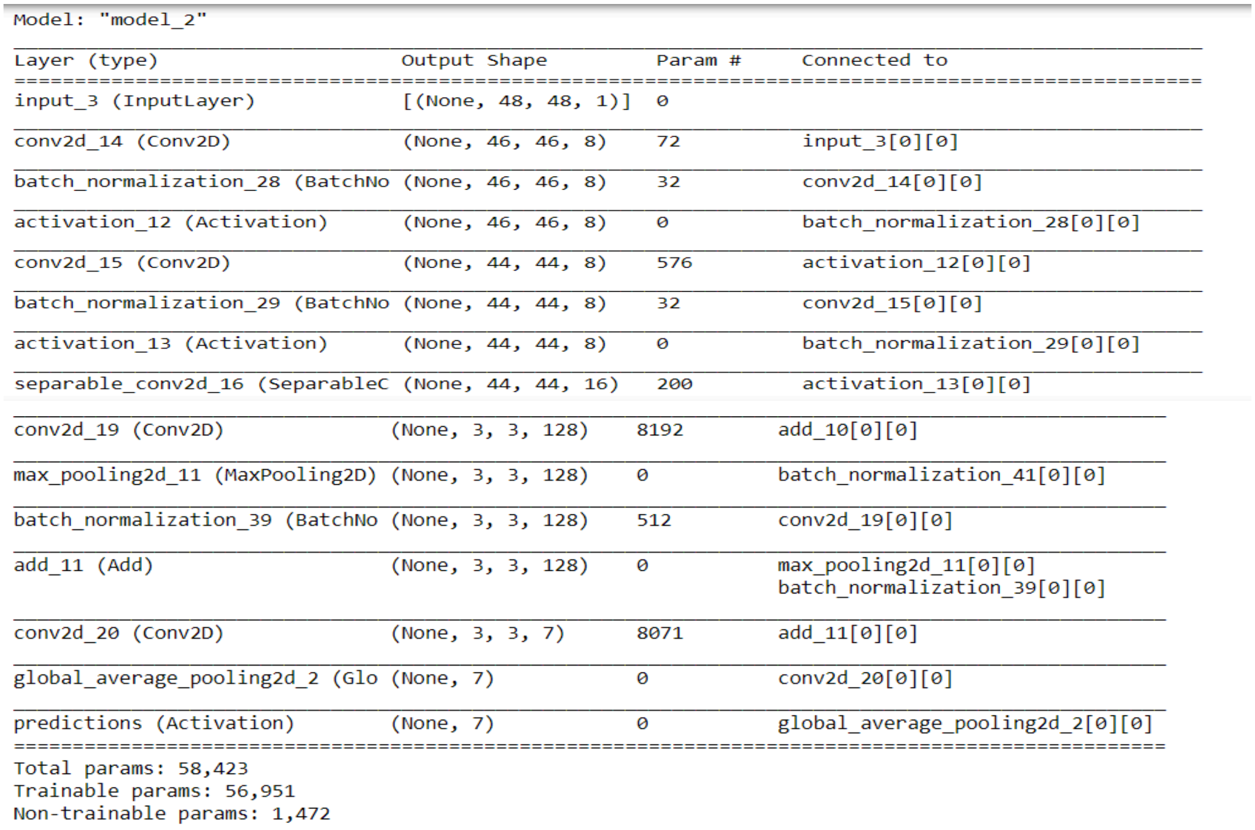


```
# parameters for loading data and images
detection_model_path = 'haarcascade_files/haarcascade_frontalface_default.xml'
emotion_model_path = 'models/_mini_XCEPTION.102-0.66.hdf5'

# hyper-parameters for bounding boxes shape
# loading models
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry" ,"disgust","scared", "happy", "sad", "surprised",
            "neutral"]

Face_emotion = ''
# starting video streaming
cv2.namedWindow('Emotion_classifier')
camera = cv2.VideoCapture(0) # change number to detect any other camera
#camera = cv2.VideoCapture('') # for video file
while True:

    frame = camera.read()[1] # this is for live video
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('img',gray)
    faces = face_detection.detectMultiScale(gray,flags=cv2.CASCADE_SCALE_IMAGE)#,scaleFactor=1.1,minNeighbors=5,minSize=(30,30))
    canvas = np.zeros((250, 300, 3), dtype="uint8")
    frameClone = frame.copy()

    Total_no_of_faces = len(faces)

    if len(faces) > 0:
        faces = sorted(faces, reverse=True,
        key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))#[0]
        for face in faces:
            (fX, fY, fW, fH) = face
            # Extract the ROI of the face from the grayscale image, resize it to a fixed 28x28 pixels, and then prepare
            # the ROI for classification via the CNN
            roi = gray[fY:fY + fH, fX:fX + fW]
            roi = cv2.resize(roi, (64, 64))
            roi = roi.astype("float") / 255.0
            roi = img_to_array(roi)
```



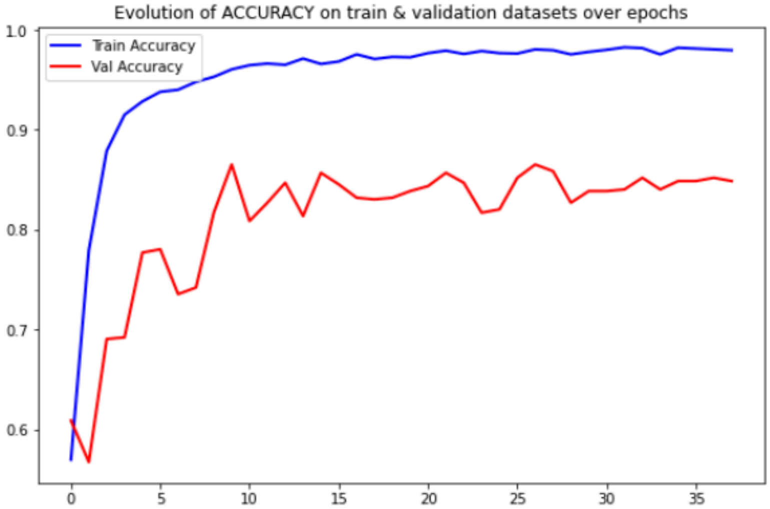
```
            roi = np.expand_dims(roi, axis=0)
            # print('roi',roi.shape)


            preds = emotion_classifier.predict(roi)[0]
            #print(preds)
            emotion_probability = np.max(preds)
            label = EMOTIONS[preds.argmax()]

            # for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):

            cv2.putText(frameClone, label, (fX, fY - 10),#for all emotion change to label
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)
            cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
                   (0, 255, 255), 2)
            cv2.putText(frameClone,'Press q to capture this emotion', (fX, fY - 25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 255, 0), 2)

    else: continue

    cv2.imshow('Emotion_classifier', frameClone)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

camera.release()
cv2.destroyAllWindows()
print("Emotion is:",label)
```

Model summary

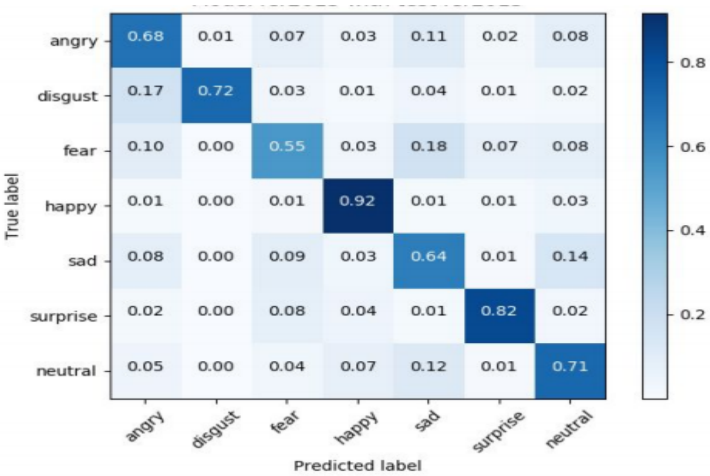


```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 48, 48, 1)]  0
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 46, 46, 8)    72          input_3[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, 46, 46, 8)    32          conv2d_14[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 46, 46, 8)    0           batch_normalization_28[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 44, 44, 8)    576         activation_12[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 44, 44, 8)    32          conv2d_15[0][0]
__________________________________________________________________________________________________
activation_13 (Activation)      (None, 44, 44, 8)    0           batch_normalization_29[0][0]
__________________________________________________________________________________________________
separable_conv2d_16 (SeparableC (None, 44, 44, 16)   200         activation_13[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, 3, 3, 128)    8192        add_10[0][0]
__________________________________________________________________________________________________
max_pooling2d_11 (MaxPooling2D) (None, 3, 3, 128)    0           batch_normalization_41[0][0]
__________________________________________________________________________________________________
batch_normalization_39 (BatchNo (None, 3, 3, 128)    512         conv2d_19[0][0]
__________________________________________________________________________________________________
add_11 (Add)                    (None, 3, 3, 128)    0           max_pooling2d_11[0][0]
                                                                 batch_normalization_39[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 3, 3, 7)      8071        add_11[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_2 (Glo (None, 7)            0           conv2d_20[0][0]
__________________________________________________________________________________________________
predictions (Activation)        (None, 7)            0           global_average_pooling2d_2[0][0]
==================================================================================================
Total params: 58,423
Trainable params: 56,951
Non-trainable params: 1,472
```

Accuracy plot and confusion matrix

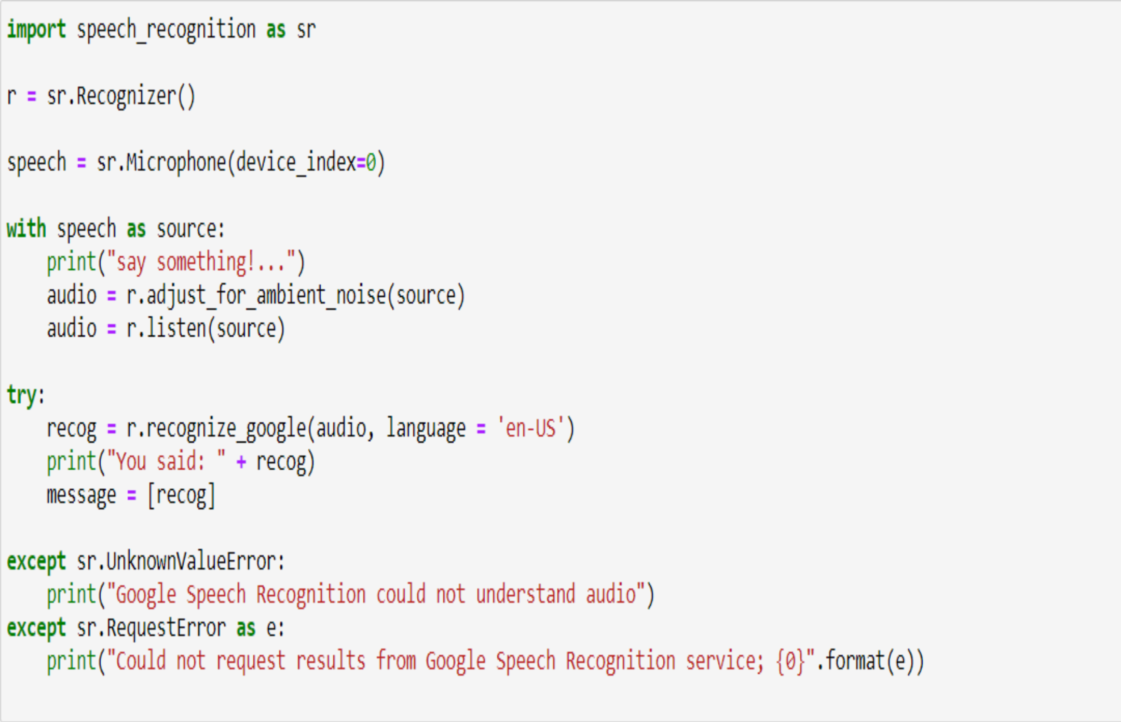


Accuracy achieved = 85%



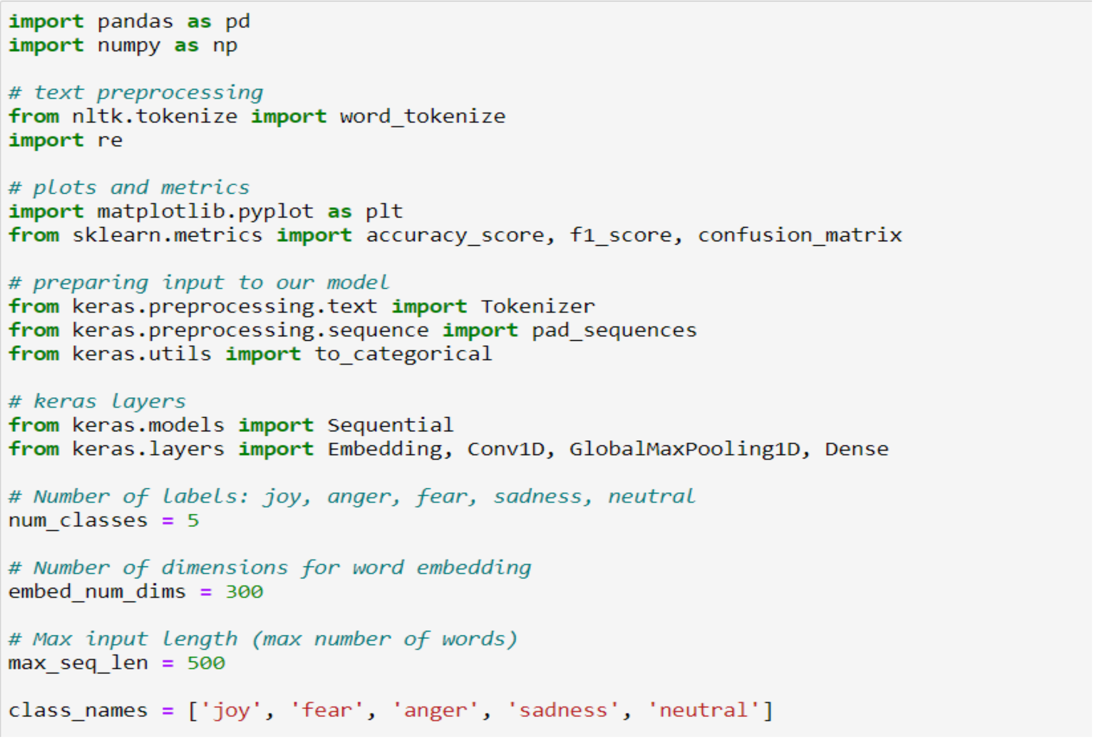
Emotion detection from voice

- Convert the speech recognised by the device to text using Google API.



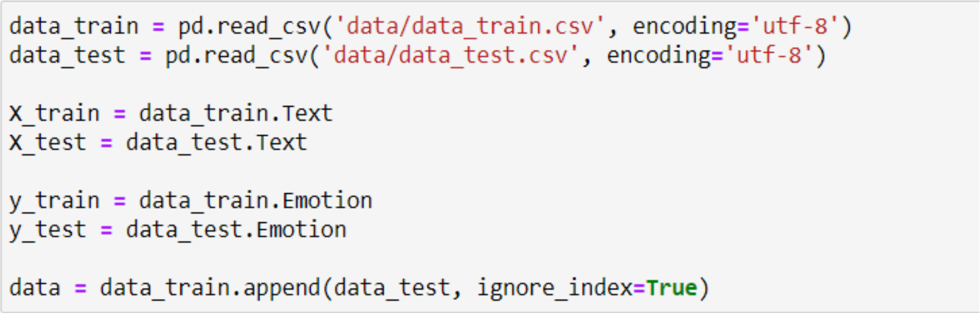
```
import speech_recognition as sr

r = sr.Recognizer()

speech = sr.Microphone(device_index=0)

with speech as source:
    print("say something!...")
    audio = r.adjust_for_ambient_noise(source)
    audio = r.listen(source)

try:
    recog = r.recognize_google(audio, language = 'en-US')
    print("You said: " + recog)
    message = [recog]

except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

- Importing the necessary libraries:

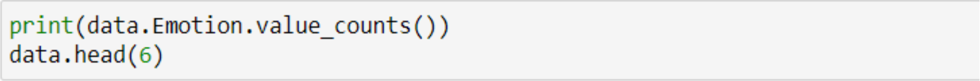


```
import pandas as pd
import numpy as np

# text preprocessing
from nltk.tokenize import word_tokenize
import re

# plots and metrics
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

# preparing input to our model
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

# keras layers
from keras.models import Sequential
from keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense

# Number of labels: joy, anger, fear, sadness, neutral
num_classes = 5

# Number of dimensions for word embedding
embed_num_dims = 300

# Max input length (max number of words)
max_seq_len = 500

class_names = ['joy', 'fear', 'anger', 'sadness', 'neutral']
```

- Loading the created dataset and check the emotion count of each emotion-joy,fear,anger,sadness and neutral.

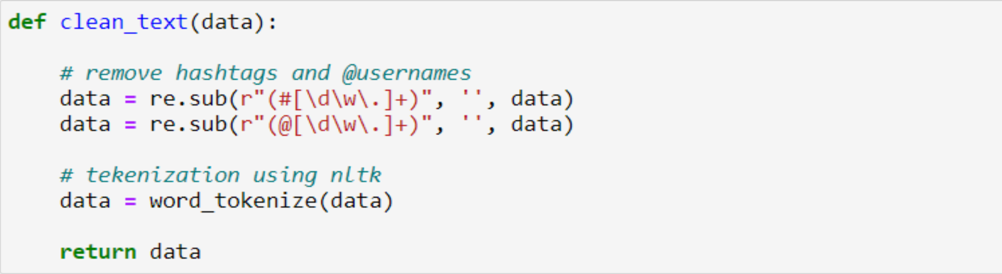


```
data_train = pd.read_csv('data/data_train.csv', encoding='utf-8')
data_test = pd.read_csv('data/data_test.csv', encoding='utf-8')

X_train = data_train.Text
X_test = data_test.Text

y_train = data_train.Emotion
y_test = data_test.Emotion

data = data_train.append(data_test, ignore_index=True)
```

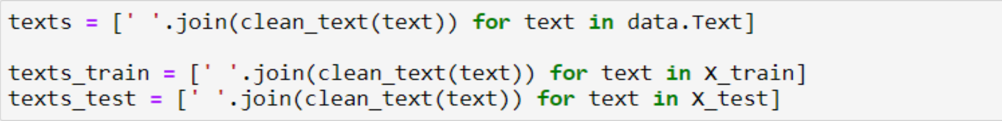


```
print(data.Emotion.value_counts())
data.head(6)
```

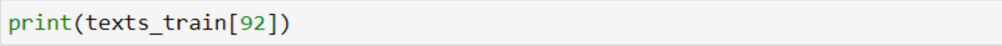
- Remove the unnecessary usernames, hashtags from the data and tokenize the data.



```python
def clean_text(data):

    # remove hashtags and @usernames
    data = re.sub(r"(#[\d\w\.]+)", '', data)
    data = re.sub(r"(@[\d\w\.]+)", '', data)

    # tekenization using nltk
    data = word_tokenize(data)

    return data
```

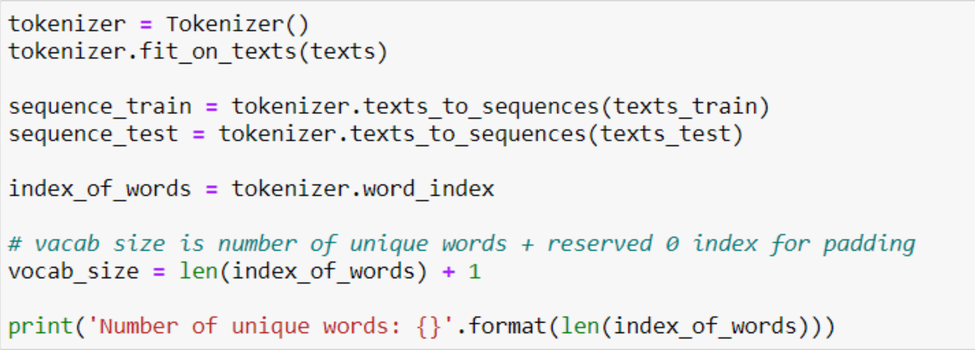


```python
texts = [' '.join(clean_text(text)) for text in data.Text]

texts_train = [' '.join(clean_text(text)) for text in X_train]
texts_test = [' '.join(clean_text(text)) for text in X_test]
```



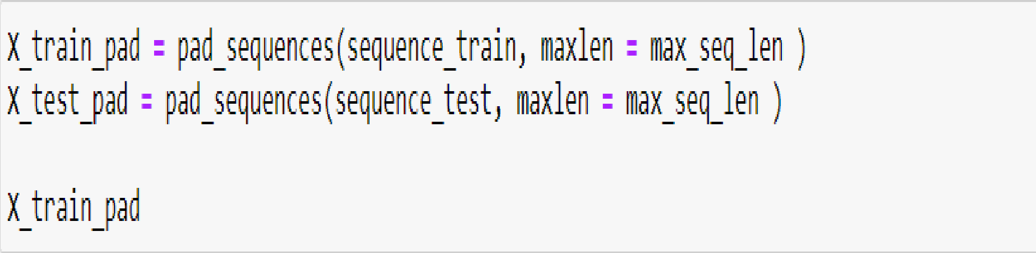
```python
print(texts_train[92])
```

- Convert the tokenized data to sequence of integers for vectorizing.



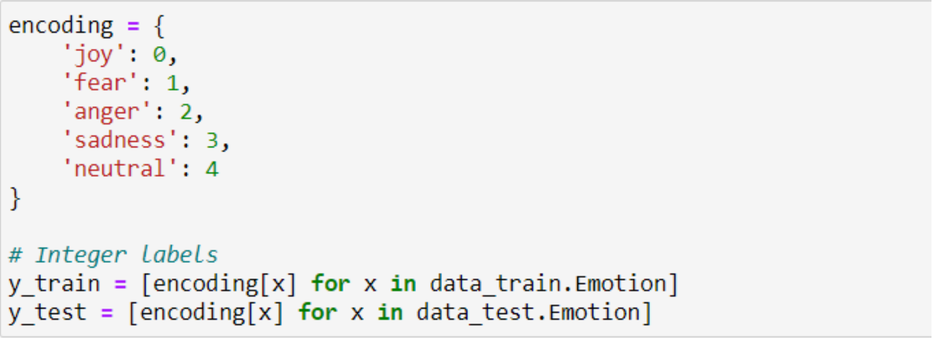
```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)

sequence_train = tokenizer.texts_to_sequences(texts_train)
sequence_test = tokenizer.texts_to_sequences(texts_test)

index_of_words = tokenizer.word_index

# vacab size is number of unique words + reserved 0 index for padding
vocab_size = len(index_of_words) + 1

print('Number of unique words: {}'.format(len(index_of_words)))
```

- Pad the data so that each integer sequence is of the same length.

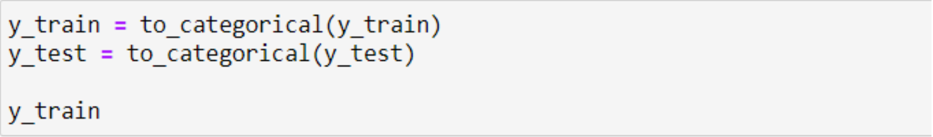


```python
X_train_pad = pad_sequences(sequence_train, maxlen = max_seq_len )
X_test_pad = pad_sequences(sequence_test, maxlen = max_seq_len )

X_train_pad
```

- The one hot encoding process for the emotions as they also need to be as integers.

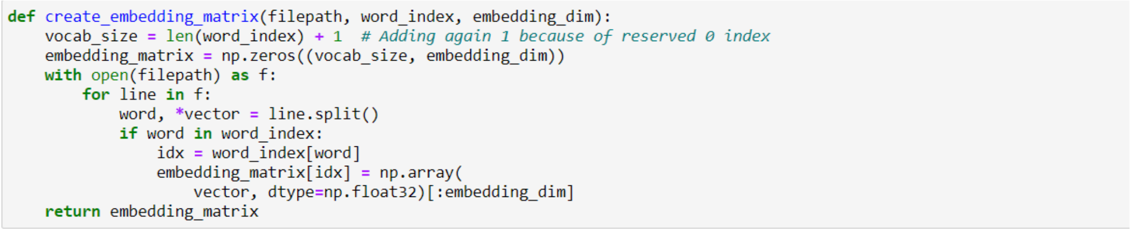


```python
encoding = {
    'joy': 0,
    'fear': 1,
    'anger': 2,
    'sadness': 3,
    'neutral': 4
}

# Integer labels
y_train = [encoding[x] for x in data_train.Emotion]
y_test = [encoding[x] for x in data_test.Emotion]
```

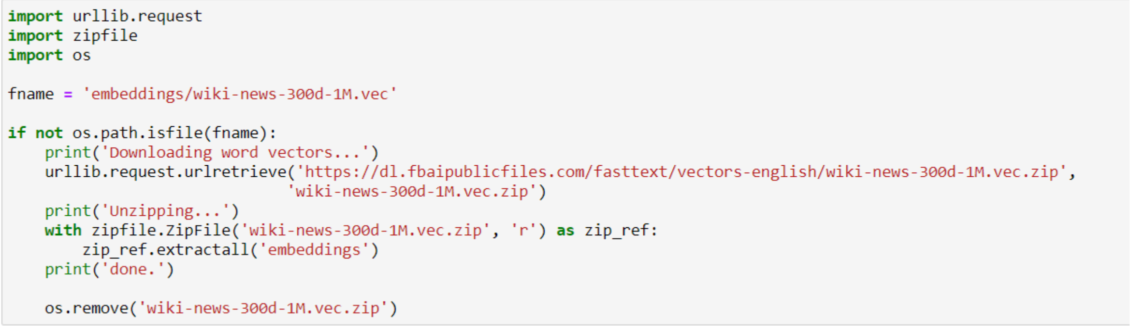


```python
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

y_train
```

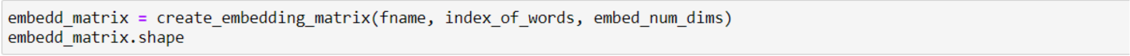
- With the pre-trained wiki dataset on word2vector, create an embedding matrix. As most of the words in our dataset will not be present in the wiki dataset, we need to generate vectors for those words also and append to the embedding matrix.



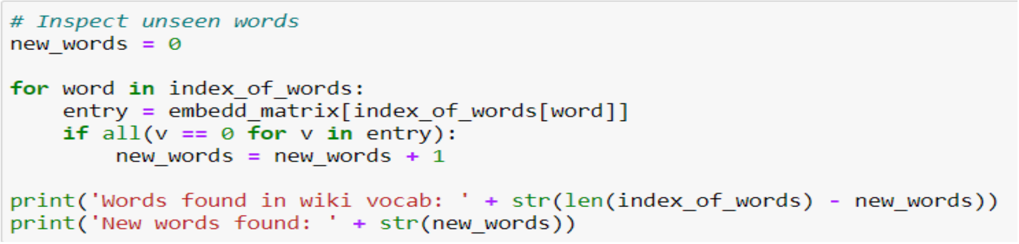
```python
def create_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # Adding again 1 because of reserved 0 index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))
    with open(filepath) as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(
                    vector, dtype=np.float32)[:embedding_dim]
    return embedding_matrix
```



```python
import urllib.request
import zipfile
import os

fname = 'embeddings/wiki-news-300d-1M.vec'

if not os.path.isfile(fname):
    print('Downloading word vectors...')
    urllib.request.urlretrieve('https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip',
                              'wiki-news-300d-1M.vec.zip')
    print('Unzipping...')
    with zipfile.ZipFile('wiki-news-300d-1M.vec.zip', 'r') as zip_ref:
        zip_ref.extractall('embeddings')
    print('done.')

    os.remove('wiki-news-300d-1M.vec.zip')
```

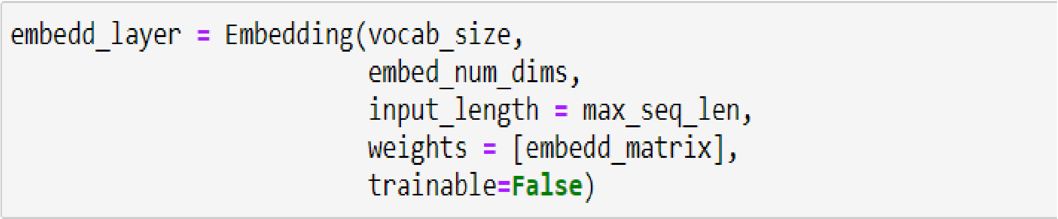


```python
embedd_matrix = create_embedding_matrix(fname, index_of_words, embed_num_dims)
embedd_matrix.shape
```

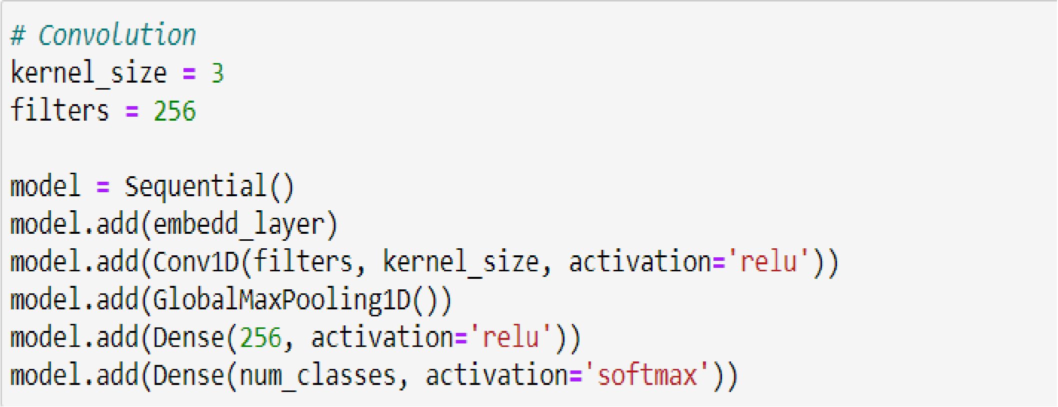


```python
# Inspect unseen words
new_words = 0

for word in index_of_words:
    entry = embedd_matrix[index_of_words[word]]
    if all(v == 0 for v in entry):
        new_words = new_words + 1

print('Words found in wiki vocab: ' + str(len(index_of_words) - new_words))
print('New words found: ' + str(new_words))
```

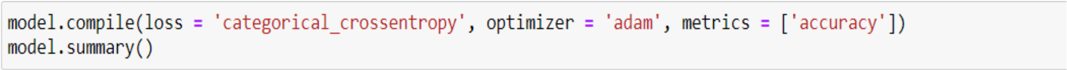
- With the embedding matrix as the weights, create the embedd layer to pass to the CNN model.



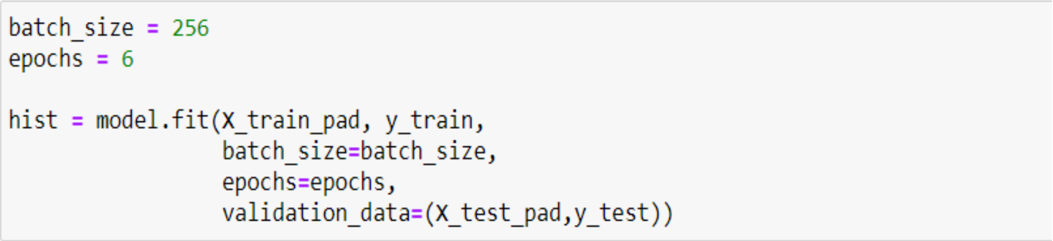
```
embedd_layer = Embedding(vocab_size,
                         embed_num_dims,
                         input_length = max_seq_len,
                         weights = [embedd_matrix],
                         trainable=False)
```



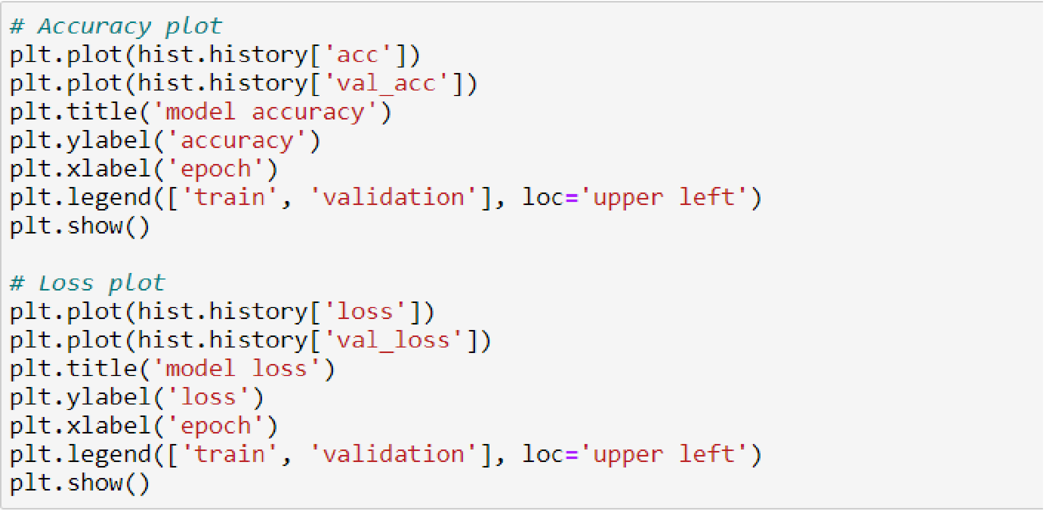
```
# Convolution
kernel_size = 3
filters = 256

model = Sequential()
model.add(embedd_layer)
model.add(Conv1D(filters, kernel_size, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(256, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```



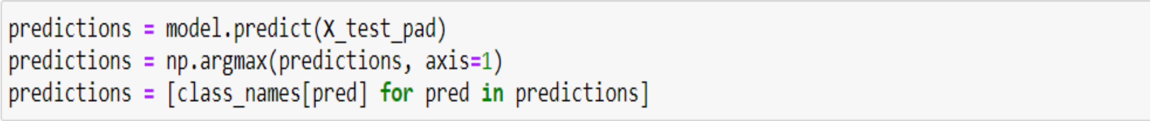
```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.summary()
```



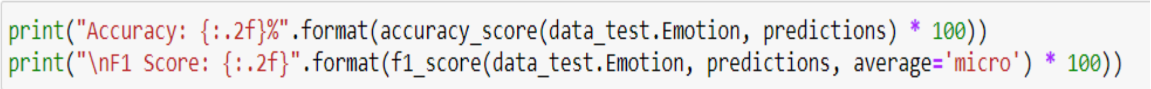
```
batch_size = 256
epochs = 6

hist = model.fit(X_train_pad, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 validation_data=(X_test_pad,y_test))
```



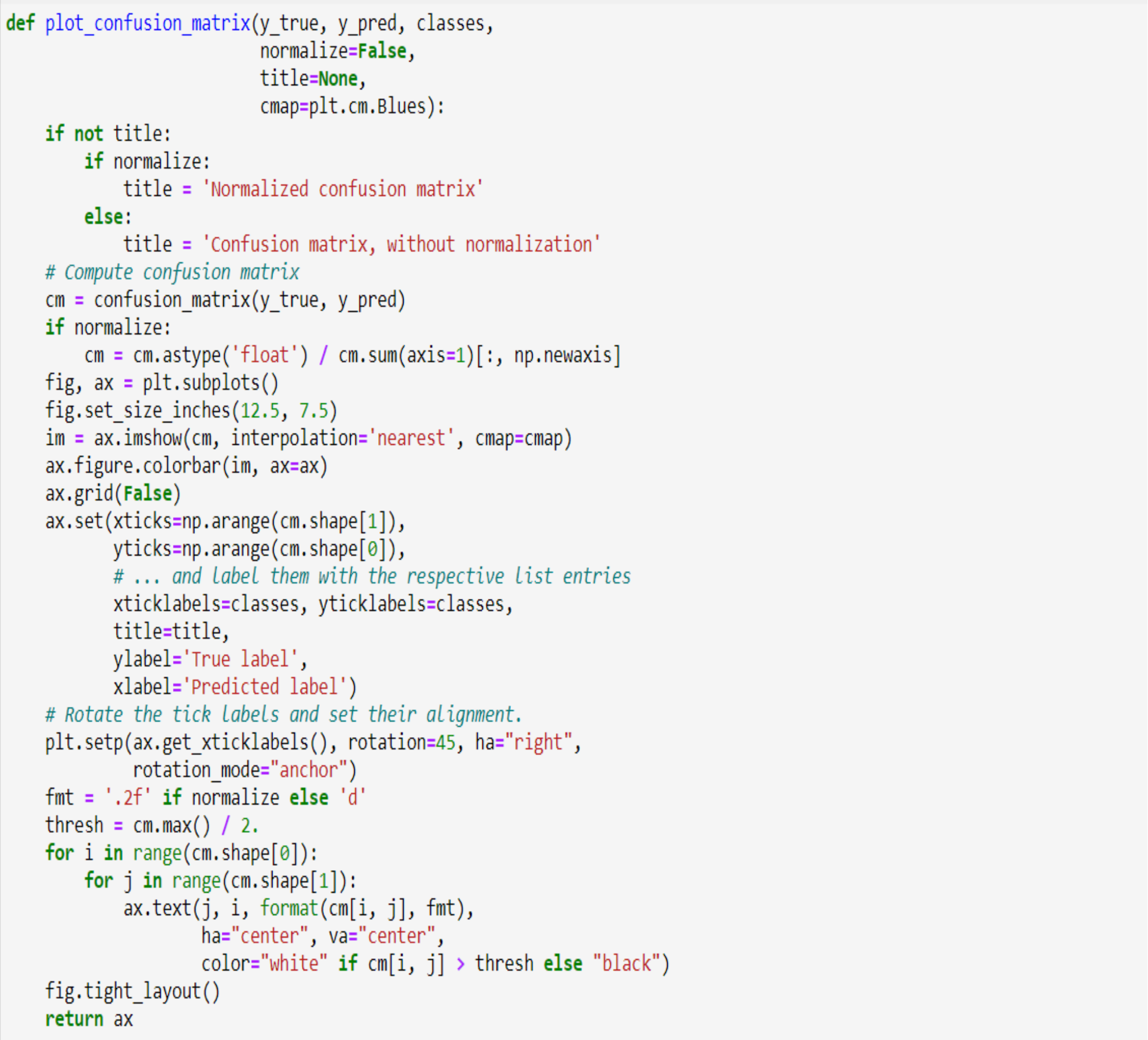
```
# Accuracy plot
plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

# Loss plot
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



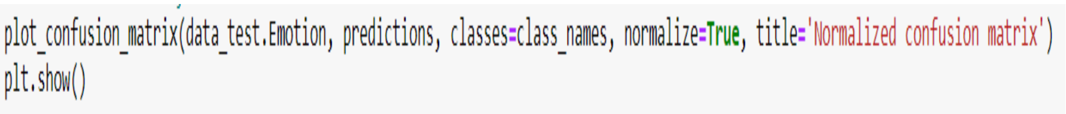
```
predictions = model.predict(X_test_pad)
predictions = np.argmax(predictions, axis=1)
predictions = [class_names[pred] for pred in predictions]
```



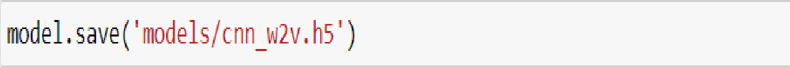
```
print("Accuracy: {:.2f}%".format(accuracy_score(data_test.Emotion, predictions) * 100))
print("\nF1 Score: {:.2f}".format(f1_score(data_test.Emotion, predictions, average='micro') * 100))
```



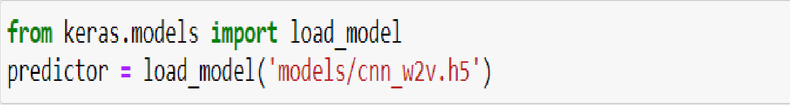
```
def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'
    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fig, ax = plt.subplots()
    fig.set_size_inches(12.5, 7.5)
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.grid(False)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')
    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax
```



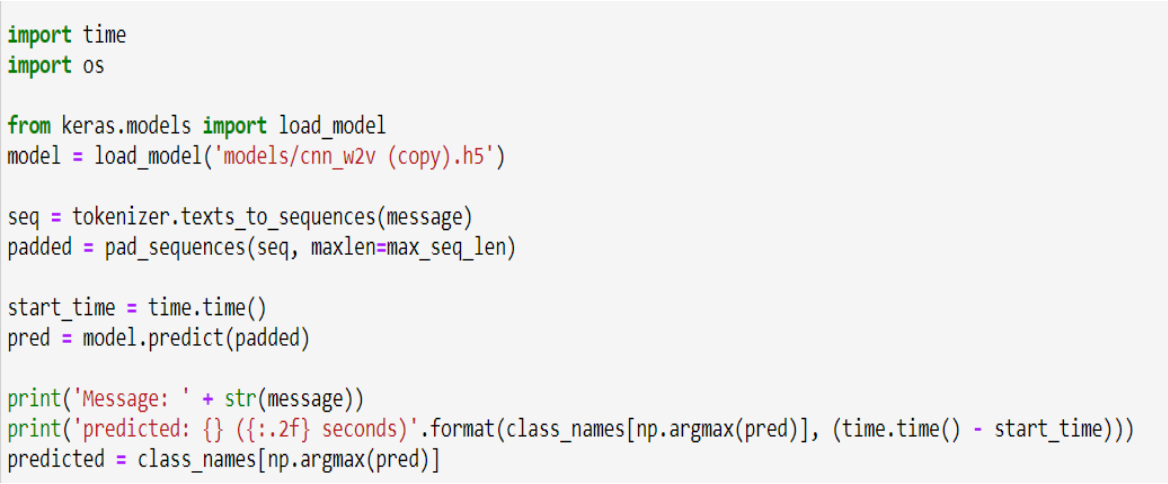
```
plot_confusion_matrix(data_test.Emotion, predictions, classes=class_names, normalize=True, title='Normalized confusion matrix')
plt.show()
```



```
model.save('models/cnn_w2v.h5')
```

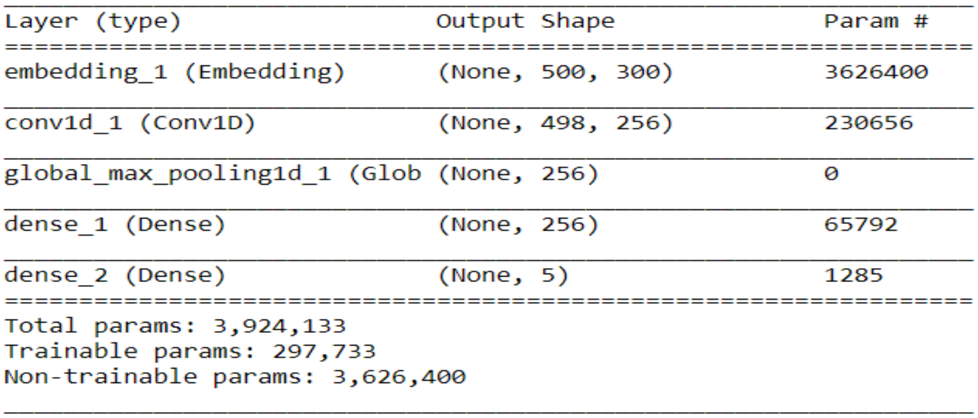


```
from keras.models import load_model
predictor = load_model('models/cnn_w2v.h5')
```



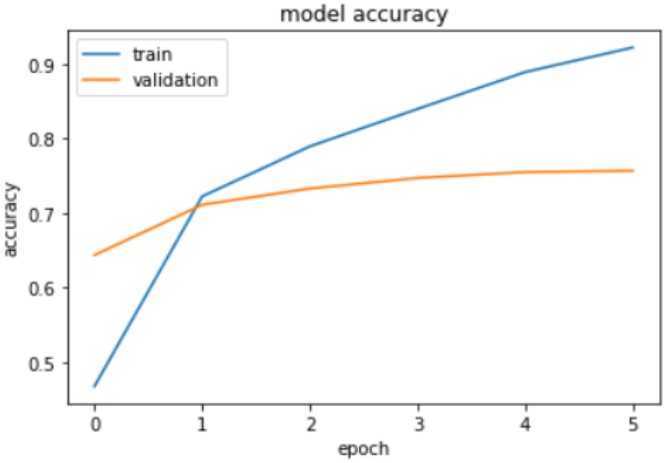
```
import time
import os

from keras.models import load_model
model = load_model('models/cnn_w2v (copy).h5')

seq = tokenizer.texts_to_sequences(message)
padded = pad_sequences(seq, maxlen=max_seq_len)

start_time = time.time()
pred = model.predict(padded)

print('Message: ' + str(message))
print('predicted: {} ({:.2f} seconds)'.format(class_names[np.argmax(pred)], (time.time() - start_time)))
predicted = class_names[np.argmax(pred)]
```

Model summary-



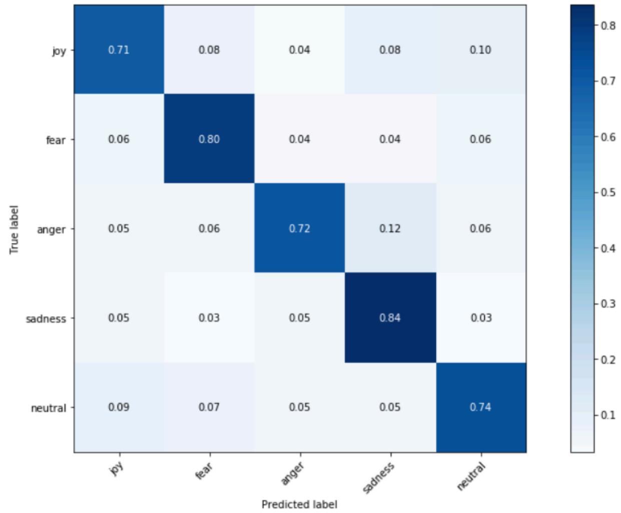
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 500, 300)          3626400   
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 498, 256)          230656    
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792     
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1285      
=================================================================
Total params: 3,924,133
Trainable params: 297,733
Non-trainable params: 3,626,400
_________________________________________________________________
```

Accuracy plots and Confusion matrix



Accuracy achieved= 76%

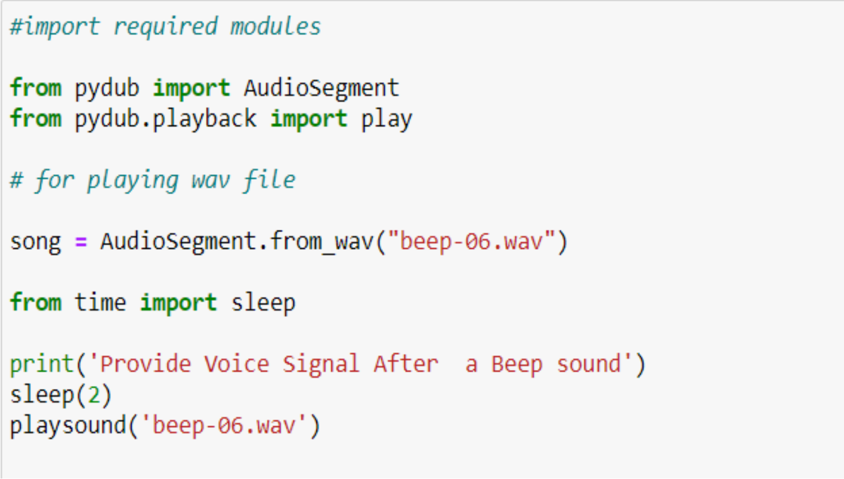
Confusion matrix:



	joy	fear	anger	sadness	neutral
joy	0.71	0.08	0.04	0.08	0.10
fear	0.06	0.80	0.04	0.04	0.06
anger	0.05	0.06	0.72	0.12	0.06
sadness	0.05	0.03	0.05	0.84	0.03
neutral	0.09	0.07	0.05	0.05	0.74

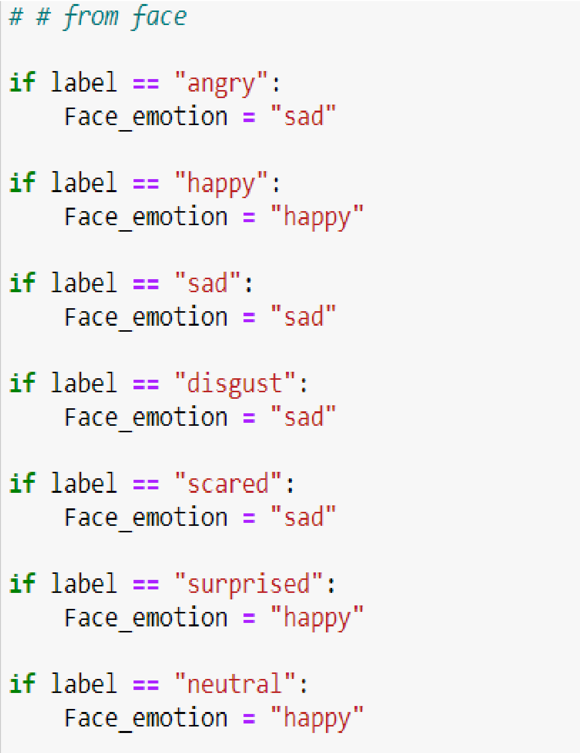
Integration and music playing

- Include a beep sound on the detection of face emotion as the user will get a flow in the process of the system. Once the beep sound is herald, the user can proceed to the voice emotion.

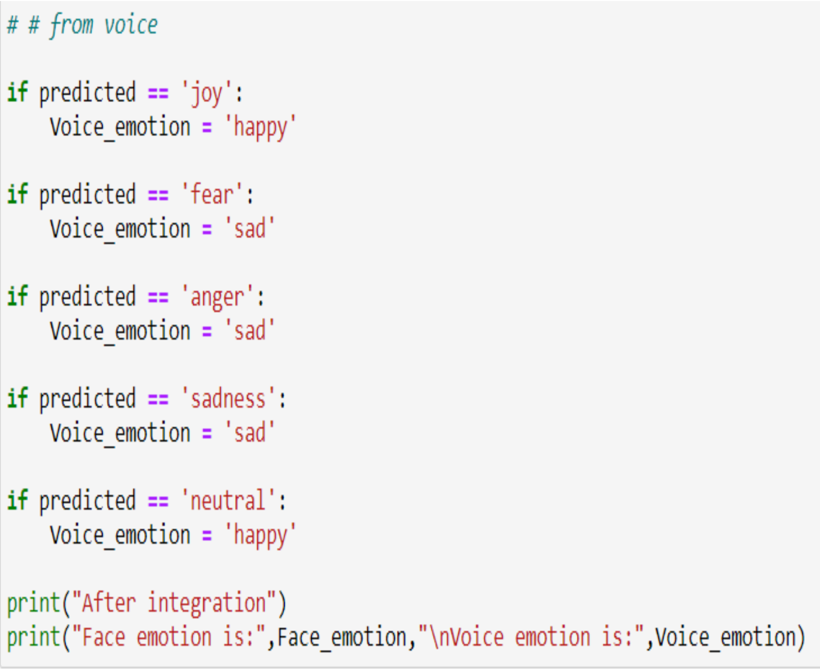


```
#import required modules

from pydub import AudioSegment
from pydub.playback import play

# for playing wav file

song = AudioSegment.from_wav("beep-06.wav")

from time import sleep

print('Provide Voice Signal After  a Beep sound')
sleep(2)
playsound('beep-06.wav')
```

- As the number of classes in the face and voice emotion are different, classify them into 2 primary classes- happy and sad.

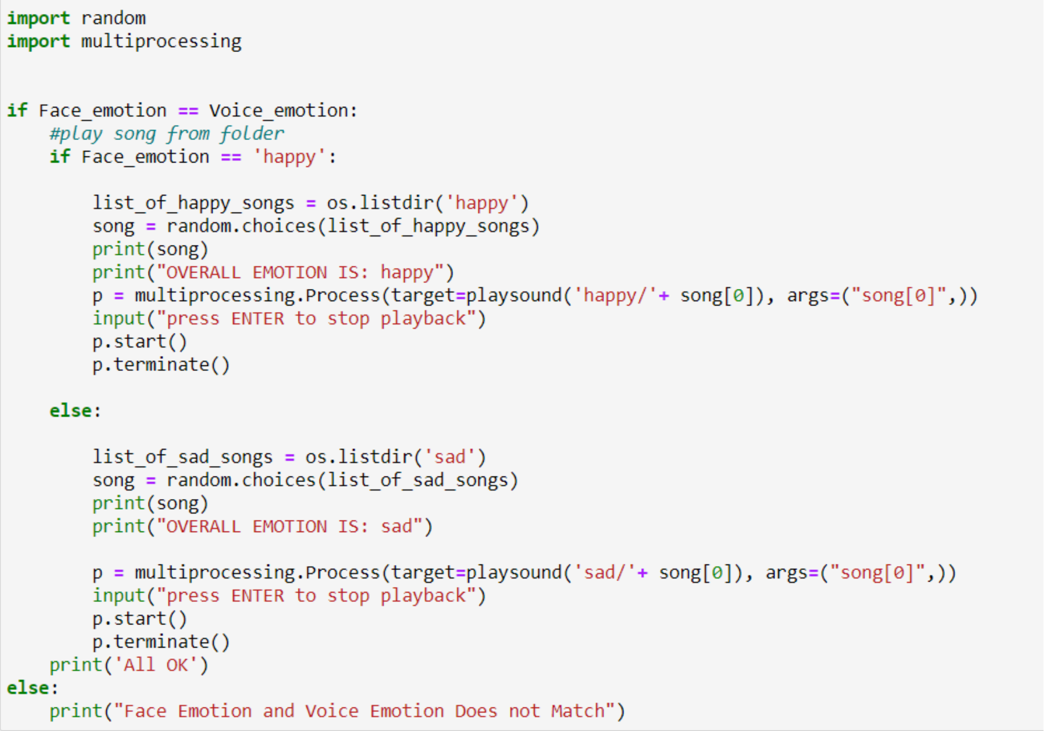


```
# # from face

if label == "angry":
    Face_emotion = "sad"

if label == "happy":
    Face_emotion = "happy"

if label == "sad":
    Face_emotion = "sad"

if label == "disgust":
    Face_emotion = "sad"

if label == "scared":
    Face_emotion = "sad"

if label == "surprised":
    Face_emotion = "happy"

if label == "neutral":
    Face_emotion = "happy"
```



```
# # from voice

if predicted == 'joy':
    Voice_emotion = 'happy'

if predicted == 'fear':
    Voice_emotion = 'sad'

if predicted == 'anger':
    Voice_emotion = 'sad'

if predicted == 'sadness':
    Voice_emotion = 'sad'

if predicted == 'neutral':
    Voice_emotion = 'happy'

print("After integration")
print("Face emotion is:",Face_emotion,"\nVoice emotion is:",Voice_emotion)
```

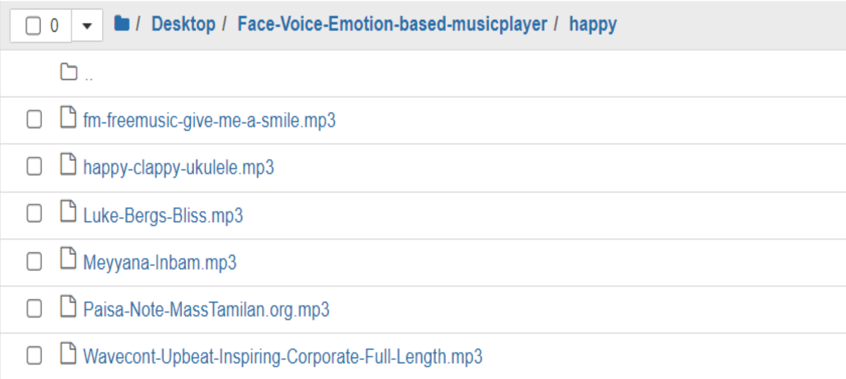
- If both the predicted emotions are same, then direct to the music player with the corresponding emotion.



```
import random
import multiprocessing


if Face_emotion == Voice_emotion:
    #play song from folder
    if Face_emotion == 'happy':

        list_of_happy_songs = os.listdir('happy')
        song = random.choices(list_of_happy_songs)
        print(song)
        print("OVERALL EMOTION IS: happy")
        p = multiprocessing.Process(target=playsound('happy/'+ song[0]), args=("song[0]",))
        input("press ENTER to stop playback")
        p.start()
        p.terminate()

    else:

        list_of_sad_songs = os.listdir('sad')
        song = random.choices(list_of_sad_songs)
        print(song)
        print("OVERALL EMOTION IS: sad")

        p = multiprocessing.Process(target=playsound('sad/'+ song[0]), args=("song[0]",))
        input("press ENTER to stop playback")
        p.start()
        p.terminate()
    print('All OK')
else:
    print("Face Emotion and Voice Emotion Does not Match")
```

- Categorizing playlists based on the emotions,happy and sad.

For example:



□ 0 ▾ / Desktop / Face-Voice-Emotion-based-musicplayer / happy

..

□ fm-freemusic-give-me-a-smile.mp3

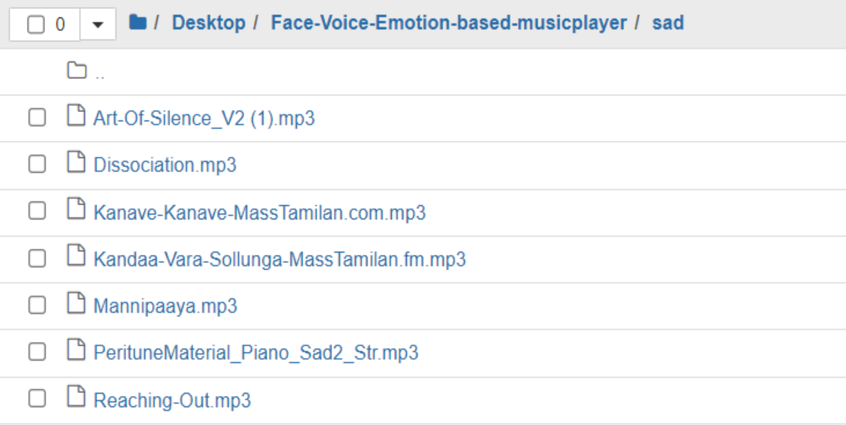
□ happy-clappy-ukulele.mp3

□ Luke-Bergs-Bliss.mp3

□ Meyyana-Inbam.mp3

□ Paisa-Note-MassTamilan.org.mp3

□ Wavecont-Upbeat-Inspiring-Corporate-Full-Length.mp3



□ 0 ▾ / Desktop / Face-Voice-Emotion-based-musicplayer / sad

..

□ Art-Of-Silence_V2 (1).mp3

□ Dissociation.mp3

□ Kanave-Kanave-MassTamilan.com.mp3

□ Kandaa-Vara-Sollunga-MassTamilan.fm.mp3

□ Mannipaaya.mp3

□ PerituneMaterial_Piano_Sad2_Str.mp3

□ Reaching-Out.mp3

9. SUPPORT INFORMATION

Experimental results have shown that the time required for audio feature extraction is negligible (around 0.0006 sec) and songs are stored pre-handed the total estimation time of the proposed system is proportional to the time required for extraction of facial features (around 0.9994 sec).Also the various classes of emotion yield a better accuracy rate as compared to previous existing systems. The computational time taken is 1.000sec which is very less thus helping in achieving a better real time performance and efficiency.
The system thus aims at providing the Windows operating system users with a cheaper, additional hardware free and accurate emotion based music system. The Emotion Based Music System will be of great advantage to users looking for music based on their mood and emotional behavior. It will help reduce the searching time for music thereby reducing the unnecessary computational time and thereby increasing the overall accuracy and efficiency of the system. The system will not only reduce physical stress but will also act as a boon for the music therapy systems and may also assist the music therapist to therapize a patient. Also with its additional features mentioned above, it will be a complete system for music lovers and listeners.
The comparison of the average accuracy of the proposed system for the all classification of emotions to the existing system:

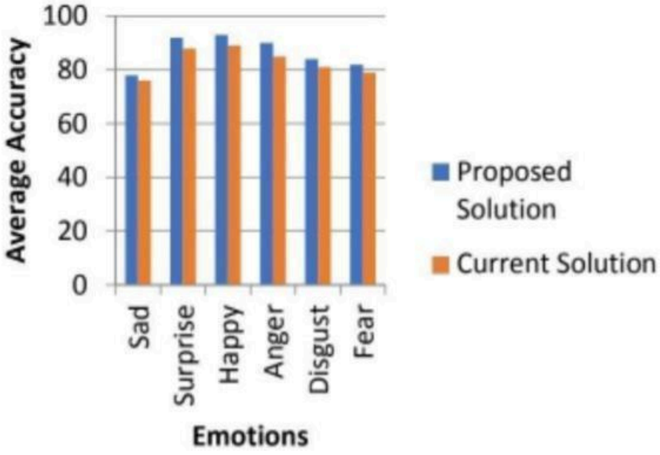


Fig.3 Comparison of the proposed system with the existing system

Few Related links

- https://www.raspberrypi.org/education/
- https://www.pyimagesearch.com/2018/09/10/ Keras-tutorial-how-to-get-started-with-keras-deep-learning-and- python
- https://towardsdatascience.com/face Face-detection-recognition-and-emotion-detection-in-8-lines-of-code- b2ce32d4d5de

10. SURVEYED CONTENT

Due to the lack of emotion-labeled datasets, many supervised classifications for emotions have been done on data gathered from microblogs (e.g. Twitter), using hashtags or emoticons as the emotional label for the data, under the assumption that these signals show the emotional state of the writer.

Purver and Battersby (2012) on Twitter data using SVM classifier reached 82% accuracy for classify- ing the emotion Happy in 10-fold cross validation, and 67% in classifying over the entire dataset for the same emotion, with emoticons as labels for the training set, and hashtags as labels for the test set. Then they tested their trained models for each emotion to see if they can distinguish emotion classes from each other rather than just distinguish one class from a general Other set. The results varied from 13% to 76% accuracy for different emotions.

Hasan et al. (2014) also used hashtags as their labels and created their features using the unigram model, removing any word from tweets which were not in their emotion lexicon (created using 28 basic emotion words in Circumplex model and extended with WordNet synsets). Four classifiers (Naive Bayes, SVM, Decision Tree, and KNN) achieved accuracies close to 90% in classifying four main classes of emotion categories in the Circumplex model.

Li and Xu (2014) proposed a ”emotion cause detection technique” to extract features that are ”mean- ingful” to emotions instead of choosing words with high co-occurrence degree. Their method is based on Lee et al.'s work on rule based emotion cause detection (Lee et al., 2010). After using predefined linguistic patterns to extract emotion causes and adding it to their features, they used Support Vector Regression (SVR) to create the classifier, and reached a higher F-score for some emotions like happiness, anger, and disgust compared to previous works.

Bandhakavi et al. (2017b) used domain-specific lexicon that they created based on unigram mixture models (Bandhakavi et al., 2014; Bandhakavi et al., 2017a) to extract features and showed that their lexi- con outperform methods like Pointwise Mutual Information, and supervised Latent Dirichlet Allocation.

A few related Links:

- Caltech Faces (2020h- http://www.vision.caltech.edu/html-files/archive.html Accessed 05 Jan 2020
- The CMU multi-pie face detection (2020)- http://ww1.multipie.org/. Accessed 05 Jan 2020
- NIST mugshot identification database (2020)- https://www.nist.gov/itl/iad/image-group/resources/biometric-special-databases-and-software ,Accessed 05 Jan 2020
- https://youtu.be/3G9M7skYUHk
- https://youtu.be/3G9M7skYUHk

11. RESULTS AND COMPARISON:

11.1 RESULTS

★ In the 1st module, we intended to detect the emotion from facial expressions of the human face which is projected in the device camera.

We show the capturing of facial emotion detection through series of figures that we used to test and work our system,as given below:

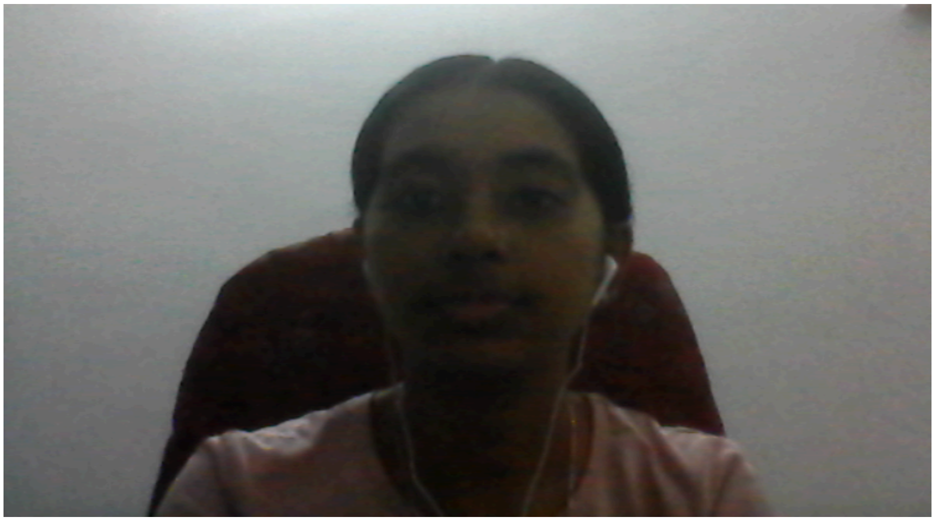


Fig.4-The input image to the device camera for emotion detection.

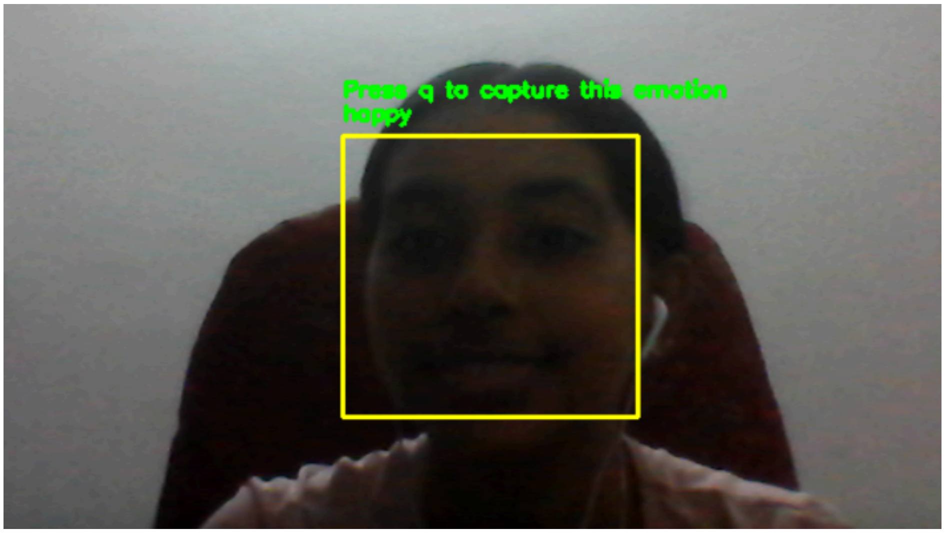
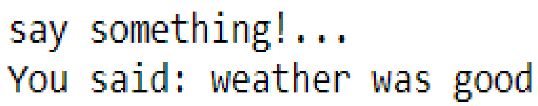


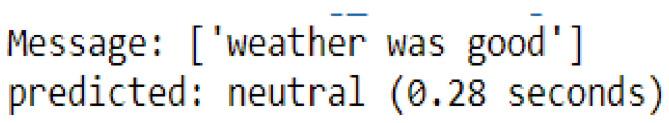
Fig.5- Once ‘q’ is pressed, the emotion is captured and the captured emotion in this case is - happy.

- ★ In the 2nd module we intend to detect the emotion from the person's voice by converting the voice to text using Google API and extract the emotion from this converted text.



```
say something!...
You said: weather was good
```

Fig.6- The text recognised from the person's voice is - weather was good.

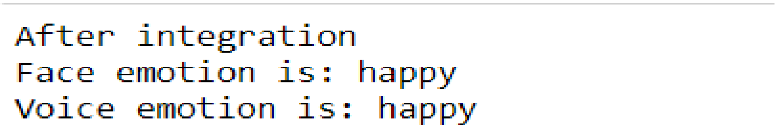


```
Message: ['weather was good']
predicted: neutral (0.28 seconds)
```

Fig.7- The predicted emotion for the given input text (voice) is neutral.

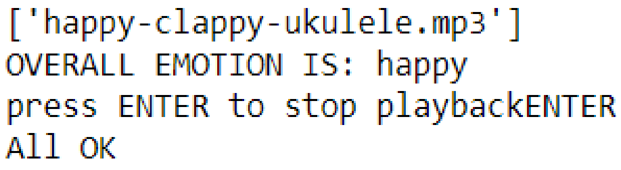
- ★ In the 3rd module we integrate the 2 emotions and direct to the music player only if both the emotions match.

Final Output:



```
After integration
Face emotion is: happy
Voice emotion is: happy
```

Fig.8- Classify the 7 classes into 2 primary classes - happy and sad.As the speech emotion was neutral it is converted into happy.



```
['happy-clappy-ukulele.mp3']
OVERALL EMOTION IS: happy
press ENTER to stop playbackENTER
All OK
```

Fig.9- As both the emotions are same, a song (randomly) from the list of happy songs directory is played.

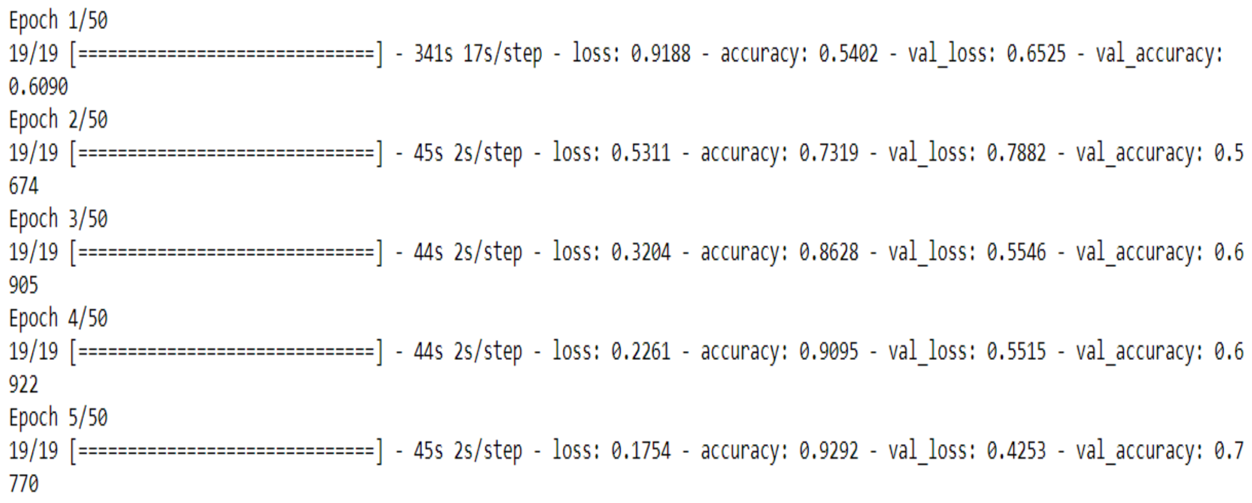
11.2 ALTERNATE MODELS -COMPARISON AND ANALYSIS

We have proposed alternate models for Facial emotion and Speech emotion detection. Starting with **facial emotion detection model-Alternate model:**

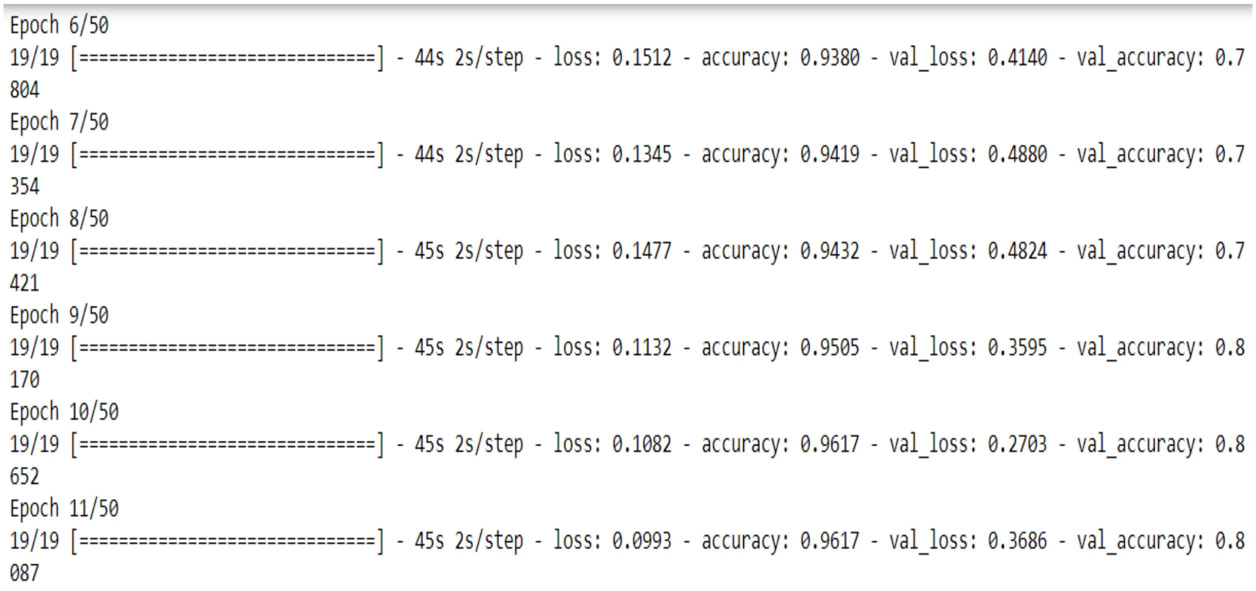
1.The 2 models used are mini_XCEPTION and tiny_XCEPTION (Alternate model)

This is a Convolution Neural network (CNN) based model with a base layer and 4 additional layers. The base layer consists of 2 2D CNN layers whereas the additional layers consist of 1 2D CNN layer and 2 Separable 2D layers. The changes incorporated in the models below are the changes in the parameters used such as kernel size, striding size, Pooling size etc. The activation function used after every layer is relu and the final activation of the fully connected dense layer is softmax. All the layers have padding=same i.e. are padded with 0 to have the inputs of the same size.

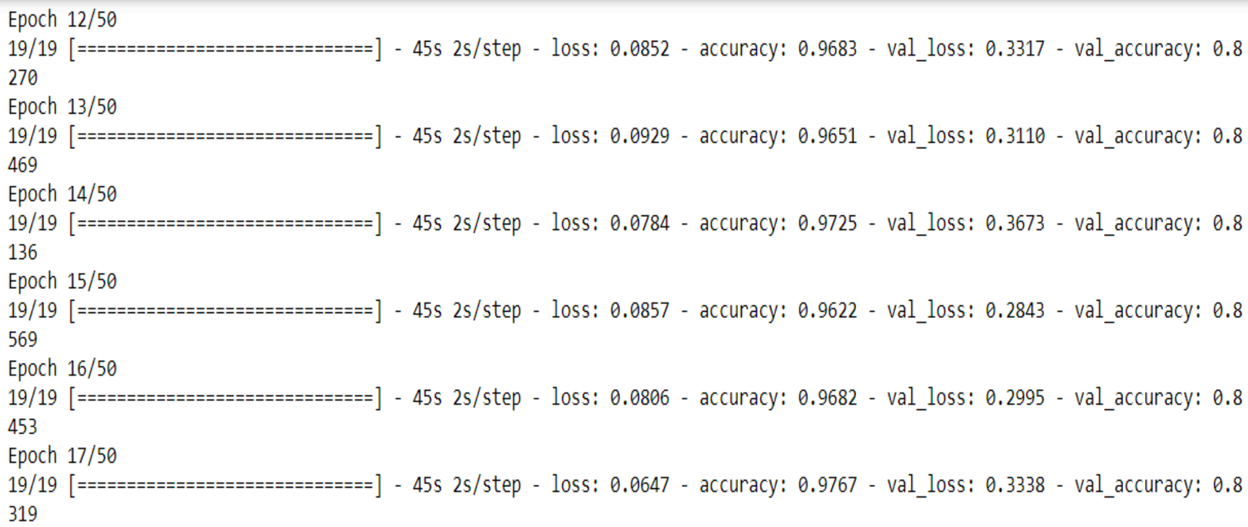
mini_XCEPTION model-



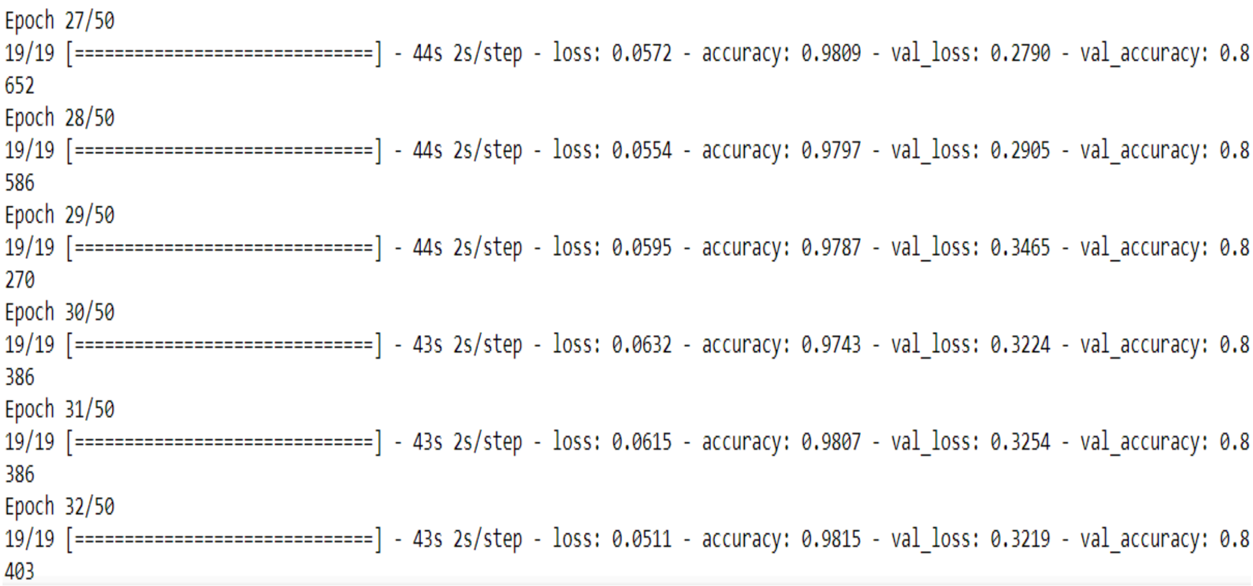
```
Epoch 1/50
19/19 [==============================] - 341s 17s/step - loss: 0.9188 - accuracy: 0.5402 - val_loss: 0.6525 - val_accuracy:
0.6090
Epoch 2/50
19/19 [==============================] - 45s 2s/step - loss: 0.5311 - accuracy: 0.7319 - val_loss: 0.7882 - val_accuracy: 0.5
674
Epoch 3/50
19/19 [==============================] - 44s 2s/step - loss: 0.3204 - accuracy: 0.8628 - val_loss: 0.5546 - val_accuracy: 0.6
905
Epoch 4/50
19/19 [==============================] - 44s 2s/step - loss: 0.2261 - accuracy: 0.9095 - val_loss: 0.5515 - val_accuracy: 0.6
922
Epoch 5/50
19/19 [==============================] - 45s 2s/step - loss: 0.1754 - accuracy: 0.9292 - val_loss: 0.4253 - val_accuracy: 0.7
770
```



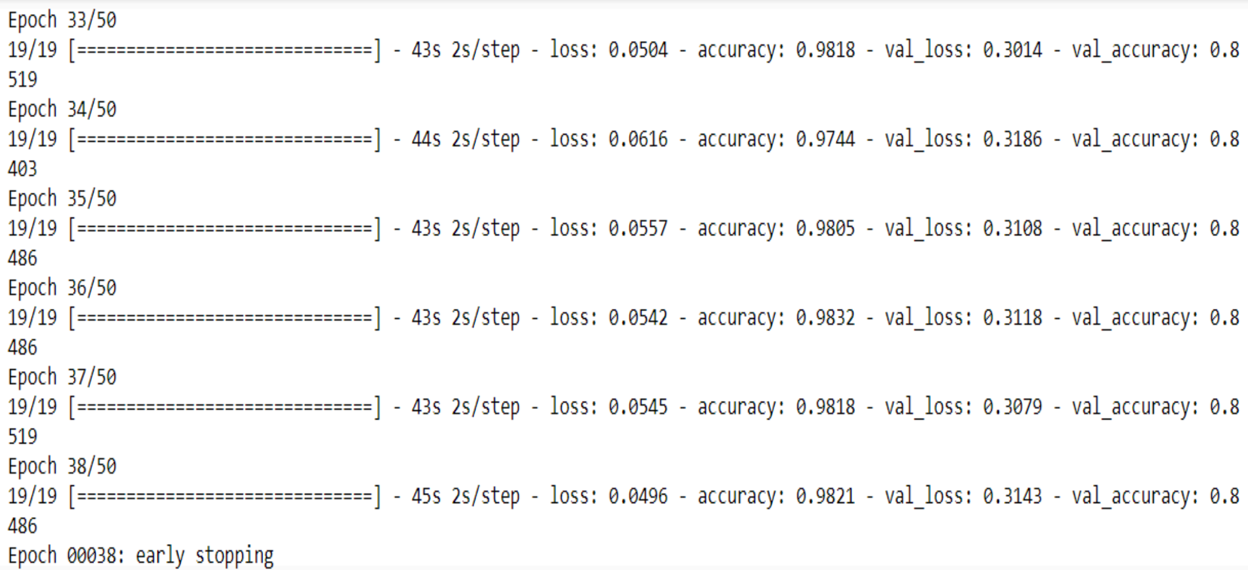
```
Epoch 6/50
19/19 [==============================] - 44s 2s/step - loss: 0.1512 - accuracy: 0.9380 - val_loss: 0.4140 - val_accuracy: 0.7
804
Epoch 7/50
19/19 [==============================] - 44s 2s/step - loss: 0.1345 - accuracy: 0.9419 - val_loss: 0.4880 - val_accuracy: 0.7
354
Epoch 8/50
19/19 [==============================] - 45s 2s/step - loss: 0.1477 - accuracy: 0.9432 - val_loss: 0.4824 - val_accuracy: 0.7
421
Epoch 9/50
19/19 [==============================] - 45s 2s/step - loss: 0.1132 - accuracy: 0.9505 - val_loss: 0.3595 - val_accuracy: 0.8
170
Epoch 10/50
19/19 [==============================] - 45s 2s/step - loss: 0.1082 - accuracy: 0.9617 - val_loss: 0.2703 - val_accuracy: 0.8
652
Epoch 11/50
19/19 [==============================] - 45s 2s/step - loss: 0.0993 - accuracy: 0.9617 - val_loss: 0.3686 - val_accuracy: 0.8
087
```



```
Epoch 12/50
19/19 [==============================] - 45s 2s/step - loss: 0.0852 - accuracy: 0.9683 - val_loss: 0.3317 - val_accuracy: 0.8
270
Epoch 13/50
19/19 [==============================] - 45s 2s/step - loss: 0.0929 - accuracy: 0.9651 - val_loss: 0.3110 - val_accuracy: 0.8
469
Epoch 14/50
19/19 [==============================] - 45s 2s/step - loss: 0.0784 - accuracy: 0.9725 - val_loss: 0.3673 - val_accuracy: 0.8
136
Epoch 15/50
19/19 [==============================] - 45s 2s/step - loss: 0.0857 - accuracy: 0.9622 - val_loss: 0.2843 - val_accuracy: 0.8
569
Epoch 16/50
19/19 [==============================] - 45s 2s/step - loss: 0.0806 - accuracy: 0.9682 - val_loss: 0.2995 - val_accuracy: 0.8
453
Epoch 17/50
19/19 [==============================] - 45s 2s/step - loss: 0.0647 - accuracy: 0.9767 - val_loss: 0.3338 - val_accuracy: 0.8
319
```

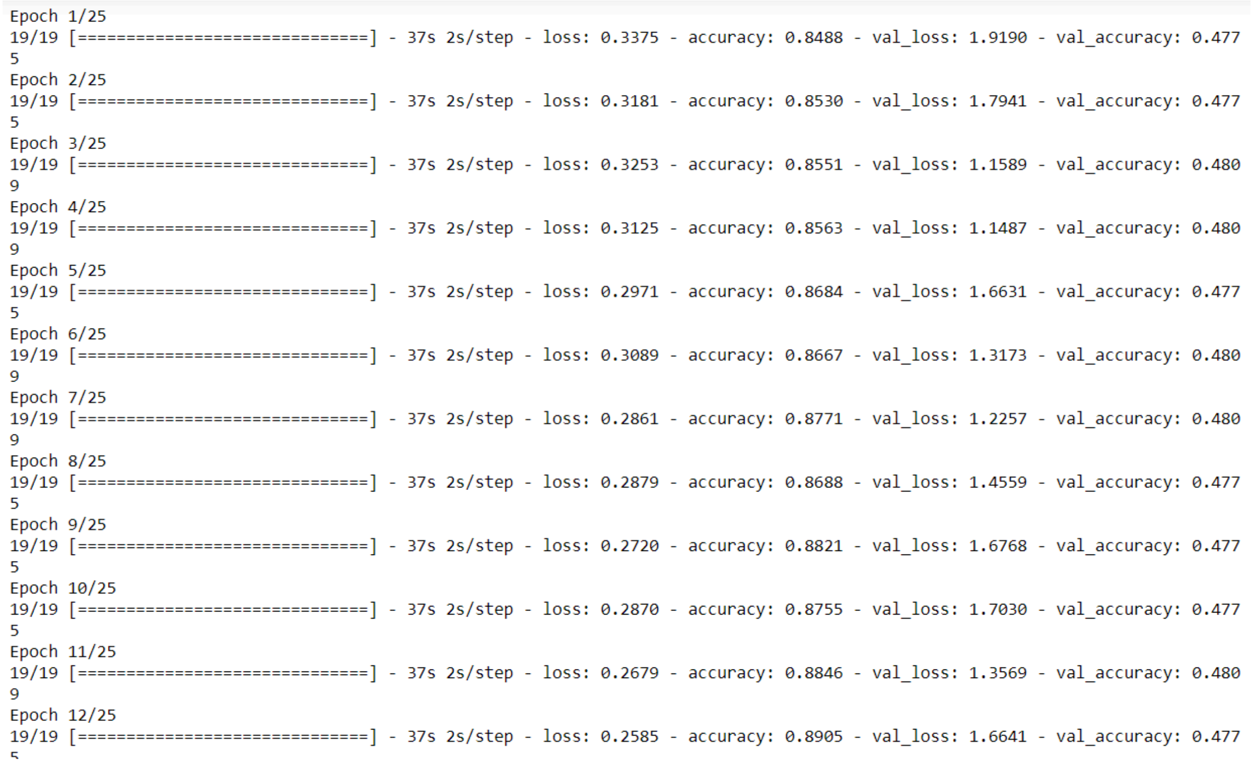


```
Epoch 27/50
19/19 [==============================] - 44s 2s/step - loss: 0.0572 - accuracy: 0.9809 - val_loss: 0.2790 - val_accuracy: 0.8
652
Epoch 28/50
19/19 [==============================] - 44s 2s/step - loss: 0.0554 - accuracy: 0.9797 - val_loss: 0.2905 - val_accuracy: 0.8
586
Epoch 29/50
19/19 [==============================] - 44s 2s/step - loss: 0.0595 - accuracy: 0.9787 - val_loss: 0.3465 - val_accuracy: 0.8
270
Epoch 30/50
19/19 [==============================] - 43s 2s/step - loss: 0.0632 - accuracy: 0.9743 - val_loss: 0.3224 - val_accuracy: 0.8
386
Epoch 31/50
19/19 [==============================] - 43s 2s/step - loss: 0.0615 - accuracy: 0.9807 - val_loss: 0.3254 - val_accuracy: 0.8
386
Epoch 32/50
19/19 [==============================] - 43s 2s/step - loss: 0.0511 - accuracy: 0.9815 - val_loss: 0.3219 - val_accuracy: 0.8
403
```

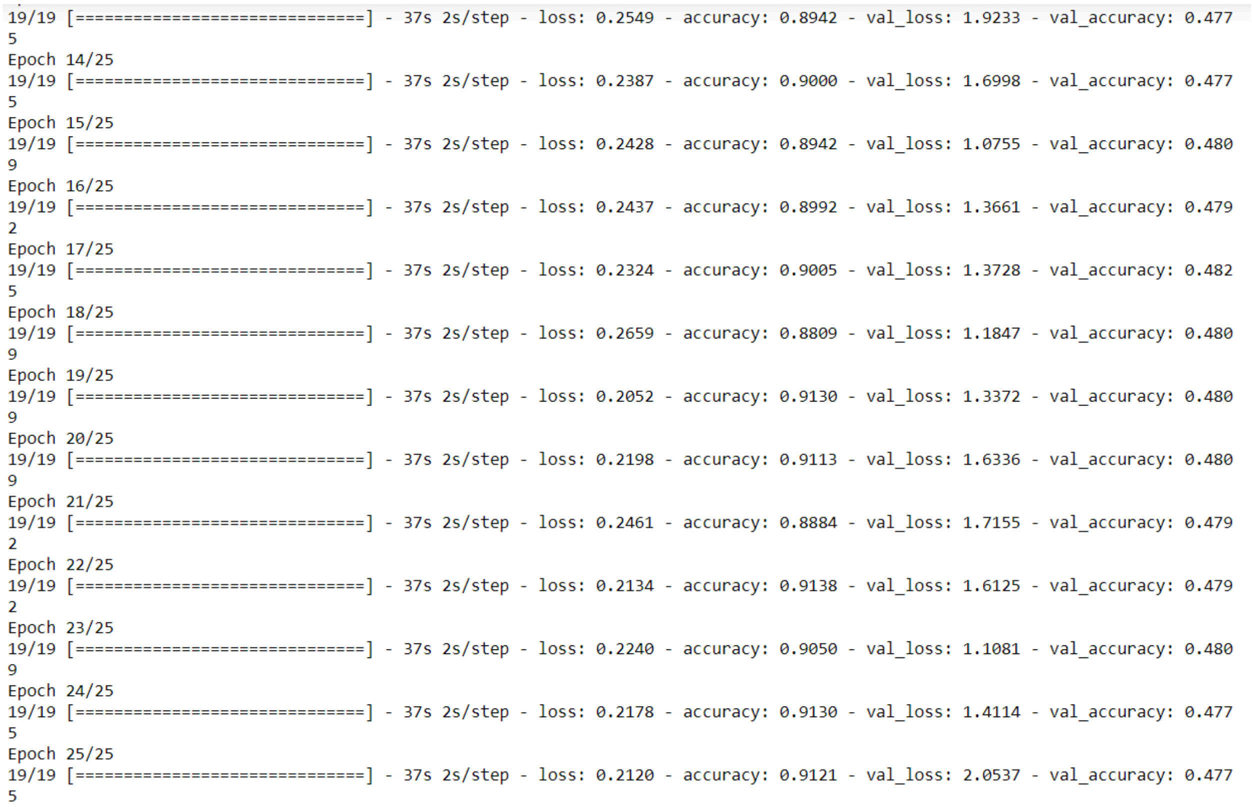


```
Epoch 33/50
19/19 [==============================] - 43s 2s/step - loss: 0.0504 - accuracy: 0.9818 - val_loss: 0.3014 - val_accuracy: 0.8
519
Epoch 34/50
19/19 [==============================] - 44s 2s/step - loss: 0.0616 - accuracy: 0.9744 - val_loss: 0.3186 - val_accuracy: 0.8
403
Epoch 35/50
19/19 [==============================] - 43s 2s/step - loss: 0.0557 - accuracy: 0.9805 - val_loss: 0.3108 - val_accuracy: 0.8
486
Epoch 36/50
19/19 [==============================] - 43s 2s/step - loss: 0.0542 - accuracy: 0.9832 - val_loss: 0.3118 - val_accuracy: 0.8
486
Epoch 37/50
19/19 [==============================] - 43s 2s/step - loss: 0.0545 - accuracy: 0.9818 - val_loss: 0.3079 - val_accuracy: 0.8
519
Epoch 38/50
19/19 [==============================] - 45s 2s/step - loss: 0.0496 - accuracy: 0.9821 - val_loss: 0.3143 - val_accuracy: 0.8
486
Epoch 00038: early stopping
```

tiny_XCEPTION model-

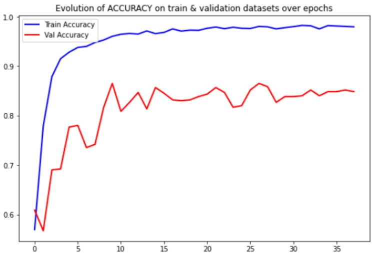
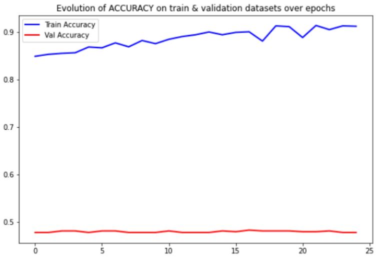
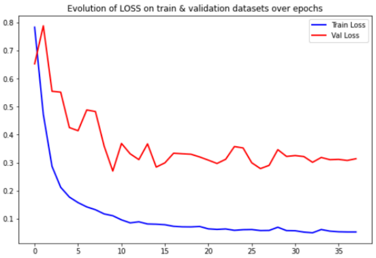
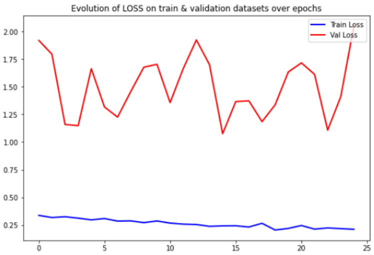
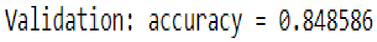
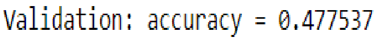
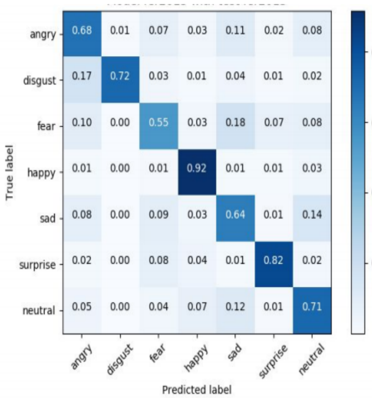
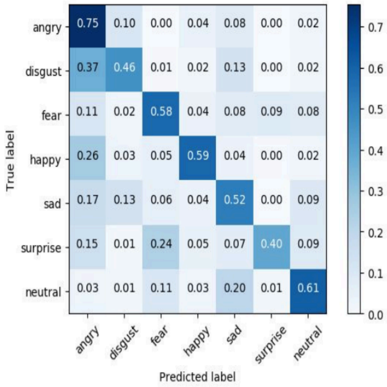


```
Epoch 1/25
19/19 [==============================] - 37s 2s/step - loss: 0.3375 - accuracy: 0.8488 - val_loss: 1.9190 - val_accuracy: 0.477
5
Epoch 2/25
19/19 [==============================] - 37s 2s/step - loss: 0.3181 - accuracy: 0.8530 - val_loss: 1.7941 - val_accuracy: 0.477
5
Epoch 3/25
19/19 [==============================] - 37s 2s/step - loss: 0.3253 - accuracy: 0.8551 - val_loss: 1.1589 - val_accuracy: 0.480
9
Epoch 4/25
19/19 [==============================] - 37s 2s/step - loss: 0.3125 - accuracy: 0.8563 - val_loss: 1.1487 - val_accuracy: 0.480
9
Epoch 5/25
19/19 [==============================] - 37s 2s/step - loss: 0.2971 - accuracy: 0.8684 - val_loss: 1.6631 - val_accuracy: 0.477
5
Epoch 6/25
19/19 [==============================] - 37s 2s/step - loss: 0.3089 - accuracy: 0.8667 - val_loss: 1.3173 - val_accuracy: 0.480
9
Epoch 7/25
19/19 [==============================] - 37s 2s/step - loss: 0.2861 - accuracy: 0.8771 - val_loss: 1.2257 - val_accuracy: 0.480
9
Epoch 8/25
19/19 [==============================] - 37s 2s/step - loss: 0.2879 - accuracy: 0.8688 - val_loss: 1.4559 - val_accuracy: 0.477
5
Epoch 9/25
19/19 [==============================] - 37s 2s/step - loss: 0.2720 - accuracy: 0.8821 - val_loss: 1.6768 - val_accuracy: 0.477
5
Epoch 10/25
19/19 [==============================] - 37s 2s/step - loss: 0.2870 - accuracy: 0.8755 - val_loss: 1.7030 - val_accuracy: 0.477
5
Epoch 11/25
19/19 [==============================] - 37s 2s/step - loss: 0.2679 - accuracy: 0.8846 - val_loss: 1.3569 - val_accuracy: 0.480
9
Epoch 12/25
19/19 [==============================] - 37s 2s/step - loss: 0.2585 - accuracy: 0.8905 - val_loss: 1.6641 - val_accuracy: 0.477
5
```



```
19/19 [==============================] - 37s 2s/step - loss: 0.2549 - accuracy: 0.8942 - val_loss: 1.9233 - val_accuracy: 0.477
5
Epoch 14/25
19/19 [==============================] - 37s 2s/step - loss: 0.2387 - accuracy: 0.9000 - val_loss: 1.6998 - val_accuracy: 0.477
5
Epoch 15/25
19/19 [==============================] - 37s 2s/step - loss: 0.2428 - accuracy: 0.8942 - val_loss: 1.0755 - val_accuracy: 0.480
9
Epoch 16/25
19/19 [==============================] - 37s 2s/step - loss: 0.2437 - accuracy: 0.8992 - val_loss: 1.3661 - val_accuracy: 0.479
2
Epoch 17/25
19/19 [==============================] - 37s 2s/step - loss: 0.2324 - accuracy: 0.9005 - val_loss: 1.3728 - val_accuracy: 0.482
5
Epoch 18/25
19/19 [==============================] - 37s 2s/step - loss: 0.2659 - accuracy: 0.8809 - val_loss: 1.1847 - val_accuracy: 0.480
9
Epoch 19/25
19/19 [==============================] - 37s 2s/step - loss: 0.2052 - accuracy: 0.9130 - val_loss: 1.3372 - val_accuracy: 0.480
9
Epoch 20/25
19/19 [==============================] - 37s 2s/step - loss: 0.2198 - accuracy: 0.9113 - val_loss: 1.6336 - val_accuracy: 0.480
9
Epoch 21/25
19/19 [==============================] - 37s 2s/step - loss: 0.2461 - accuracy: 0.8884 - val_loss: 1.7155 - val_accuracy: 0.479
2
Epoch 22/25
19/19 [==============================] - 37s 2s/step - loss: 0.2134 - accuracy: 0.9138 - val_loss: 1.6125 - val_accuracy: 0.479
2
Epoch 23/25
19/19 [==============================] - 37s 2s/step - loss: 0.2240 - accuracy: 0.9050 - val_loss: 1.1081 - val_accuracy: 0.480
9
Epoch 24/25
19/19 [==============================] - 37s 2s/step - loss: 0.2178 - accuracy: 0.9130 - val_loss: 1.4114 - val_accuracy: 0.477
5
Epoch 25/25
19/19 [==============================] - 37s 2s/step - loss: 0.2120 - accuracy: 0.9121 - val_loss: 2.0537 - val_accuracy: 0.477
5
```

Table.1- Comparison of the face emotion models

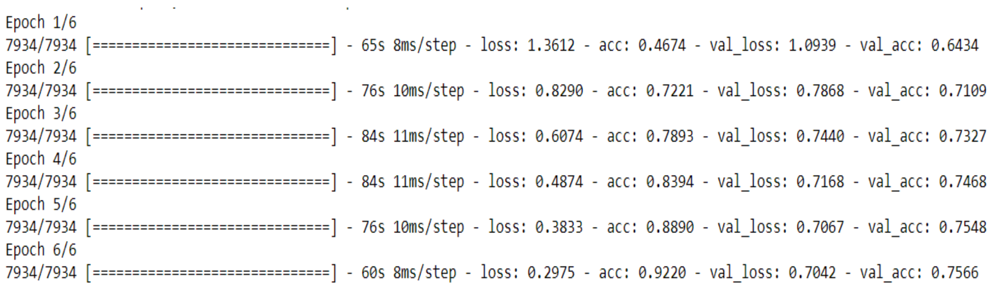
EVALUATION METRICS	MODEL 1 (mini_XCEPTION)	MODEL 2 (tiny_XCEPTION)
ACCURACY PLOT		
LOSS PLOT		
ACCURACY	 Validation: accuracy = 0.848586	 Validation: accuracy = 0.477537
CONFUSION MATRIX		

ANALYSIS

From the tabulations, we find that the mini_XCEPTION model has shown significant accuracy of about **85%** compared to the tiny_XCEPTION model with an accuracy of **48%**.

Result

Hence, we choose the mini_XCEPTION model for our emotion detection from face.



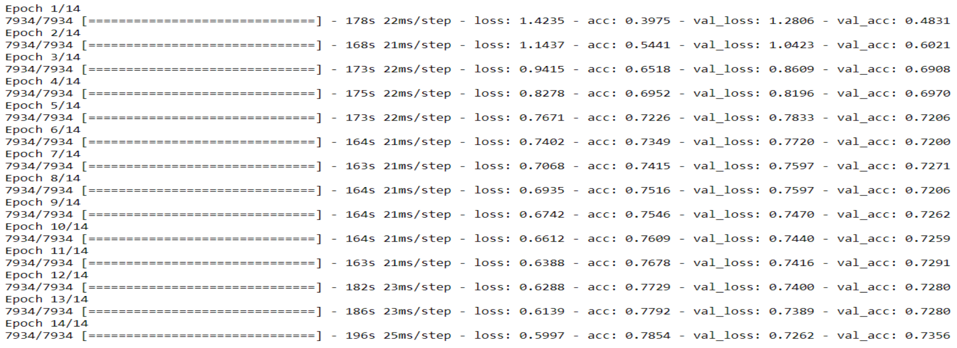
```
Epoch 1/6
7934/7934 [==============================] - 65s 8ms/step - loss: 1.3612 - acc: 0.4674 - val_loss: 1.0939 - val_acc: 0.6434
Epoch 2/6
7934/7934 [==============================] - 76s 10ms/step - loss: 0.8290 - acc: 0.7221 - val_loss: 0.7868 - val_acc: 0.7109
Epoch 3/6
7934/7934 [==============================] - 84s 11ms/step - loss: 0.6074 - acc: 0.7893 - val_loss: 0.7440 - val_acc: 0.7327
Epoch 4/6
7934/7934 [==============================] - 84s 11ms/step - loss: 0.4874 - acc: 0.8394 - val_loss: 0.7168 - val_acc: 0.7468
Epoch 5/6
7934/7934 [==============================] - 76s 10ms/step - loss: 0.3833 - acc: 0.8890 - val_loss: 0.7067 - val_acc: 0.7548
Epoch 6/6
7934/7934 [==============================] - 60s 8ms/step - loss: 0.2975 - acc: 0.9220 - val_loss: 0.7042 - val_acc: 0.7566
```

Speech emotion detection-Alternate models

1.The 2 models used are CNN pipeline and GRU pipeline (both for emotion detection from text with the input embedded layer)

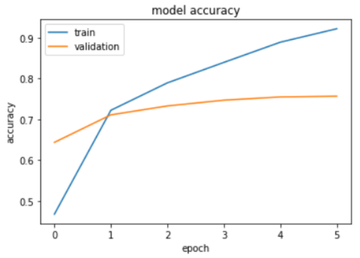
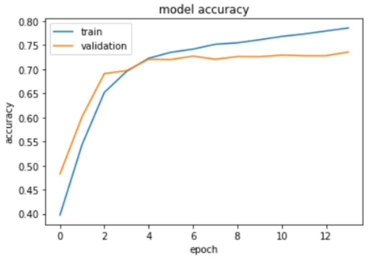
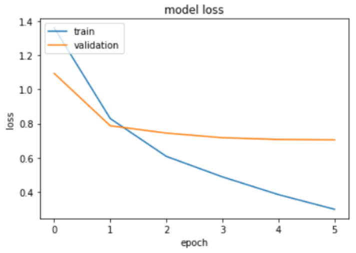
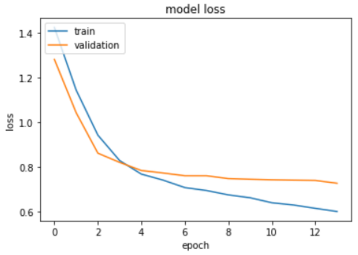
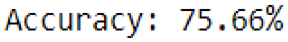
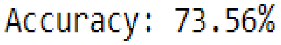
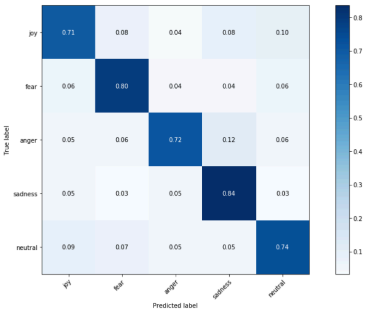
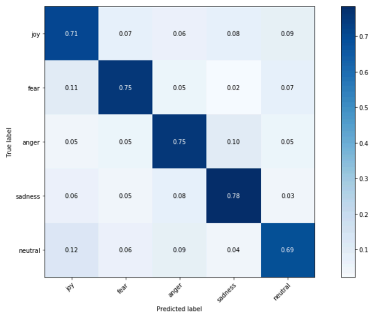
1. We propose a deep learning model for emotion detection from the text sentences from the user speech. The model consists of three Convolutional Neural Networks (CNNs). Each CNN contains a convolutional layer and a max-pooling layer, followed by a fully-connected layer for classifying the sentences into respective emotions. The model employs the word vector representation as textual features, which works on random initialization for the word vectors, and are set to be trainable and updated through the model training phase. Eventually, task-specific vectors are generated as the model learns to distinguish the meaning of words in the dataset.

2. We propose an alternate emotion detection in this model,Bi-directional Gate Recurrent Unit (Bi-GRU) network to capture the emotion vectors for the aspect of input words, and second, we statistically analyze the emoticon that appears in our data set to obtain the emotion distribution, then, use the emoticon distribution to enhance the emotion vectors.



```
Epoch 1/14
7934/7934 [==============================] - 178s 22ms/step - loss: 1.4235 - acc: 0.3975 - val_loss: 1.2806 - val_acc: 0.4831
Epoch 2/14
7934/7934 [==============================] - 168s 21ms/step - loss: 1.1437 - acc: 0.5441 - val_loss: 1.0423 - val_acc: 0.6021
Epoch 3/14
7934/7934 [==============================] - 173s 22ms/step - loss: 0.9415 - acc: 0.6518 - val_loss: 0.8609 - val_acc: 0.6908
Epoch 4/14
7934/7934 [==============================] - 175s 22ms/step - loss: 0.8278 - acc: 0.6952 - val_loss: 0.8196 - val_acc: 0.6970
Epoch 5/14
7934/7934 [==============================] - 173s 22ms/step - loss: 0.7671 - acc: 0.7226 - val_loss: 0.7833 - val_acc: 0.7206
Epoch 6/14
7934/7934 [==============================] - 164s 21ms/step - loss: 0.7402 - acc: 0.7349 - val_loss: 0.7720 - val_acc: 0.7200
Epoch 7/14
7934/7934 [==============================] - 163s 21ms/step - loss: 0.7068 - acc: 0.7415 - val_loss: 0.7597 - val_acc: 0.7271
Epoch 8/14
7934/7934 [==============================] - 164s 21ms/step - loss: 0.6935 - acc: 0.7516 - val_loss: 0.7597 - val_acc: 0.7206
Epoch 9/14
7934/7934 [==============================] - 164s 21ms/step - loss: 0.6742 - acc: 0.7546 - val_loss: 0.7470 - val_acc: 0.7262
Epoch 10/14
7934/7934 [==============================] - 164s 21ms/step - loss: 0.6612 - acc: 0.7609 - val_loss: 0.7440 - val_acc: 0.7259
Epoch 11/14
7934/7934 [==============================] - 163s 21ms/step - loss: 0.6388 - acc: 0.7678 - val_loss: 0.7416 - val_acc: 0.7291
Epoch 12/14
7934/7934 [==============================] - 182s 23ms/step - loss: 0.6288 - acc: 0.7729 - val_loss: 0.7400 - val_acc: 0.7280
Epoch 13/14
7934/7934 [==============================] - 186s 23ms/step - loss: 0.6139 - acc: 0.7792 - val_loss: 0.7389 - val_acc: 0.7280
Epoch 14/14
7934/7934 [==============================] - 196s 25ms/step - loss: 0.5997 - acc: 0.7854 - val_loss: 0.7262 - val_acc: 0.7356
```

Table.2- Comparison of the text emotion models

EVALUATION METRICS	MODEL 1 (CNN)	MODEL 2 (GRU)
ACCURACY PLOT		
LOSS PLOT		
ACCURACY	 Accuracy: 75.66%	 Accuracy: 73.56%
CONFUSION MATRIX		

ANALYSIS

From the tabulations, we find that the CNN model has shown significant accuracy of about 75.66% compared to the GRU model with an accuracy of 73.56% with a lesser number of epochs. If we train the CNN model with more number of epochs then the accuracy will significantly increase.

Result

Hence, we choose the CNN model for our emotion detection from text.

12. FUTURE SCOPE

One of the future scope of the system would be enhancing the proposed system for efficient accuracy and working,and implementation in all upcoming devices.The proposed system also tends to avoid in future the unpredictable results produced in extreme bad light conditions and very poor camera resolution.Another future scope would be to design a mechanism that would be helpful in music therapy treatment and provide the music therapist the help needed to treat the patients suffering from disorders like mental stress, anxiety, acute depression and trauma.
The multi-sensory emotion recognition framework proposed in this dissertation can be extended to many other research topics in the field of human computer interaction. We hope our research can trigger more investigations in the area of human computer interaction to make computers more friendly, natural, and human-like in the near future.In the aspects of business streams, emotion recognition will be of greater use. In general, recognizing the face is used to authenticate the user. The software would find use also in the android stream. While the user is driving, it can be used to find if the user is sleepy. Physically challenged people can best use this to hear songs. The music player can act as an alert system for driving. It can be used as a Social robot emotion recognition system. It can be used for the Medical practices. It can be used as a feedback system for e-learning. The proposed music player can be used in the Automatic counseling system. It can be used in face expression synthesis. The music player can be used in research related to psychology. It is very much useful in human behavior in interviews.

13. CONCLUSION

In the above proposed paper, we have developed a model that plays a song based on the user's emotions,detected through facial expressions and the speech recognition of users. The fundamental purpose of the system is to identify the emotional state of the user and match personalized music preferences based on current emotions and presenting relevant music tracks.The Emotion-Based Music Player saves time and the effort of manual playlist creation by browsing and playing them.It is used to highly facilitate the use by physically challenged people to automate and give them better music player experience. The application solves the basic needs of music listeners without troubling them as most of today’s existing applications do.

14. REFERENCES

1. *Ninad Mehendale, "Facial emotion recognition using convolutional neural networks (FERC)" Springer Nature Switzerland AG 2020*
2. *S. Deebika, K. A. Indira, Dr. Jesline "A Machine Learning Based Music Player by Detecting Emotions" 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), IEEE*
3. *Xusheng Wang , Xing Chen, Congjun Cao "Human emotion recognition by optimally fusing facial expression and speech feature" Signal Processing: Image Communication Volume 84, May 2020, 115831, Elsevier*
4. *Saeed Turabzadeh , Hongying Meng , Rafiq M. Swash , Matus Pleva Jozef Juhar "Facial Expression Emotion Detection for Real-Time Embedded Systems" Technologies 2018, 6, 17; doi:10.3390/technologies6010017.*
5. *Maruthi Raja S K, Kumaran V, Keerthi Vasan A , Kavitha N "Real Time Intelligent Emotional Music Player using Android" Journal for Research | Volume 03| Issue 01 | March 2017.*
6. *Eduard FRANT, Ioan ISPAS , Voichita DRAGOMIR , Monica DASCALU, Elteto ZOLTAN , Ioan Cristian STOICA " Voice Based Emotion Recognition with Convolutional Neural Networks for Companion Robots" ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY Volume 20, Number 3, 2017.*
7. *Karan Mistry, Prince Pathak , Prof. Suvarna Aranjo " Mood based Music Player" International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 3 | Mar -2017.*
8. *Sathya Bursic , Giuseppe Boccignone , Alfio Ferrara , Alessandro D'Amelio , Raffaella Lanzarotti " Improving the Accuracy of Automatic Facial Expression Recognition in Speaking Subjects with Deep Learning" Appl. Sci. 2020, 10, 4002; doi:10.3390/app10114002*
9. *Qingmei Yao " Multi-Sensory Emotion Recognition with Speech and Facial Expression" The University of Southern Mississippi The Aquila Digital Community.*
10. *Vladimir Chernykh, Pavel Prikhodko "Emotion Recognition From Speech With Recurrent Neural Networks" arXiv:1701.08071v2 [cs.CL] 5 Jul 2018.*
11. *Renuka R. Londhe, Dr. Vrushshen P. Pawar, "Analysis of Facial Expression and Recognition Based On Statistical Approach", International Journal of Soft Computing and Engineering (IJSCE) Volume-2, May 2012.*
12. *Z. Zeng "A survey of affect recognition methods: Audio, visual, and spontaneous expressions",IEEE. Transaction Pattern Analysis, vol 31, January 2009*
13. *Sri Charan Nimmagadda "Emotion based music player app" ,Report-2017.*
14. *Mutasem K. Alsmadi "Patent-Facial emotion recognition",IJER,February 2015.*
15. *Hafeez Kabani, Sharik Khan , Omar Khan, Shabana Tadvi "Smart player" University of Washington,2012*

15. APPENDIX A

The undersigned acknowledge they have completed implementing the project “Emotion based music player ” and agree with the approach it presents.

Signature	Date	Name	Roll number
Aparna SS	24.05.2021	Aparna S S	2018103008
Juanita	24.05.2021	Juanita J	2018103544

16. APPENDIX B : References

Document name	Description	Link
Emo based music player Authors: Sarvesh Pal, Ankit Mishra, Hridaypratap Mourya	In this project they generate a music playlist based on the current mood of the user. They use **Haarcascade classifier** to extract the facial features. If the user's detected emotion is neutral then the background will be detected and the music will play according to the background.	https://easychair.org/publications/preprint/8RFk
HeartPlayer: A Smart Music Player Involving Emotion Recognition, Expression and Recommendation Authors: Songchun FanCheng TanXin FanHan SuJinyu Zhang.	In this project,when a song is being played, an animation figure expresses the emotion of the song by certain facial expressions. Meanwhile, the emotion of the music item is calculated according to the music features retrieved by a program running in the backstage. In GUI, they use six colors to indicate the six classes of songs with different emotions.	https://link.springer.com/chapter/10.1007/978-3-642-17829-0_47
Speech emotion recognition (SER) through machine learning. Authors: Mohit Wadhwa, Anurag Gupta, Prateek Kumar Pandey	In this paper they use ML algorithms such as SVM, XGB, CNN-1D(Shallow) and CNN-1D on 1D data frame and CNN-2D on 2D-tensor to obtain the underlying emotion from speech audio data and some insights on the human expression of emotion through voice.	https://www.analyticsinsight.net/speech-emotion-recognition-ser-through-machine-learning/

17. APPENDIX C : Key terms

Term	Definition
CNN	Convolutional Neural Network, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.).
Haarcascade algorithm	Haarcascade is an object detection algorithm used to identify faces in an image or a real time video.The algorithm uses edge or line detection features.The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them. The repository has the models stored in XML files, and can be read with the OpenCV methods. These include models for face detection, eye detection, upper body and lower body detection, license plate detection etc.
Word embedding	Word embeddings are a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network.Each word is represented by a real-valued vector, often tens or hundreds of dimensions.