Specification Document
(Emotion Detection 2.0)

Team Members:

- Navya Mamoria
- Akriti Jain
- Aayush Kumar
- Chaitanya Gupta
- Suriya R S

Mentors:

- Agrim
- Mohit

Problem Statement

Analyze speech and video simultaneously to predict emotions.

Introduction

Emotion recognition has become an important topic of research in recent years due to the multiple areas where it can be applied, such as in healthcare or in road safety systems etc.

Human emotions can be detected using speech signals, facial expressions and body language. Thus, an algorithm that performs detection, extraction, and evaluation of these features will allow for automatic recognition of human emotion in images and videos.

In the last five years, the field of AI has made major progress in almost all its standard sub-areas, including vision, speech recognition and generation, image and video generation coupled with advancement in various deep learning techniques, now it is possible to predict human emotions with much higher accuracy.

Our Approach

Since both the audio and video parts are independent of each other, two of us (Akriti and Navya) worked on the audio part and the rest (Aayush, Chaitanya and Suriya) worked on the video model.

VISUAL ANALYSIS

Our WorkFlow

1. Segregating the FER2013 dataset into Train,Test and Validation Dataset
2. Designing a CNN based Network for classifying emotions .
3. Training the Network using the Train_dataset.
4. Obtaining the Results as acc vs epoch and loss vs epoch
5. Testing the trained model in RealTime.

FER 2013

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples.

Models Used

First we tried using normal convolutional layers with BatchNorm ,MaxPooling and dropout in between . We used standard filter size of 3 by 3 , padding as 'same' and activation function as relu . For top layers we added a few DenseLayers ,finishing with a 7 layer classifier . As we were able to

only achive a 58 % val_accuracy through this model we tried various otherthings.

Model 1

First , we started of by building a simple CNN Network with Conv2D, BatchNorm , Dropout and MaxPooling layers. For top layers we added a Dense layers ,toped it with a 7 Neuron classfier

```
#Build Model
model = models.Sequential()

model.add(layers.Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=input_shape, data_format='channels_last'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
```

```
model.add(layers.Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(512, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(512, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Flatten())

model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))

 model.summary()
```

Model 1

Train acc 70 percent Val acc 58 percent

As the validation acc is quite low we went for different Network architecture ,one of them is shown below.

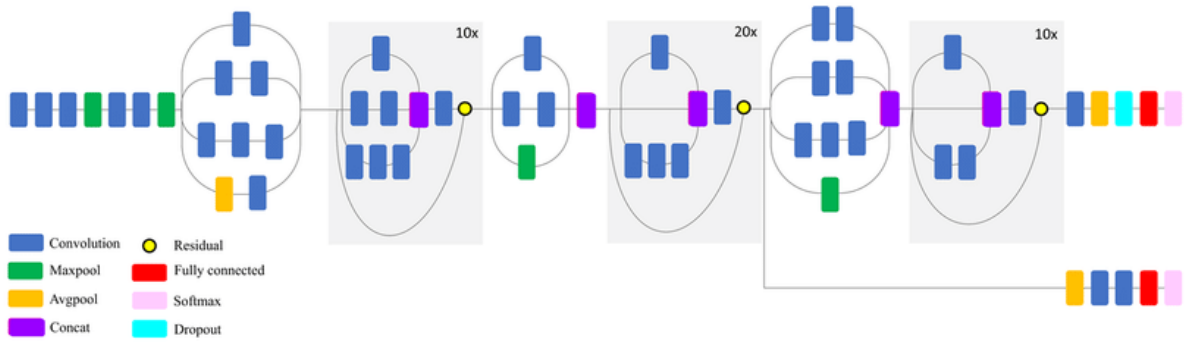
Model 2

In this model we are using a well known and well tested architecture InceptionResnetV2 as our BaseModel. Rest of the concepts used are same as the previous one.

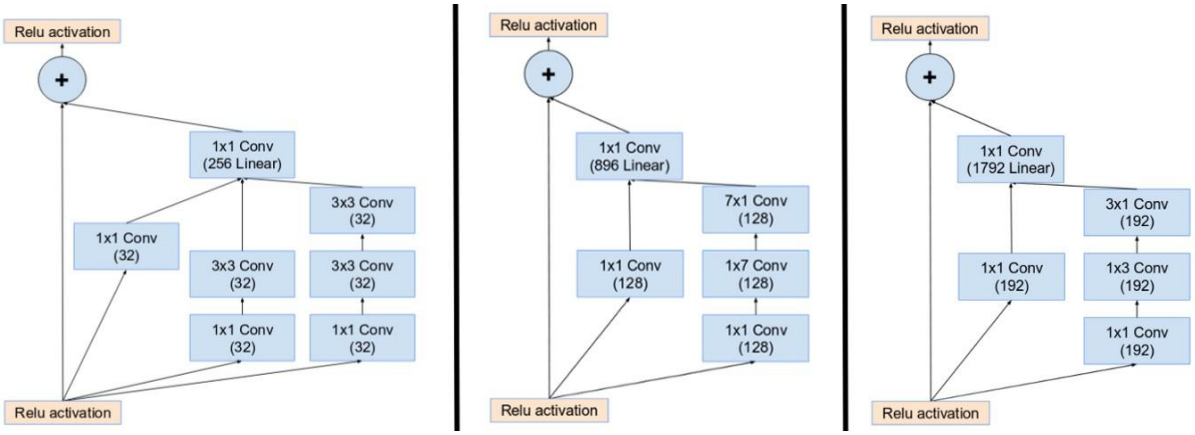
```
InceptionResNetV2=tf.keras.applications.InceptionResNetV2(weights='ima
genet',input_shape=input_shape, include_top=False)
model = models.Sequential()
# load the model
model.add(InceptionResNetV2)

model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.75))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.50))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.25))
model.add(layers.Dense(128,activation='relu'))
model.add(layers.Dropout(0.25))
model.add(layers.Dense(num_classes, activation='softmax'))
model.summary()
```

InceptionResNetV2 Architecture



Inspired from the performance of Inception and ResNet ,the idea of residual links have been introduced into the inception module from InceptionV4 . Based on that 3 different InceptionResnet modules were formed namely A,B and C .



(From left) Inception modules A,B,C in an Inception ResNet. Note how the pooling layer was replaced by the residual connection, and also the additional 1x1 convolution before addition. (Source: Inception v4)

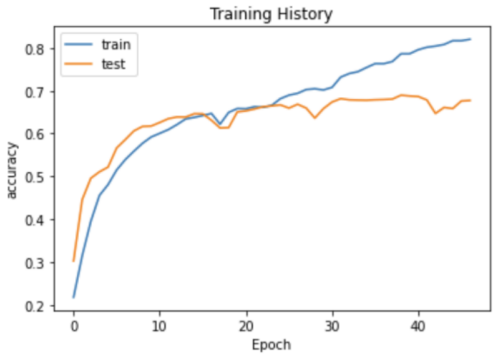
The idea of using multiple convolutional filters and MaxPooling at the sametime towards a featuremap and then combining the results from those multiple operations to form a new feature map forms an inception module(InceptionV1) , doing some minor changes to this to make training faster we will arrive at InceptionV4. ResNet-50 is a Network consists of normal convolutional layers with Residual link . what residual link does is feeds the output of a particular layer to input of next to next layer . Both of these ideas were used to create InceptionResNetV2.

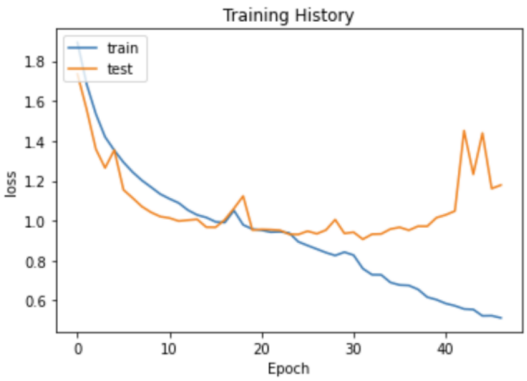
Results

We were able to arrive at a result of (MODEL 2) ,

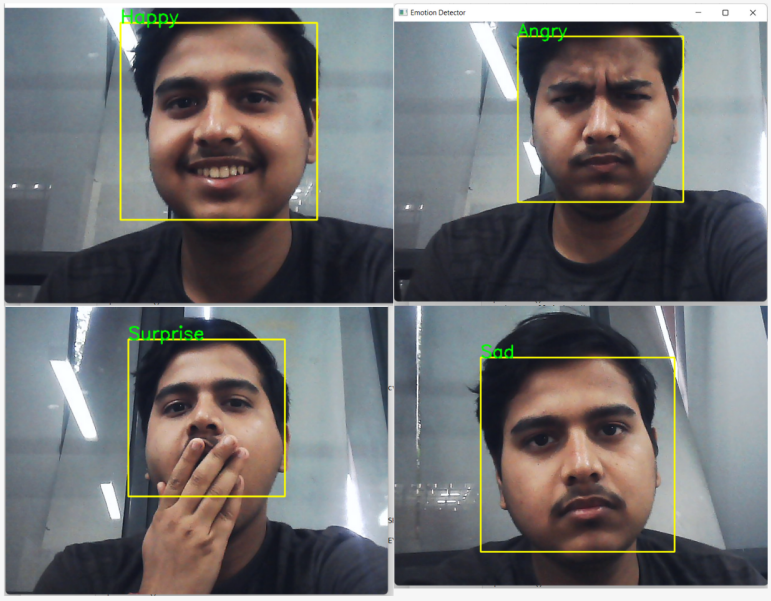
Train accuracy 77 percent
val accuracy 69 percent
test accuracy 68.8 percent

Training and Validation Results .





Real Time Emotion Detection Results



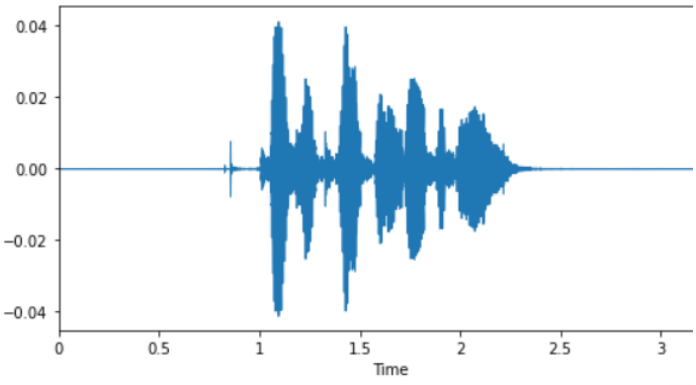
Audio

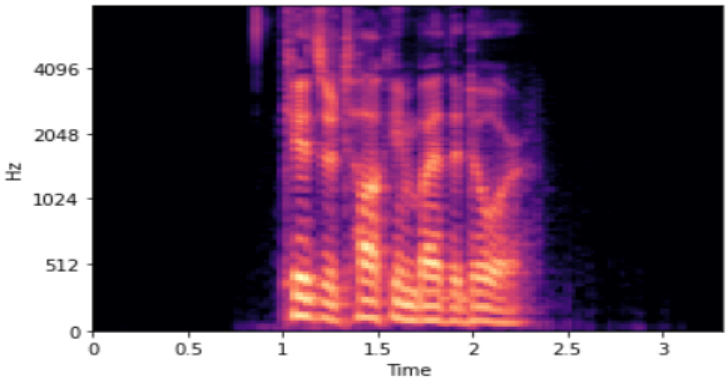
Workflow

Dataset- RAVDESS and TESS
Software used- Google collab
Framework- Sci-kit learn, librosa, and Keras
Algorithm- 1D CNN and/or LSTM
Using TensorFlow backend.

Step 1- Mount the datasets on Google drive and unzip them separately.

Step 2- Test one audio file by plotting its waveform and spectrogram.





Step 3- Feature Extraction
The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.
The digital representation of an audio clip can be easily obtained by using Python packages such as Librosa- a music and audio analysis package.

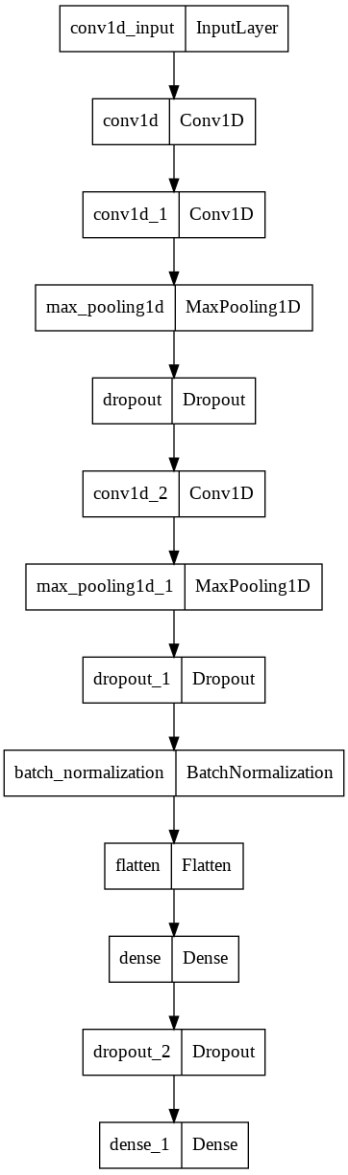
The above plot describes the change in amplitude (loudness) of a signal over time domain. The next challenge is extracting the significant features from this wave form that can easily help to distinguish emotions embedded. We have focussed on two features namely Mel-Frequency Cepstral Coefficients (MFCC) and Mel Spectrogram.
Mel spectrogram plots amplitude on frequency vs time graph on a “Mel” scale. As the project is on emotion recognition, a purely subjective item, we found it better to plot the amplitude on Mel scale as Mel scale changes the recorded frequency to “perceived frequency”.
Sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

Step-4 Creating a data-frame

Step-5 Splitting the data into test and train set

Step-6 Data Preprocessing (using MinMaxScaler)

Step-7 Building 1D CNN model



Step-8 Prediction

Step-9 Confusion Matrix- For checking the accuracy

Conclusion

- 1D CNN model gave an F1 score of 81% for testing seven emotions that are happy, sad, neutral, disgust, angry, fear, and surprise

- . Other technologies like LSTM takes a lot more time for training and testing.

Future work:

- We can add the feature of analyzing real time audio input using either audio recorder software or extracting audio from a real-time video feed.
- Secondly, in order to increase the accuracy, we can also make our model gender sensitive, meaning a gender based recognition model where gender classification is performed first, followed by an emotion classification model of each gender.