



15CSE387 – Open Lab

SER

(Speech Emotion Recognition)

By

Aluvala Nikhil-BL.EN.U4CSE18002

Bachu Krishna Lokesh-BL.EN.U4CSE18011

B.Sai Ram Charan-BL.EN.U4CSE18017

Introduction



- Generally, when we are far from our friends ,relatives. Etc, so we need some way to communicate with them like phone call ,texting(through WhatsApp),video call(through duo).
- But communication is not so effective because we are talking indirectly not face to face .Of Course we have video calls. But making a video call every time is not an efficient idea.
- We use emojis while texting just to know others what's our emotion, feeling, expression.
- As emotion 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.

Problem Statement:

To develop a model that recognize the emotion of a person using speech. Two different models are developed using

- CNN
- MLP classifier

Study on existing systems

- ❖ Many of the existing systems are using CNN models and very few developed using MLP-classifier.
- ❖ The available datasets for this project are RAVADESS,SAVEE,IEMOCAP,TESS.
- ❖ Emotions are subjective and it is hard to notate them.
- ❖ Exploratory Data Analysis always grant us good insight.
- ❖ Lack of data is a crucial factor to achieve success in SER, however, it is complex and very expensive to build a good speech emotion dataset.

Solution

- The problem of speech recognition can be solved by analysing acoustic features.
- Analysis of acoustic features can be done in real time with the help of machine learning while the conversation is taking place as we had just needed the audio data for accomplishing tasks .
- We can represent the emotions in discrete classification which means classifying emotions in discrete labels like anger, happiness, sad. Etc.
- Developing a model using Python libraries and MLP classifier, CNN and training with the dataset and then testing for the accuracy.

Design Methodology



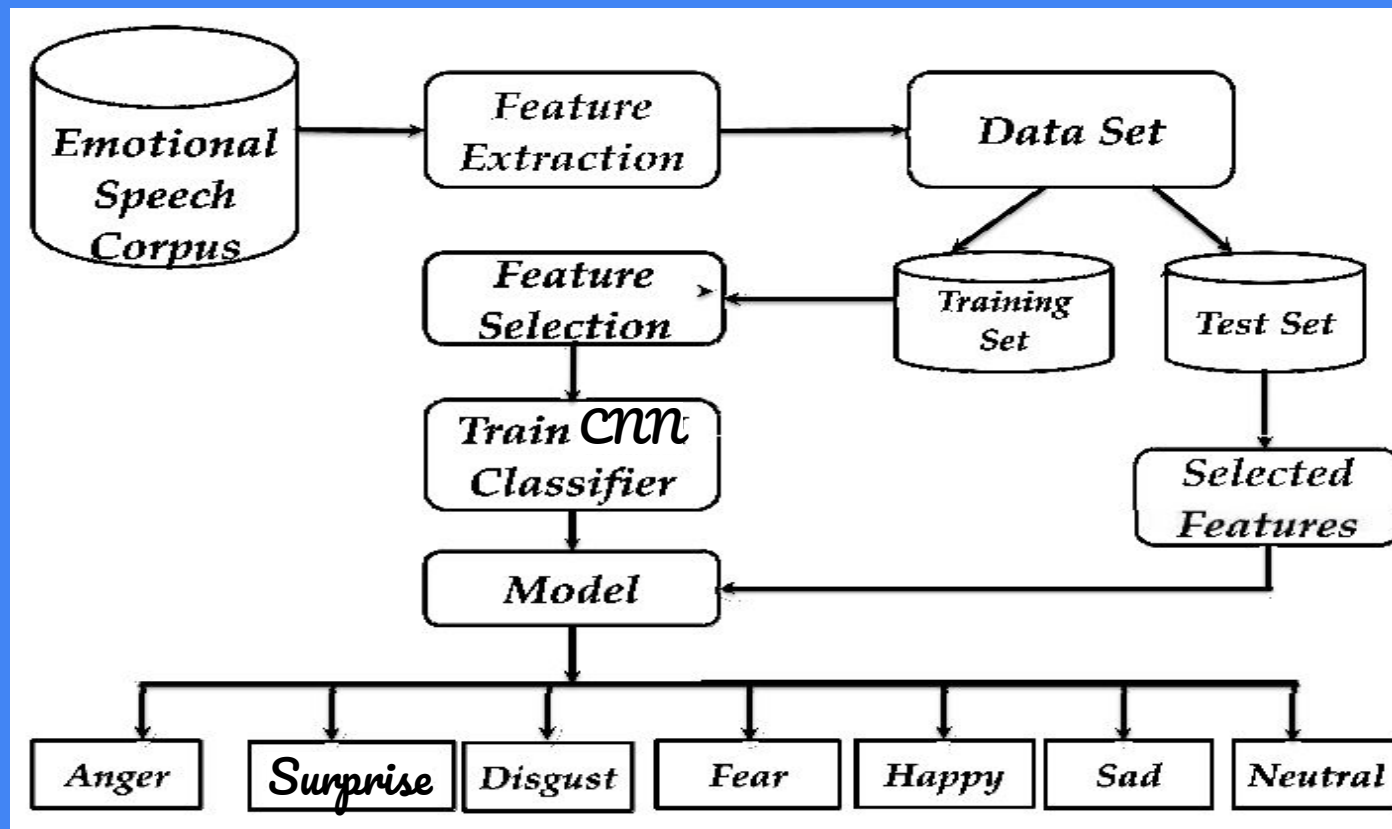
- ❖ Install required libraries
- ❖ Setting up Environment
- ❖ Download the required dataset-RAVDESS
- ❖ Loading the required dataset
- ❖ Plotting to understand audio file through time series data graph
- ❖ Speech to text API
- ❖ Feature extraction
- ❖ Labels classification
- ❖ Loading of data and splitting of data (testing and training)

Design Methodology



- ❖ Mapping of testing data to their corresponding filenames as their labels
- ❖ Building a CNN Model
- ❖ Test the data of RAVDESS
- ❖ Building a MLP-classifier
- ❖ Predict the test data using saved model
- ❖ Summarization of predicted data
- ❖ Developing a Voice Recorder
- ❖ Applying the extract feature on random file and then loading the model to predict the result

Block-Diagram For SER



ACOUSTIC FEATURES



We have two types

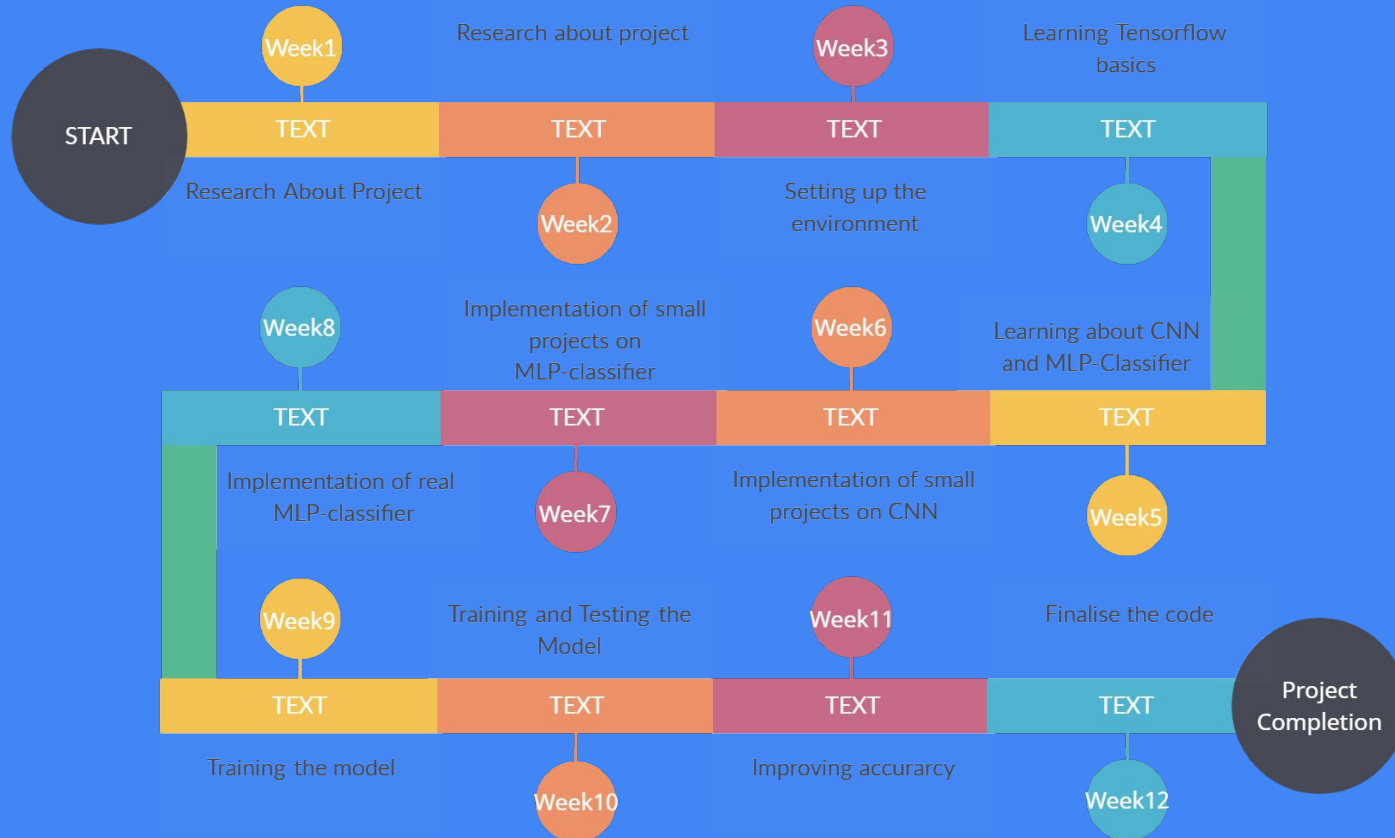
1)MFCC(Mel frequency cepstrual coefficients):

These are coefficients that collectively make up an MFC. There are derived from a type of cepstral representation of the audio clip. The difference between the cepstrum and mel frequency cepstrum is that in MFC, the frequency bands are equally spaced on mel scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in normal spectrum.

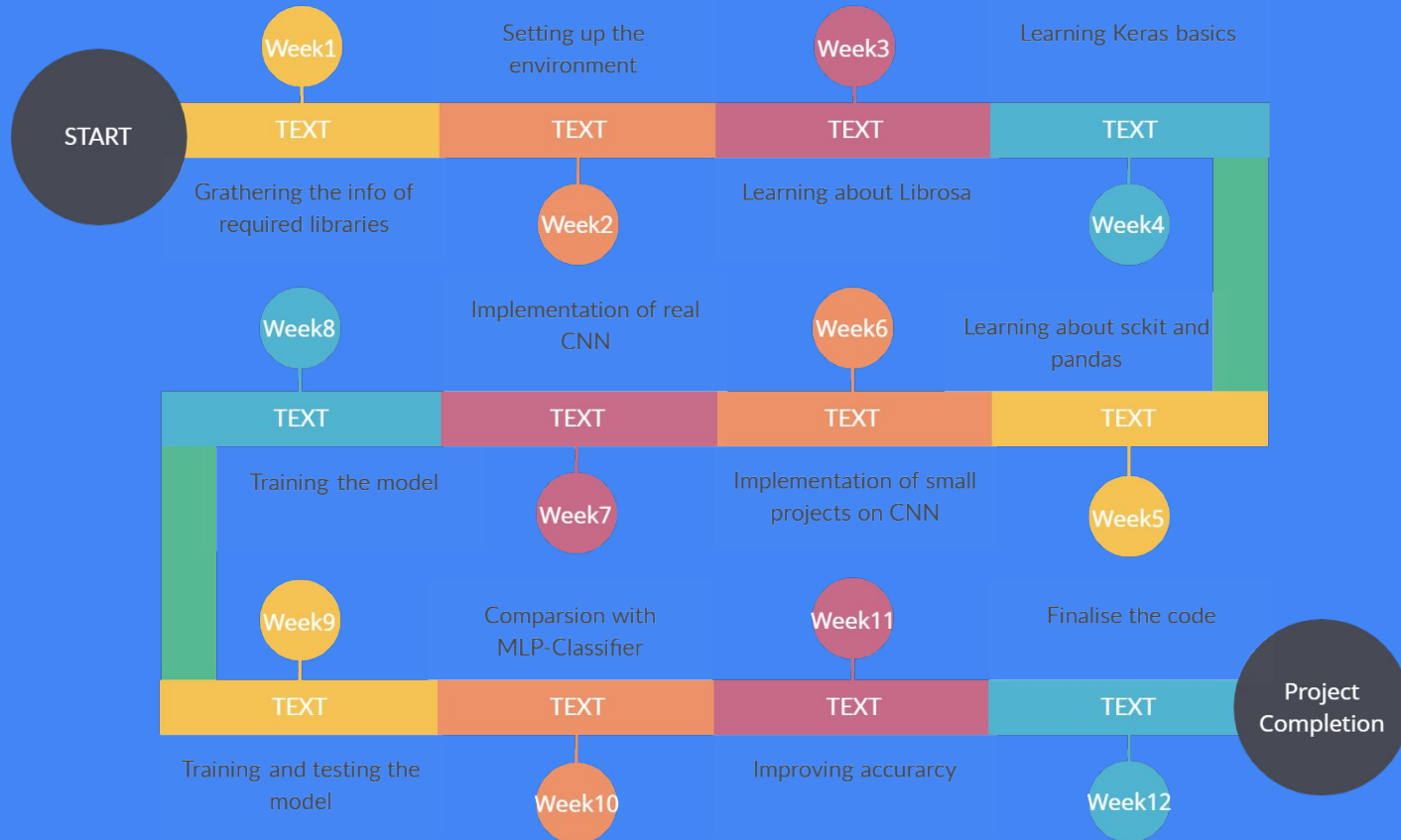
2)Chroma:

Chroma based features, which are also referred to as "pitch class profiles", are a powerful tool for analysing music whose pitches can be meaningfully categorised and whose tuning approximates to the equal tempered scale.

Nikhil Timeline Chart



Lokesh Timeline Chart



Sai Ram Charan Timeline Chart



About Dataset



The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)

- The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains 7356 files (total size: 24.8 GB).
- But we are using 2880 files for the project which are in form of .wav files
- The database contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent.
- Speech includes **calm, happy, sad, angry, fearful, surprise, and disgust , neutral, expressions**, and the song contains calm, happy, sad, angry, and fearful emotions.
- Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.
- In this dataset we are going to use only the speech of the actors.
- Link to download **RAVDESS**

Analysis of Dataset

In the RADVESS dataset, each actor has to perform 8 emotions by saying and singing two sentences and two times for each. As a result, each actor would induce 4 samples for each emotion except neutral, disgust and surprised since there is no singing data for these emotions. Each audio wave is around 4 second, the first and last second are most likely silenced.

The Standard Sentences are:

1. Kids are talking by the door
2. Dogs are sitting by the door

File naming convention

Each of the 7356 RAVDESS files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong).

NOTE: There is no strong intensity for the 'neutral' emotion.

- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Example for file



Filename example: 03-01-06-01-02-01-12.wav

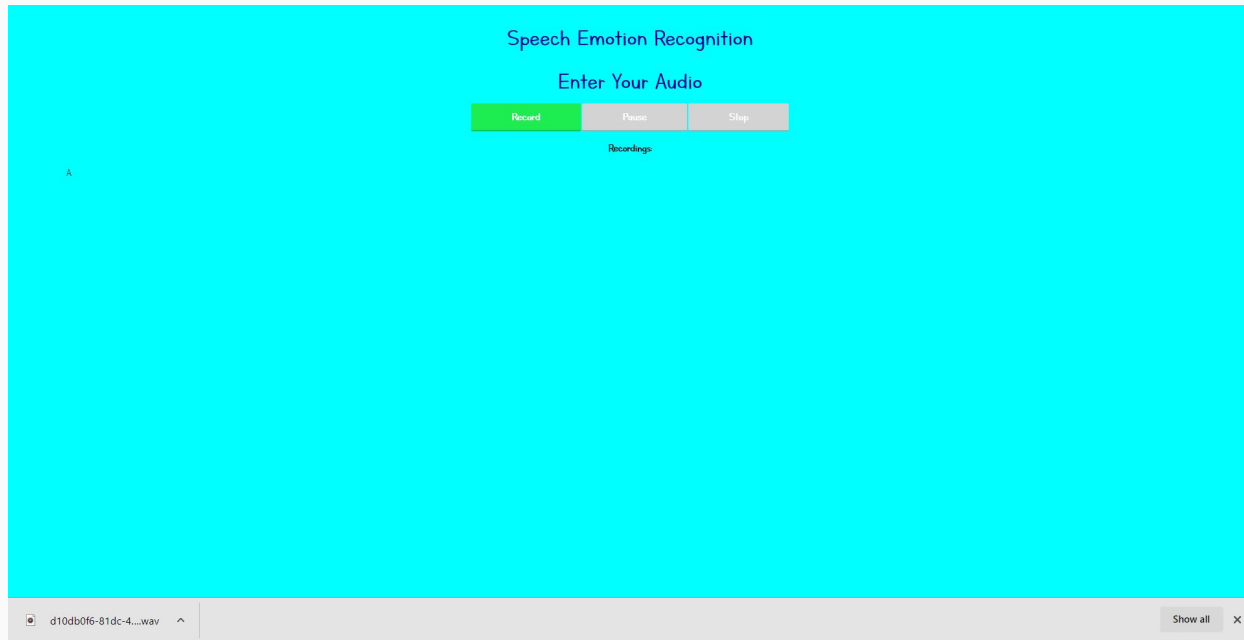
1. Audio-only(03)
2. Speech (01)
3. Fearful (06)
4. Normal intensity (01)
5. Statement "dogs" (02)
6. 1st Repetition (01)
7. 12th Actor (12)
8. Female, as the actor ID number is even.

Observations

After we selected 1 actor and 1 actress's dataset and listened to all of them. I found out male and female are expressing their emotions in a different way. Here are some findings:

- Male's **Angry** is simply increased in volume.
- Male's **Happy** and **Sad** significant features were laughing and crying tone in the silenced period in the audio.
- Female's **Happy**, **Angry** and **Sad** are increased in volume.
- Female's **Disgust** would add vomiting sound inside.

Voice Recorder



This is the
interface the of the
voice recorder.

from tqdm.autonotebook import tqdm



```
os.listdir(path='/content/drive/MyDrive/Speech/')
def getListOfFiles(dirName):
    listOfFile=os.listdir(dirName)
    allFiles=list()
    for entry in listOfFile:
        fullPath=ospathjoin(dirName, entry)
        if ospath.isdir(fullPath):
            allFiles=allFiles + getListOfFiles(fullPath)
        else:
            allFilesappend(fullPath)
    return allFiles

dirName = '/content/drive/MyDrive/Speech/'
listOfFile = getListOfFiles(dirName)
len(listOfFile)
```



1440

```
[25] from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

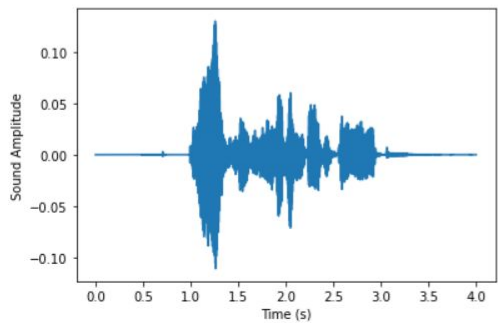
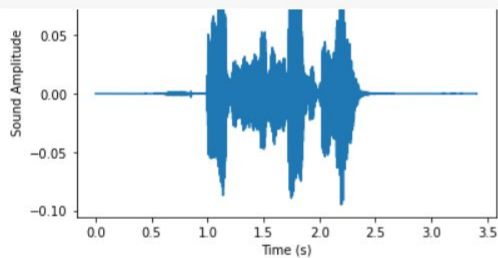
```
[12] !pip install SpeechRecognition
```


[]

```
#Plotting the Basic Graphs for understanding of Audio Files :
for file in range(0, len(listOfFiles), 1):
    audio, sfreq = lload(listOfFiles[file])
    time = np.arange(0, len(audio)) / sfreq
    fig, ax = plt.subplots()
```

```
#Plotting the Basic Graphs for understanding of Audio Files :
```

```
for file in range(0 , len(listOffiles) , 1):  
    audio , sfreq = lload(listOffiles[file])  
    time = np.arange(0 , len(audio)) / sfreq  
    fig , ax = plt.subplots()  
    axplot(time , audio)  
    ax.set(xlabel = 'Time (s)' , ylabel = 'Sound Amplitude')  
    plt.show()
```



Accuracy

- For the first time, we used only voice files of the RAVDESS and by using MLP classifier we could get an accuracy of 42.78%.

+ Code + Text

```
[42] model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

```
model.fit(x_train, y_train)
```

```
MLPClassifier(activation='relu', alpha=0.01, batch_size=256, beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(300,), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=5000, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

+ Code

+ Text

```
[45] y_pred=model.predict(x_test)
```

```
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
# Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

Accuracy: 42.78%

```
[31] from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
angry	0.66	0.47	0.55	49
calm	0.62	0.28	0.38	47

Improving Accuracy of MLP-classifier

```
# Print the accuracy  
print("Accuracy: {:.2f}%".format(accuracy*100))
```

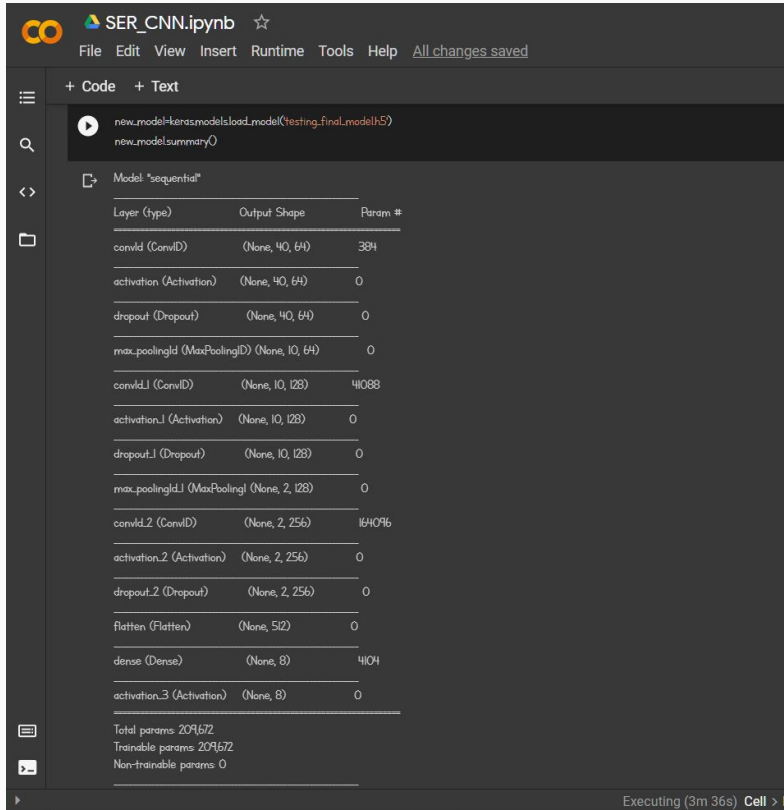
Accuracy: 76.25%

```
from sklearn.metrics import classification_report  
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
angry	0.88	0.82	0.85	92
calm	0.77	0.83	0.80	102
disgust	0.71	0.79	0.75	92
fearful	0.80	0.81	0.81	102
happy	0.73	0.87	0.79	97
neutral	0.46	0.86	0.60	44
sad	1.00	0.38	0.55	89
surprised	0.89	0.75	0.81	102
accuracy			0.76	720
macro avg	0.78	0.77	0.75	720
weighted avg	0.80	0.76	0.76	720

- On the second phase, we tried with the dataset that contains the voice and got the accuracy of 76% with MLP classifier

Model Summary of CNN model



The screenshot shows a Jupyter Notebook interface with a file named 'SER_CNN.ipynb'. The code cell contains the following lines:

```
new_model=keras.models.load_model('testing_final_model.h5')
new_model.summary()
```

The output displays the summary of a 'Model "sequential"' with the following layers and parameters:

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 40, 64)	384
activation (Activation)	(None, 40, 64)	0
dropout (Dropout)	(None, 40, 64)	0
max_pooling1d (MaxPooling1D)	(None, 10, 64)	0
conv1d_1 (Conv1D)	(None, 10, 128)	41088
activation_1 (Activation)	(None, 10, 128)	0
dropout_1 (Dropout)	(None, 10, 128)	0
max_pooling1d_1 (MaxPooling1D)	(None, 2, 128)	0
conv1d_2 (Conv1D)	(None, 2, 256)	164096
activation_2 (Activation)	(None, 2, 256)	0
dropout_2 (Dropout)	(None, 2, 256)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 8)	4104
activation_3 (Activation)	(None, 8)	0

Summary statistics:

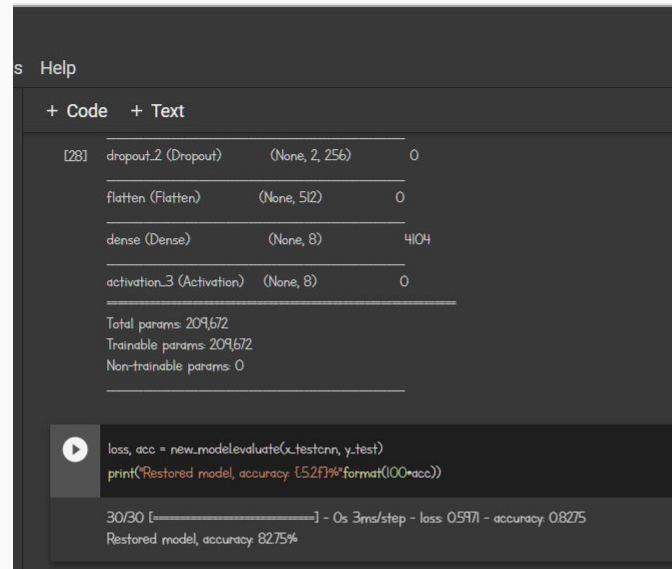
- Total params: 209672
- Trainable params: 209672
- Non-trainable params: 0

The bottom status bar indicates 'Executing (3m 36s) Cell'.

- In CNN model, we used 3 Convolution layers and 4 activation functions(3 Relu,1 Softmax), dropout layers.
- At last the optimiser used id RMSprop.

Accuracy of CNN Model

- By the RAVDESS dataset with voice, we could achieve 82.5% accuracy.



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there are tabs for 'Code' and 'Text'. Below the tabs, a table displays the model's architecture layers. The layers are: dropout_2 (Dropout) with parameters (None, 2, 256) and 0 trainable parameters; flatten (Flatten) with parameters (None, 512) and 0 trainable parameters; dense (Dense) with parameters (None, 8) and 4104 trainable parameters; and activation_3 (Activation) with parameters (None, 8) and 0 trainable parameters. Below the table, the total number of parameters is 209,672, and the number of trainable parameters is also 209,672. At the bottom, a code cell shows the evaluation results: 'loss, acc = new_model.evaluate(x_test_cnn, y_test)' and 'print("Restored model, accuracy {:.52f}%".format(100*acc))'. The output of the code cell shows '30/30 [-----] - 0s 3ms/step - loss: 0.5971 - accuracy: 0.8275' and 'Restored model, accuracy 82.75%'.

Layer	Parameters	Trainable Parameters
dropout_2 (Dropout)	(None, 2, 256)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 8)	4104
activation_3 (Activation)	(None, 8)	0

Total params: 209,672
Trainable params: 209,672
Non-trainable params: 0

```
loss, acc = new_model.evaluate(x_test_cnn, y_test)
print("Restored model, accuracy {:.52f}%".format(100*acc))
```

30/30 [-----] - 0s 3ms/step - loss: 0.5971 - accuracy: 0.8275
Restored model, accuracy 82.75%

References



- <https://zenodo.org/record/1188976#.YCDoT-gzZ EZ>
- <https://docs.python.org/3/library/glob.html>
- <https://librosa.org/doc/latest/index.html>
- <https://pypi.org/project/noisereduce/>
- https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/audio/ipynb/speaker_recognition_using_cnn.ipynb#scrollTo=h1sg2ToIrIVl
- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/audio/simple_audio.ipynb#scrollTo=zRxauKMdhofU

Thank you