A Mini- Project Report

on

“Audio based Sentiment Analysis using Deep Learning”

Submitted to the

Pune Institute of Computer Technology, Pune

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by

Sakshi Tantak	43159
Riddhi Toshniwal	43171
Vishap Malik	43175

Under the guidance of

Prof. R. R. Chhajed

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2020-2021

CERTIFICATE

This is to certify that the project report entitled

Audio based Sentiment Analysis

Submitted by

Sakshi Tantak	43159
Riddhi Toshniwal	43171
Vishap Malik	43175

is a bonafide work carried out by them under the supervision of Prof. R. R.Chhajed and it is approved
for the partial fulfillment of the requirement of **Computer Laboratory -X** for the award of the Degree of Bachelor of Engineering (Information Technology)

Prof. R. R. Chhajed
Lab Teacher
Department of Information Technology

Dr. A. M. Bagade
Head of Department
Department of Information Technology

Place:
Date:

ACKNOWLEDGEMENT

We thank everyone who has helped and provided valuable suggestions for successfully creating a wonderful project.

We are very grateful to our guide, Prof. R. R. Chhajed, Head of Department Dr. A. M. Bagade and our principal Dr. P. T. Kulkarni. They have been very supportive and have ensured that all facilities remained available for smooth progress of the project.

We would like to thank our professor and Prof. R. R. Chhajed for providing very valuable and timely suggestions and help.

Student Name

ABSTRACT

Speech or audio emotion recognition purpose is to automatically identify the emotional or physical state of a human being from his voice. The emotional state of a person hidden in his speech is an important factor of human communication and interaction as it provides feedback in communication without altering linguistic contents. Analysis of emotions or sentiment from audio inputs and voice data has numerous applications in the real world. We propose a deep learning based end-to-end system that lets the users test themselves in a simulated interview environment so as to help the users prepare for interviews and realize their flaws and fortitude in their voice while in an interview, while also providing feedback to the user, relative to the performance of other candidates. The aim of this system is to make audio-based deep learning systems more accessible and interactive with a clean and intuitive UI.

LIST OF FIGURES

LIST OF TABLES

LIST OF TOPICS

1. INTRODUCTION

Emotion recognition is a challenging task that goes beyond conventional sentiment analysis by including many more emotions apart from the conventional neutral, positive and negative. Several sentiment analysis models can be found in use today because of the prevailing need to recognize sentiment of user/client for several reasons such as to detect spam/fraud, generate feedback, etc. Most common sentiment analysis models generate only a binary output for specialized tasks. Such classifiers are called Binary classifiers. Other models which recognize emotions from input such as sad, happy, excited etc. require specialized datasets and much more computation and better algorithms.

Our emotion recognition model is trained on the Ryerson Audio-Visual Database of Emotional Speech and Song(RAVDEES) which contains 7356 files made of 24 professional actors vocalizing two lexically-matched statements. Emotions enacted include calm, happy, sad, angry, surprise, fearful and disgust with an added neutral expression to each statement. This model trains on the dataset using CNN and classifies future input accordingly.

Several linguistic-models can be used to analyze a person’s personality and the same may be used in modern interview processes. In our project, we use the Multi-class emotion recognition model, for the task of familiarizing the inflicting emotion in their speech by simulating an interview-like environment by generating an input-window of 15 seconds for answering to the model. Leveraging the results of this, the user may be actively working to bring about a change in mannerisms to bring forth a positive interview experience.

2. SCOPE AND OBJECTIVE

2.1. SCOPE

The project is focussed on providing users/students with a concise tool to understand their shortcomings especially in the speech part of their interviews while comparing their results with other users of the tool. This is achieved by simulating an interview experience where the user has to answer a question selected from a question pool within 15 seconds. The easily understandable nature of the tool has an added advantage and is usable by users within a varied age demographic. Easily understandable graphs to show evaluation help in understanding the results for even layman users.

2.2. OBJECTIVE

Our objective while making this project is to familiarize the user with the format of modern online interviews where the user is prompted to answer certain questions within a span of time and is evaluated on the same using deep learning techniques for audio sentiment analysis. Sounding confident, excited or happy is important in such situations which is not possible when the user is not familiar with such a concept of interview. Our project is a very easy-to-interact tool which makes the users familiar with such interview processes and gives them an exhaustive evaluation of their performance in the test with simple diagrams for better understanding and showing them their performance in comparison to other users.

3. ARCHITECTURE

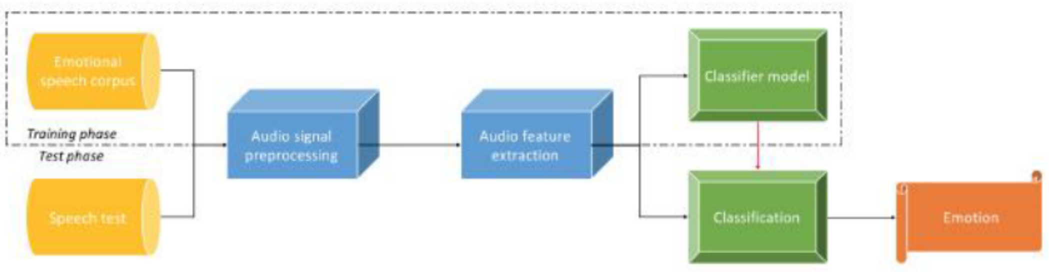


Fig. 1. Audio Sentiment Analysis Pipeline

The usual process for audio sentiment analysis consists of three parts : signal processing, feature extraction and classification. Signal processing applies an acoustic filter on original audio signals and splits it into meaningful units. The feature extraction is the sensitive point in speech emotion recognition because features need to both efficiently characterize the emotional content of a human speech and not depend on the lexical content or even the speaker. Finally, emotion classification will map feature matrix to emotion labels.

3.1. CHOICE OF MODEL

This system uses an emotion recognition model trained on the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDEES) which contains 7356 files made of 24 professional actors vocalizing two lexically-matched statements. Emotions enacted include calm, happy, sad, angry, surprise, fearful and disgust with an added neutral expression to each statement. This model is trained on the dataset using CNN and classifies future input accordingly.

3.2. FEATURE EXTRACTION

Once having extracted the basic speech features from the preprocessed audio signal, we obtain a matrix of features per audio le. We then compute the first derivatives of each of those features to capture frame to frame changes in the signal. Finally, we calculate the following global statistics on these features: mean, median, standard deviation, kurtosis, skewness, 1% percentile, 99% percentile, min, max and range between min and max. Thereby, a vector of 200 candidate features is obtained for each audio signal. Normalization could be meaningful as extracted feature values have different orders of magnitude or different units. Secondly, it is also common to use dimensionality reduction techniques in order to reduce the memory and computation requirements of the classifier. There are two options for dimensionality reduction: features selection (statistical tests) and features transformation (Principal Component Analysis).

The mel-frequency scale is a quasi-logarithmic spacing roughly resembling the resolution of the human auditory system. To compute log-mel-spectrogram you only have to apply the Mel-spaced lterbank (set of L triangular lters) to the audio spectrogram and get the logarithm of the result.

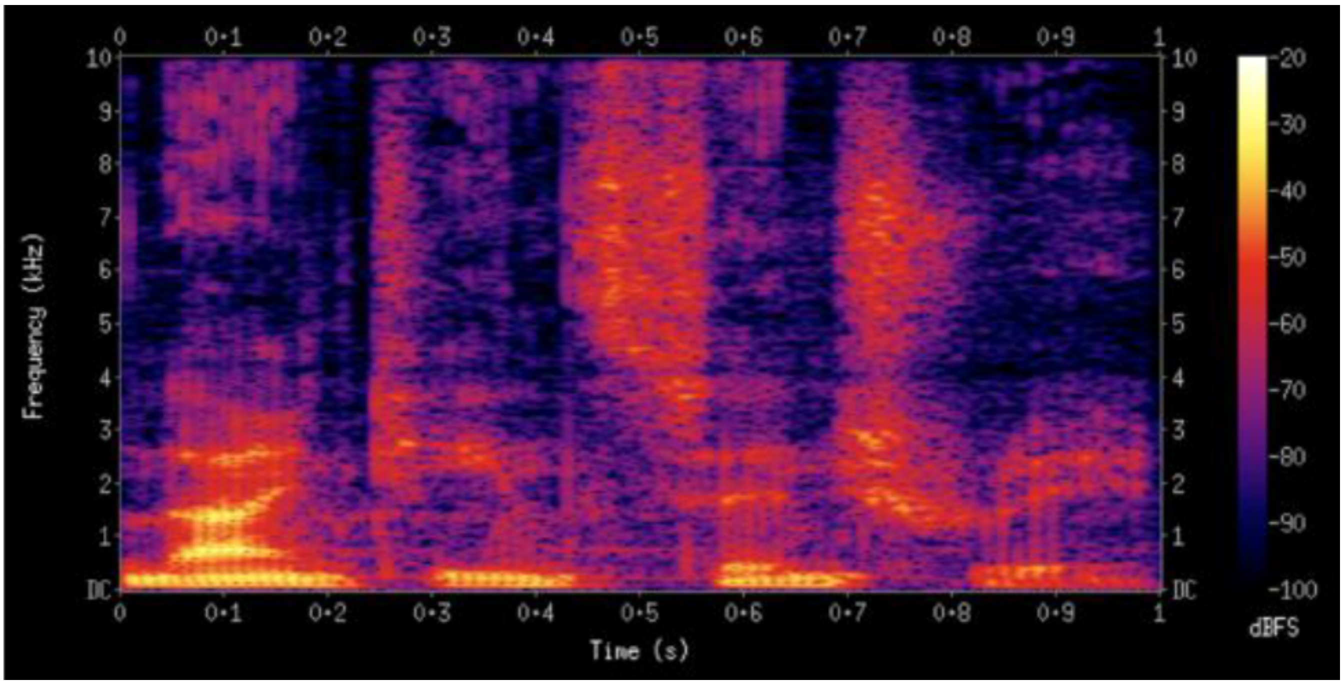
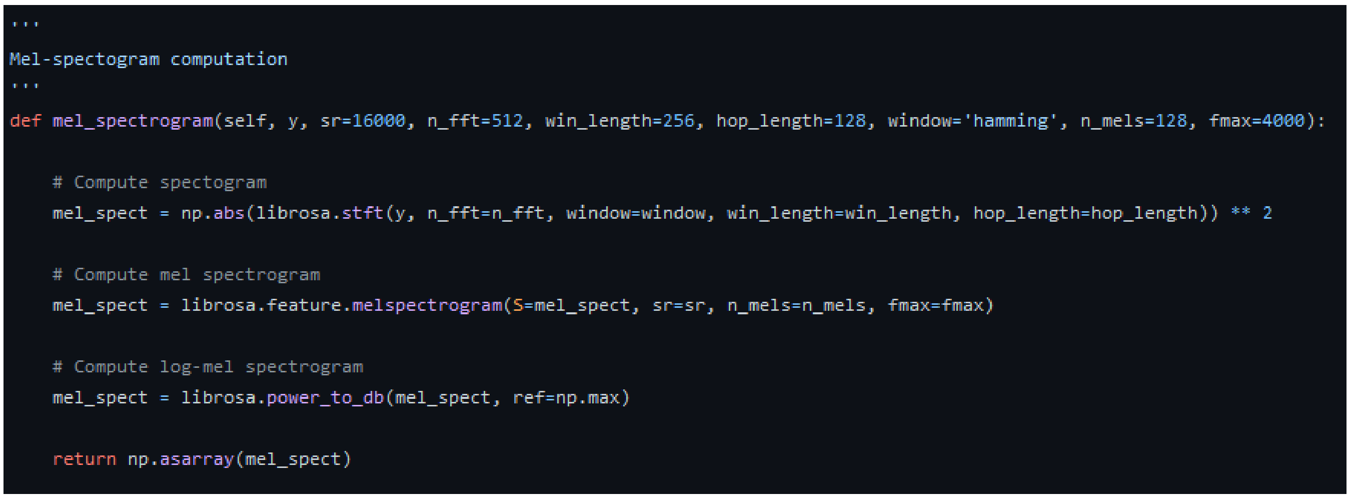


Fig. 2. Log-Mel Spectrogram of an audio signal

Following is the snapshot of the code that creates the mel-spectrogram of audio input



```
'''
Mel-spectogram computation
'''
def mel_spectrogram(self, y, sr=16000, n_fft=512, win_length=256, hop_length=128, window='hamming', n_mels=128, fmax=4000):

    # Compute spectogram
    mel_spect = np.abs(librosa.stft(y, n_fft=n_fft, window=window, win_length=win_length, hop_length=hop_length)) ** 2

    # Compute mel spectrogram
    mel_spect = librosa.feature.melspectrogram(S=mel_spect, sr=sr, n_mels=n_mels, fmax=fmax)

    # Compute log-mel spectrogram
    mel_spect = librosa.power_to_db(mel_spect, ref=np.max)

    return np.asarray(mel_spect)
```

Fig. 3. Code snapshot of mel-spectrogram

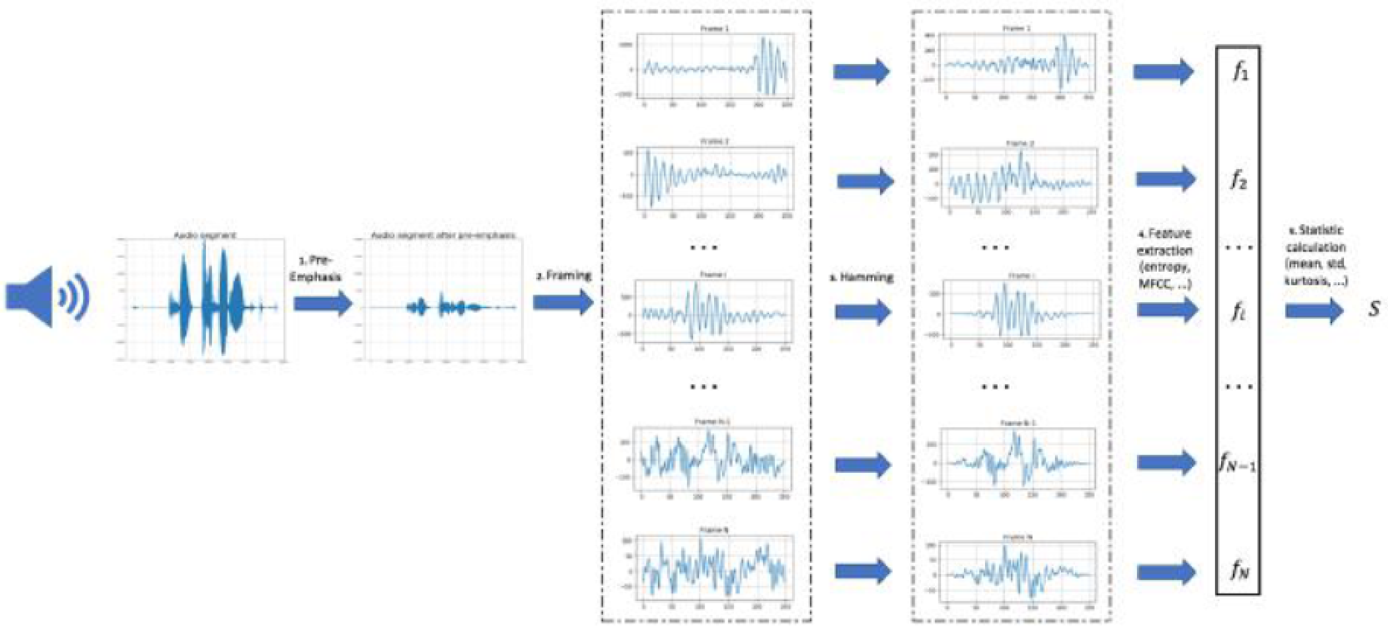
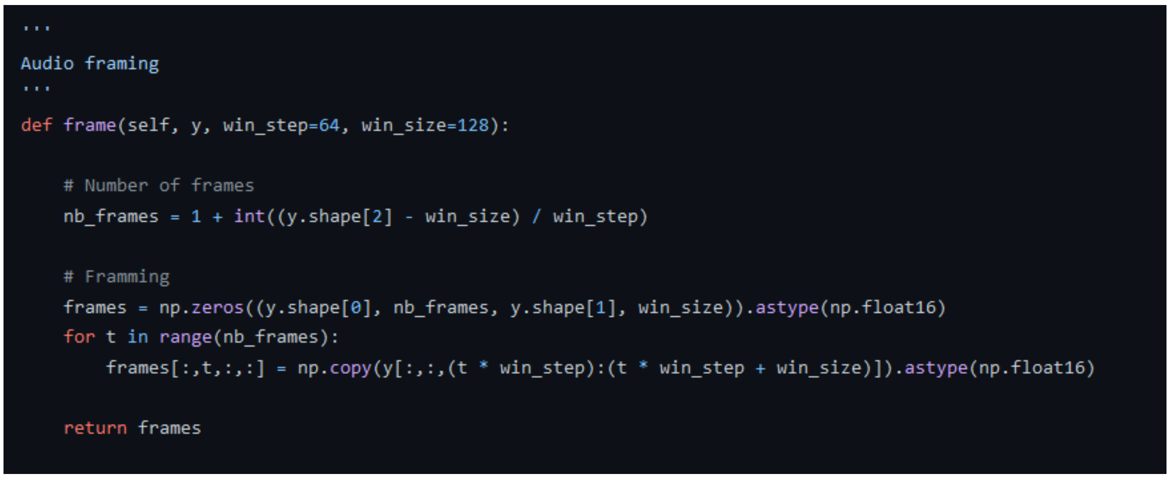

Audio segment
1. Pre-Emphasis
Audio segment after pre-emphasis
2. Framing
Frame 1
Frame 2
Frame i
Frame N-1
Frame N
3. Hamming
4. Feature extraction (entropy, MFCC, ...)
f_1
f_2
f_i
f_{N-1}
f_N
5. Statistic calculation (mean, std, kurtosis, ...)
S

Fig. 4. Signal Processing and audio feature extraction

Following is the snapshot of the code that does framing of audio input



```
'''
Audio framing
'''
def frame(self, y, win_step=64, win_size=128):

    # Number of frames
    nb_frames = 1 + int((y.shape[2] - win_size) / win_step)

    # Framming
    frames = np.zeros((y.shape[0], nb_frames, y.shape[1], win_size)).astype(np.float16)
    for t in range(nb_frames):
        frames[:,t,:,:] = np.copy(y[:,:,(t * win_step):(t * win_step + win_size)]).astype(np.float16)

    return frames
```

Fig. 5. Code snapshot of framing

3.3. CLASSIFIER

Convolutional Neural Networks (CNNs) show remarkable recognition performance for computer vision tasks while Recurrent Neural Networks (RNNs) show impressive achievement in many sequential data processing tasks. The concept of time distributed convolutional neural network is to combine a deep hierarchical CNNs feature extraction architecture with a recurrent neural network model that can learn to recognize sequential dynamics in a speech signal. This network only takes the log-mel-spectrogram as input. The main idea of time distributed CNN is to apply a rolling window (fixed size and time-step) all along the log-mel-spectrogram. Each of these windows will be the entry of a convolutional neural network, composed by four Local Feature Learning Blocks (LFLBs) and the output of each of these convolutional networks will be fed into a recurrent neural network composed by 2 cells LSTM (Long Short Term Memory) to learn the long-term contextual dependencies. Finally, a fully connected layer with softmax activation is used to predict the emotion detected in the voice.

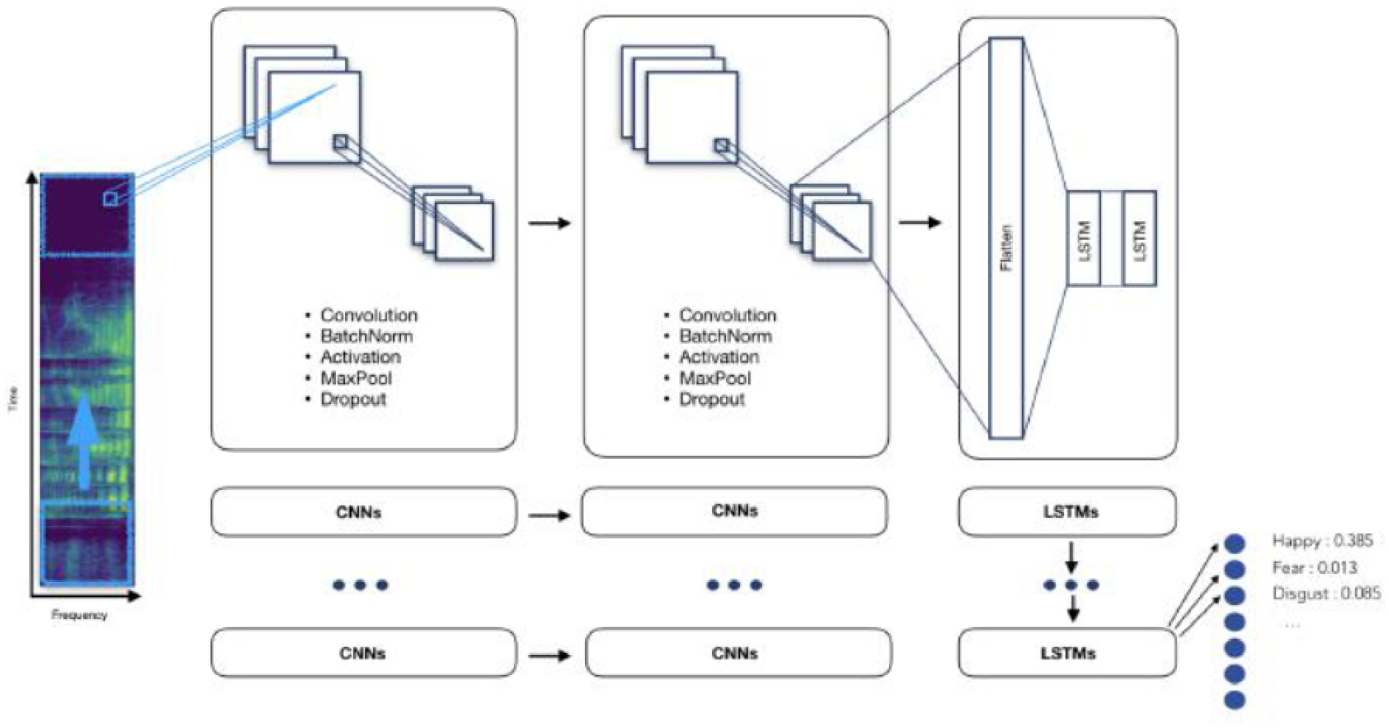
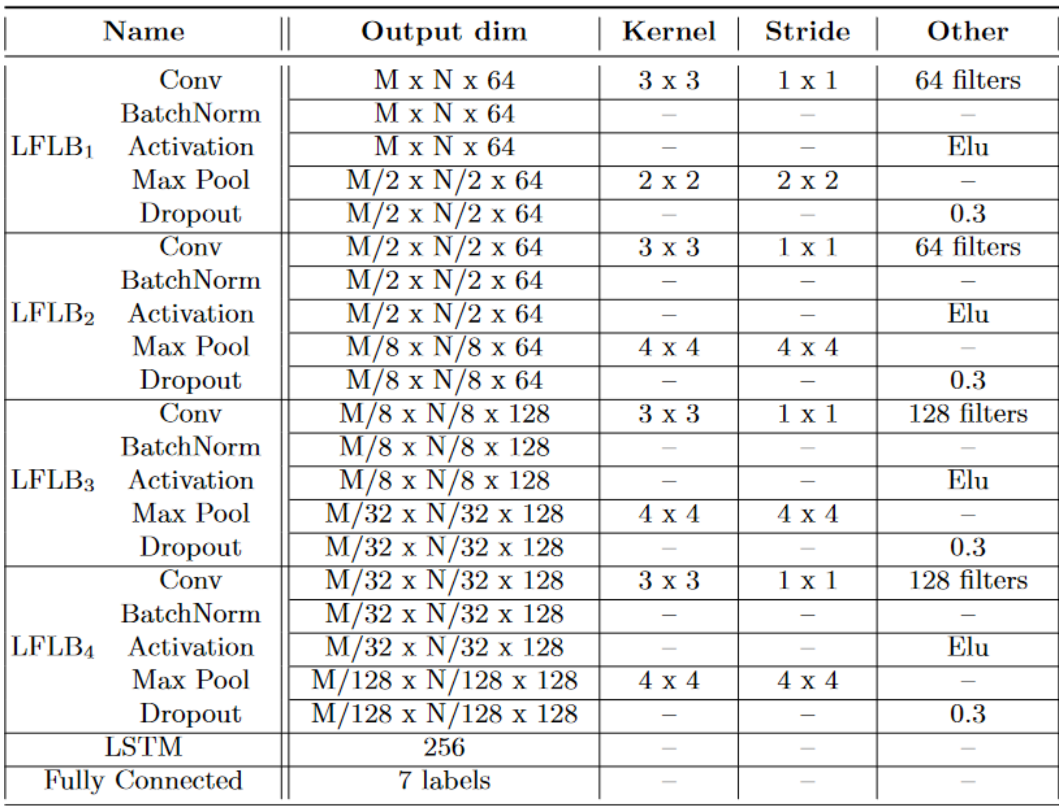


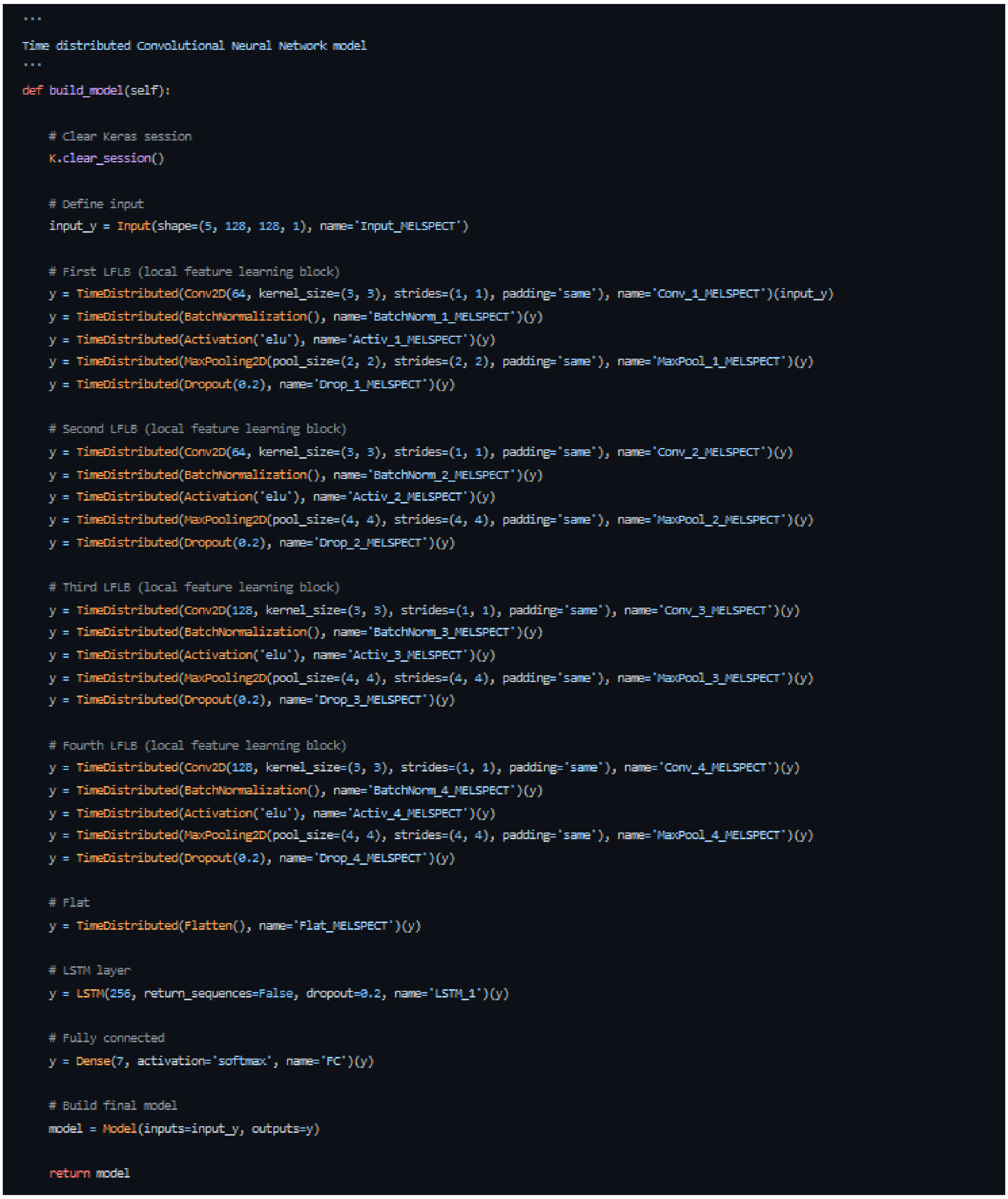
Fig. 6. Time distributed CNN schema



Name		Output dim	Kernel	Stride	Other
	Conv	M x N x 64	3 x 3	1 x 1	64 filters
	BatchNorm	M x N x 64	–	–	–
$LFLB_1$	Activation	M x N x 64	–	–	Elu
	Max Pool	M/2 x N/2 x 64	2 x 2	2 x 2	–
	Dropout	M/2 x N/2 x 64	–	–	0.3
	Conv	M/2 x N/2 x 64	3 x 3	1 x 1	64 filters
	BatchNorm	M/2 x N/2 x 64	–	–	–
$LFLB_2$	Activation	M/2 x N/2 x 64	–	–	Elu
	Max Pool	M/8 x N/8 x 64	4 x 4	4 x 4	–
	Dropout	M/8 x N/8 x 64	–	–	0.3
	Conv	M/8 x N/8 x 128	3 x 3	1 x 1	128 filters
	BatchNorm	M/8 x N/8 x 128	–	–	–
$LFLB_3$	Activation	M/8 x N/8 x 128	–	–	Elu
	Max Pool	M/32 x N/32 x 128	4 x 4	4 x 4	–
	Dropout	M/32 x N/32 x 128	–	–	0.3
	Conv	M/32 x N/32 x 128	3 x 3	1 x 1	128 filters
	BatchNorm	M/32 x N/32 x 128	–	–	–
$LFLB_4$	Activation	M/32 x N/32 x 128	–	–	Elu
	Max Pool	M/128 x N/128 x 128	4 x 4	4 x 4	–
	Dropout	M/128 x N/128 x 128	–	–	0.3
LSTM		256	–	–	–
Fully Connected		7 labels	–	–	–

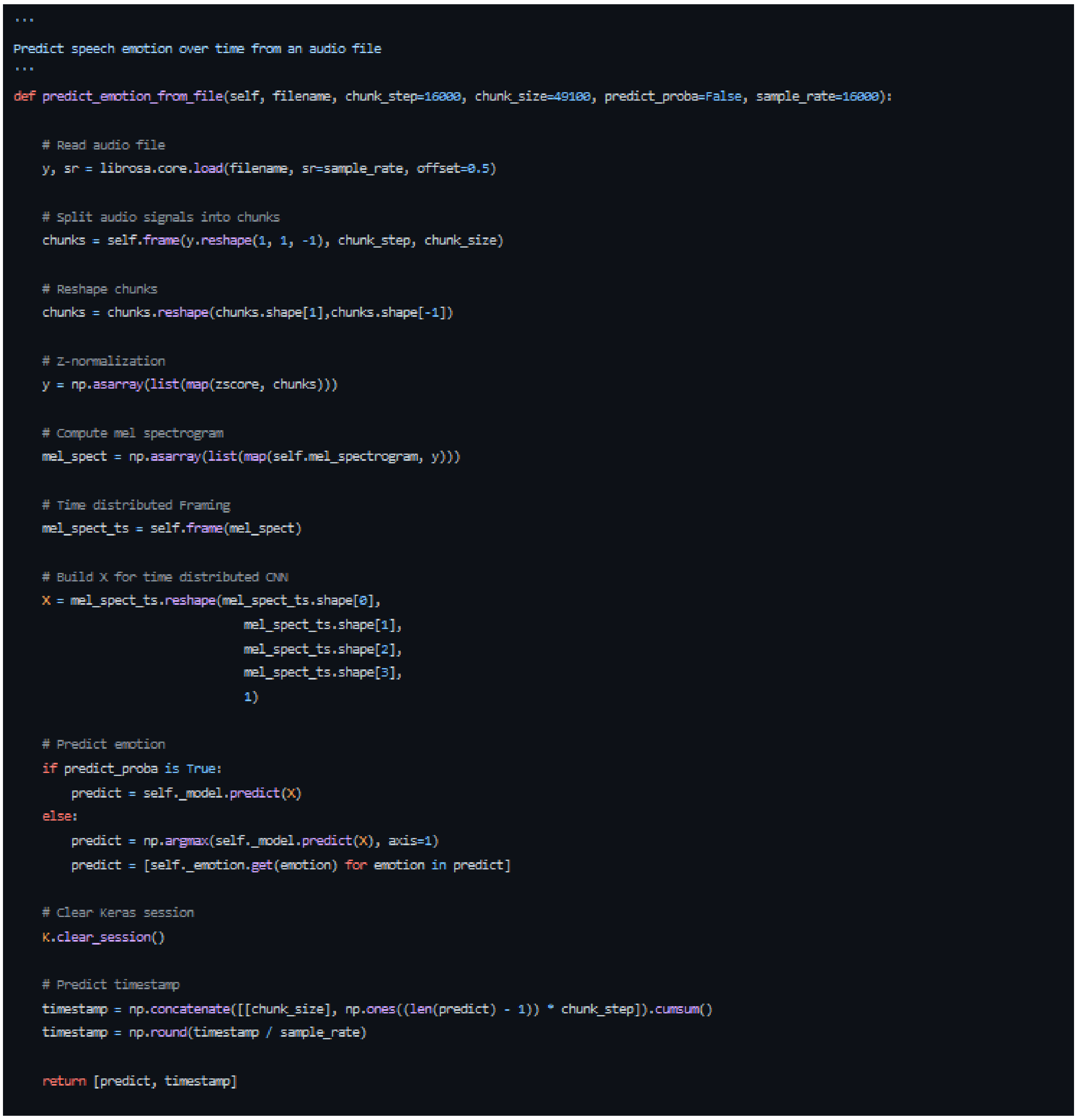
Table 1. Layer parameters of time distributed CNN

The weights of the saved model are loaded and a keras model is built and inference is done. Following is the snapshot of the code that does this



```
'''
Time distributed Convolutional Neural Network model
'''
def build_model(self):

    # Clear Keras session
    K.clear_session()

    # Define input
    input_y = Input(shape=(5, 128, 128, 1), name='Input_MELSPECT')

    # First LFLB (local feature learning block)
    y = TimeDistributed(Conv2D(64, kernel_size=(3, 3), strides=(1, 1), padding='same'), name='Conv_1_MELSPECT')(input_y)
    y = TimeDistributed(BatchNormalization(), name='BatchNorm_1_MELSPECT')(y)
    y = TimeDistributed(Activation('elu'), name='Activ_1_MELSPECT')(y)
    y = TimeDistributed(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'), name='MaxPool_1_MELSPECT')(y)
    y = TimeDistributed(Dropout(0.2), name='Drop_1_MELSPECT')(y)

    # Second LFLB (local feature learning block)
    y = TimeDistributed(Conv2D(64, kernel_size=(3, 3), strides=(1, 1), padding='same'), name='Conv_2_MELSPECT')(y)
    y = TimeDistributed(BatchNormalization(), name='BatchNorm_2_MELSPECT')(y)
    y = TimeDistributed(Activation('elu'), name='Activ_2_MELSPECT')(y)
    y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'), name='MaxPool_2_MELSPECT')(y)
    y = TimeDistributed(Dropout(0.2), name='Drop_2_MELSPECT')(y)

    # Third LFLB (local feature learning block)
    y = TimeDistributed(Conv2D(128, kernel_size=(3, 3), strides=(1, 1), padding='same'), name='Conv_3_MELSPECT')(y)
    y = TimeDistributed(BatchNormalization(), name='BatchNorm_3_MELSPECT')(y)
    y = TimeDistributed(Activation('elu'), name='Activ_3_MELSPECT')(y)
    y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'), name='MaxPool_3_MELSPECT')(y)
    y = TimeDistributed(Dropout(0.2), name='Drop_3_MELSPECT')(y)

    # Fourth LFLB (local feature learning block)
    y = TimeDistributed(Conv2D(128, kernel_size=(3, 3), strides=(1, 1), padding='same'), name='Conv_4_MELSPECT')(y)
    y = TimeDistributed(BatchNormalization(), name='BatchNorm_4_MELSPECT')(y)
    y = TimeDistributed(Activation('elu'), name='Activ_4_MELSPECT')(y)
    y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4), padding='same'), name='MaxPool_4_MELSPECT')(y)
    y = TimeDistributed(Dropout(0.2), name='Drop_4_MELSPECT')(y)

    # Flat
    y = TimeDistributed(Flatten(), name='Flat_MELSPECT')(y)

    # LSTM layer
    y = LSTM(256, return_sequences=False, dropout=0.2, name='LSTM_1')(y)

    # Fully connected
    y = Dense(7, activation='softmax', name='FC')(y)

    # Build final model
    model = Model(inputs=input_y, outputs=y)

    return model
```

Fig. 7. Code snapshot of model being built

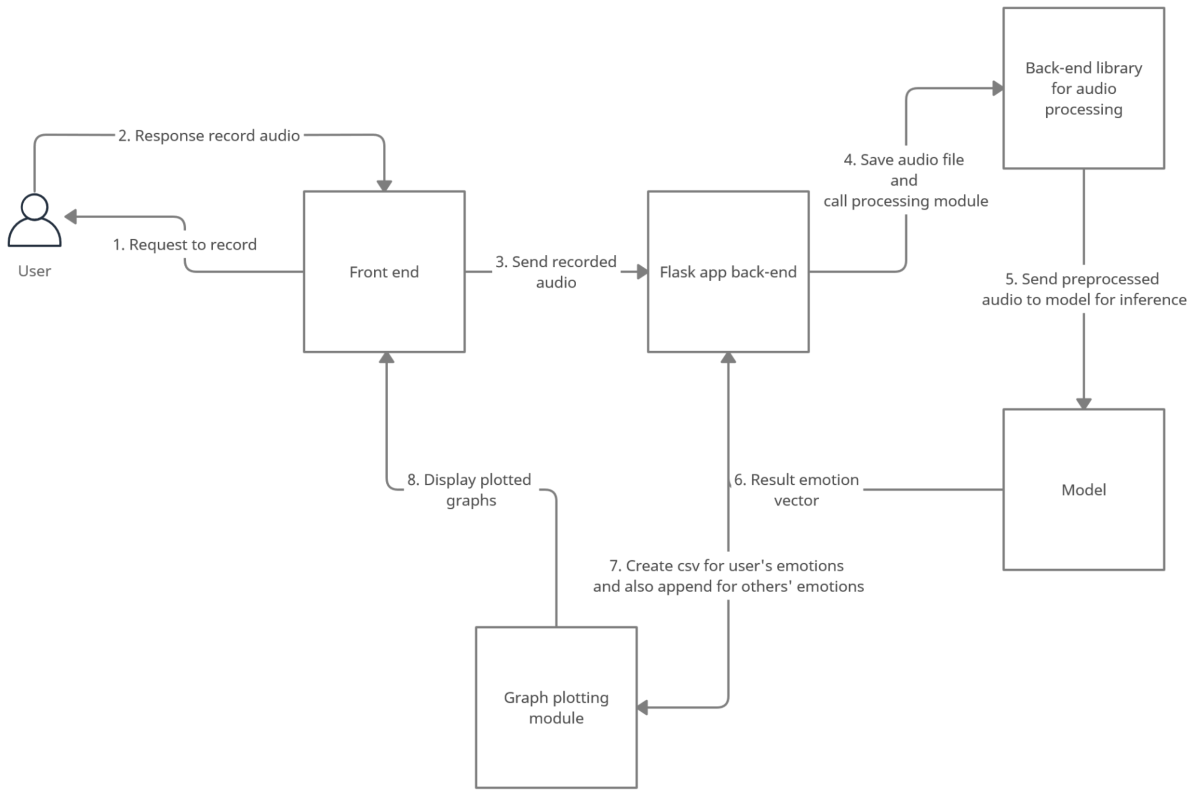


```
'''
Predict speech emotion over time from an audio file
'''
def predict_emotion_from_file(self, filename, chunk_step=16000, chunk_size=49100, predict_proba=False, sample_rate=16000):

    # Read audio file
    y, sr = librosa.core.load(filename, sr=sample_rate, offset=0.5)

    # Split audio signals into chunks
    chunks = self.frame(y.reshape(1, 1, -1), chunk_step, chunk_size)

    # Reshape chunks
    chunks = chunks.reshape(chunks.shape[1],chunks.shape[-1])

    # Z-normalization
    y = np.asarray(list(map(zscore, chunks)))

    # Compute mel spectrogram
    mel_spect = np.asarray(list(map(self.mel_spectrogram, y)))

    # Time distributed Framing
    mel_spect_ts = self.frame(mel_spect)

    # Build X for time distributed CNN
    X = mel_spect_ts.reshape(mel_spect_ts.shape[0],
                             mel_spect_ts.shape[1],
                             mel_spect_ts.shape[2],
                             mel_spect_ts.shape[3],
                             1)

    # Predict emotion
    if predict_proba is True:
        predict = self._model.predict(X)
    else:
        predict = np.argmax(self._model.predict(X), axis=1)
        predict = [self._emotion.get(emotion) for emotion in predict]

    # Clear Keras session
    K.clear_session()

    # Predict timestamp
    timestamp = np.concatenate([[chunk_size], np.ones((len(predict) - 1)) * chunk_step]).cumsum()
    timestamp = np.round(timestamp / sample_rate)

    return [predict, timestamp]
```

Fig. 8. Code snapshot of model inference

4. SYSTEM DESIGN

The system is designed and deployed as a web-based application. It uses various libraries of python for the task of audio preprocessing and recording and flask for the purpose of deployment. The user is prompted to record and answer the said question. The recording goes to the backend flask app that does the audio processing and inference on model. Following diagram explains the design of the system.



User
1. Request to record
2. Response record audio
Front end
3. Send recorded audio
Flask app back-end
4. Save audio file and call processing module
Back-end library for audio processing
5. Send preprocessed audio to model for inference
Model
6. Result emotion vector
7. Create csv for user's emotions and also append for others' emotions
Graph plotting module
8. Display plotted graphs

Fig. 9. System Design

5. REQUIREMENTS

5.1. HARDWARE REQUIREMENTS

The hardware requirements to use the system are :

1. Processor : Intel/AMD
2. GPU : Not necessary but recommended
3. Audio sensor

5.2. SOFTWARE REQUIREMENTS

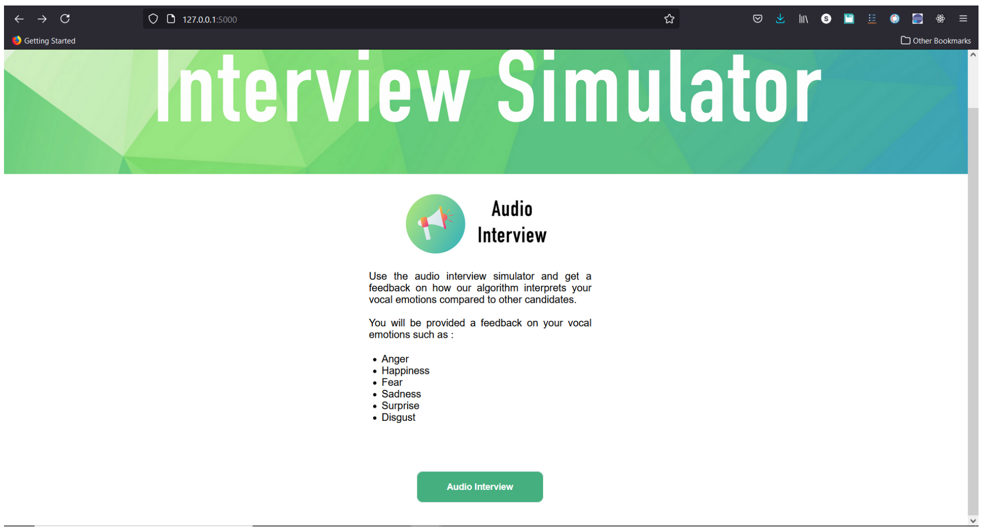
The software requirements to use the system are :

1. Python Libraries : numpy, pandas, flask, collections, pyaudio, wave, scipy, tensorflow
2. Browser : Latest versions of Mozilla/Microsoft Edge/Google Chrome recommended
3. JavaScript
4. Front-end : HTML, CSS

6. RESULTS

The model used in this system obtains a maximum score of 74% on the validation set and 72% on the test set.

Following are the snapshots of the system

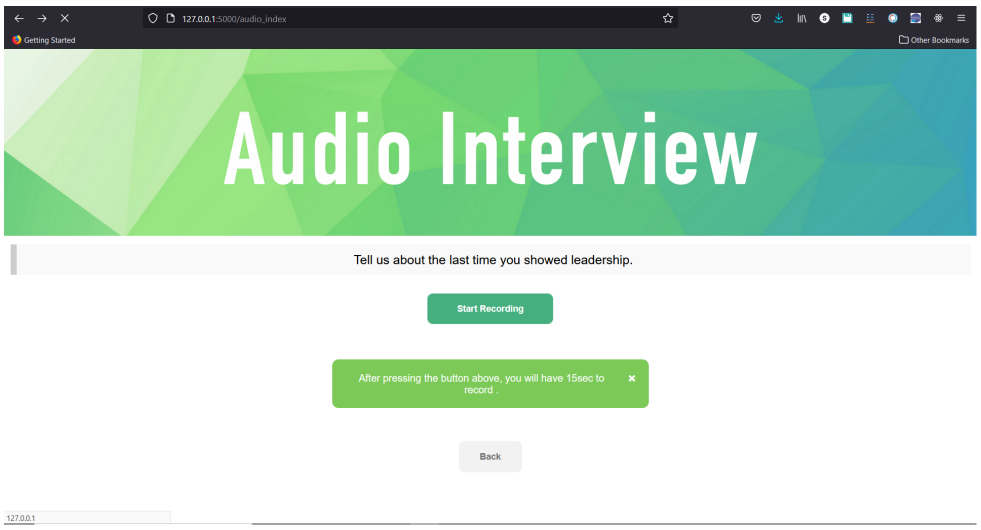


Use the audio interview simulator and get a feedback on how our algorithm interprets your vocal emotions compared to other candidates.

You will be provided a feedback on your vocal emotions such as :

- Anger
- Happiness
- Fear
- Sadness
- Surprise
- Disgust

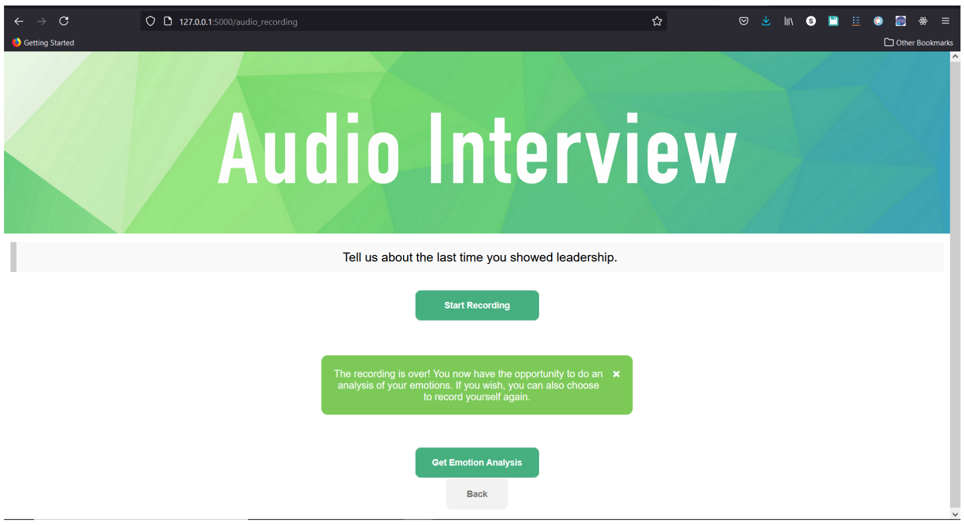
Fig. 10. Landing Page



Tell us about the last time you showed leadership.

After pressing the button above, you will have 15sec to record .

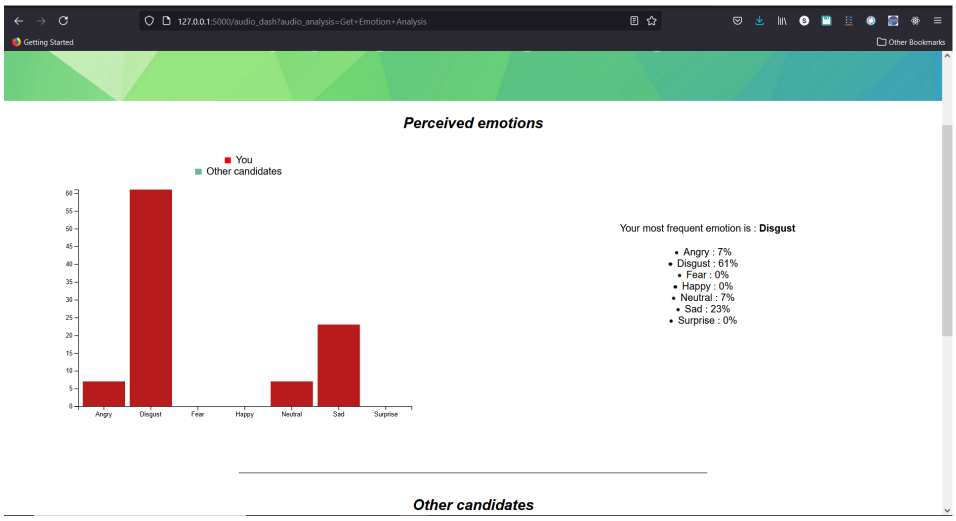
Fig. 11. Audio Sentiment Analysis page



Tell us about the last time you showed leadership.

The recording is over! You now have the opportunity to do an analysis of your emotions. If you wish, you can also choose to record yourself again.

Fig. 12. Recording page



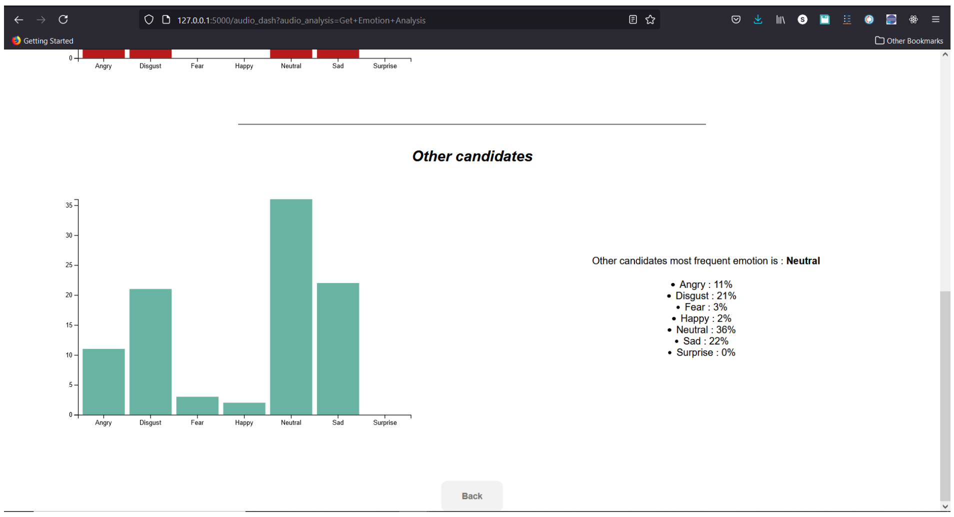
Perceived emotions

Your most frequent emotion is : **Disgust**

- Angry : 7%
- Disgust : 61%
- Fear : 0%
- Happy : 0%
- Neutral : 7%
- Sad : 23%
- Surprise : 0%

Other candidates

Fig. 13. User’s analysis graphs



Other candidates

Other candidates most frequent emotion is : **Neutral**

- Angry : 11%
- Disgust : 21%
- Fear : 3%
- Happy : 2%
- Neutral : 36%
- Sad : 22%
- Surprise : 0%

Fig. 14. Comparative analysis graphs

7. CONCLUSION

7.1. FUTURE SCOPE

We wish to expand the scope of the system in future to incorporate other forms of emotion and sentiment recognition based on video, text and audio to build a complete interview simulation system that can give valuable feedback to its users using more sophisticated models with a higher accuracy.

7.2. CONCLUSION

This system focuses on operationalizing a deep learning based audio sentiment analysis model that can provide valuable feedback to its users via an intuitive front-end. The system uses modern web-development technologies that make it easy to deploy and host the system in future. It is possible to build a system as we have with more forms of input other than audio input and we thus wish to expand its scope in future towards the same goal of building a complete multimodal emotion recognition system that can provide a seamless and valuable interview simulation environment.

8. REFERENCES

1. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), https://zenodo.org/record/1188976/?f=3.XAcEs5NKhQK
2. B. Basharirad,and M. Moradhaseli. *Speech emotion recognition methods : A literature review. AIPConference Proceedings 2017.*
   https://aip.scitation.org/doi/pdf/10.1063/1.5005438
3. L. Chen, M. Mao, Y. Xueand L.L. Cheng. *Speech emotion recognition: Features and classification models.* Digit. Signal Process, vol 22 Dec. 2012.
4. T. Vogt, E. Andreand J. Wagner. *Automatic Recognition of Emotions from Speech : A Review of the Literature and Recommendations for Practical Realisation.* Affect and Emotion in Human-Computer Interaction, 2008.
5. T. Giannakopoulos. *pyAudioAnalysis: An Open-Source Python Library for AudioSignal Analysis. Dec. 2015*
   https://doi.org/10.1371/journal.pone.0144610
6. *End-to-End Multimodal Emotion Recognition using Deep Neural Networks*
   https://arxiv.org/pdf/1704.08619.pdf

GitHub Repo : https://github.com/sakshitantak/Interview-Simulator