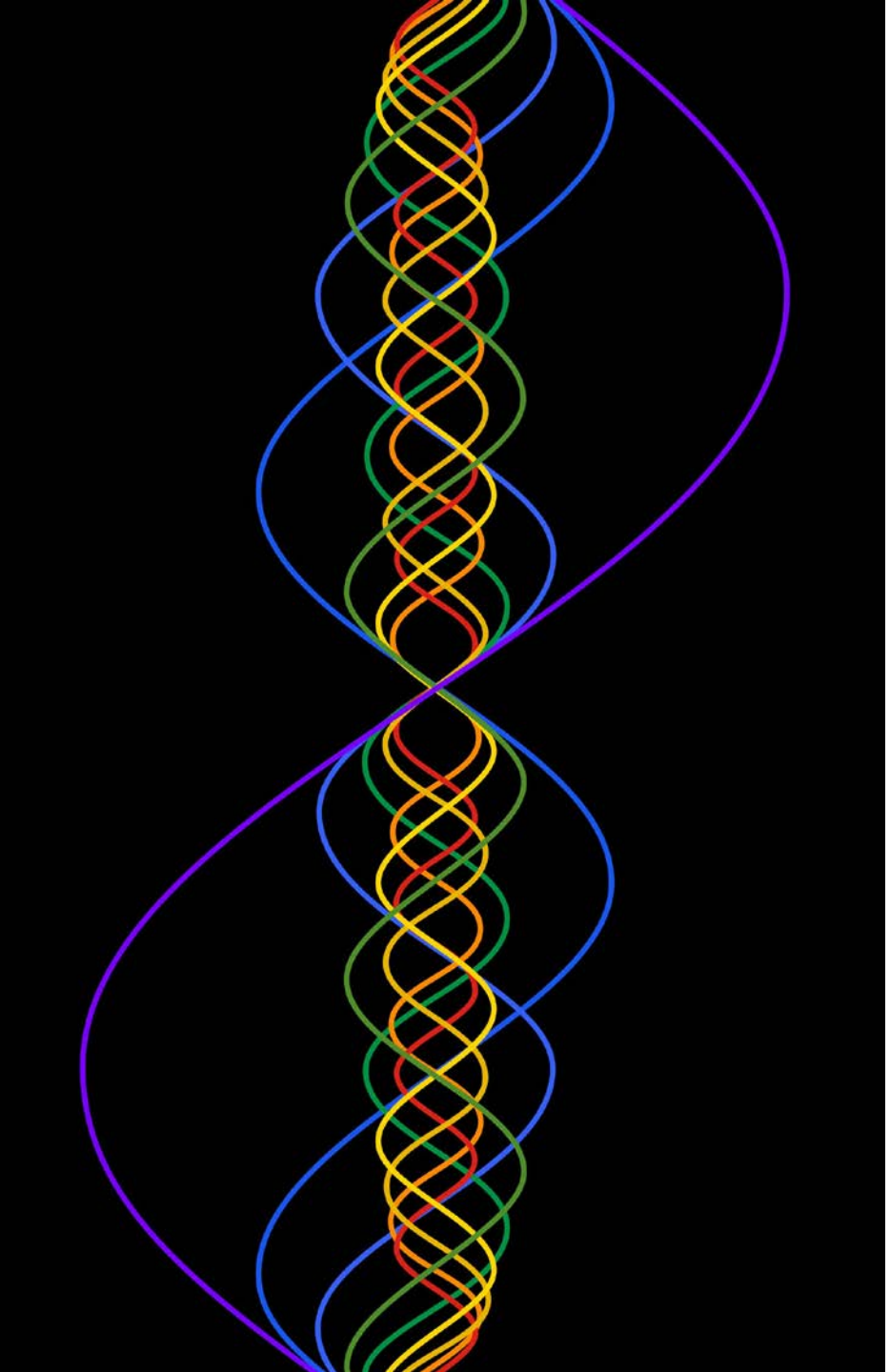
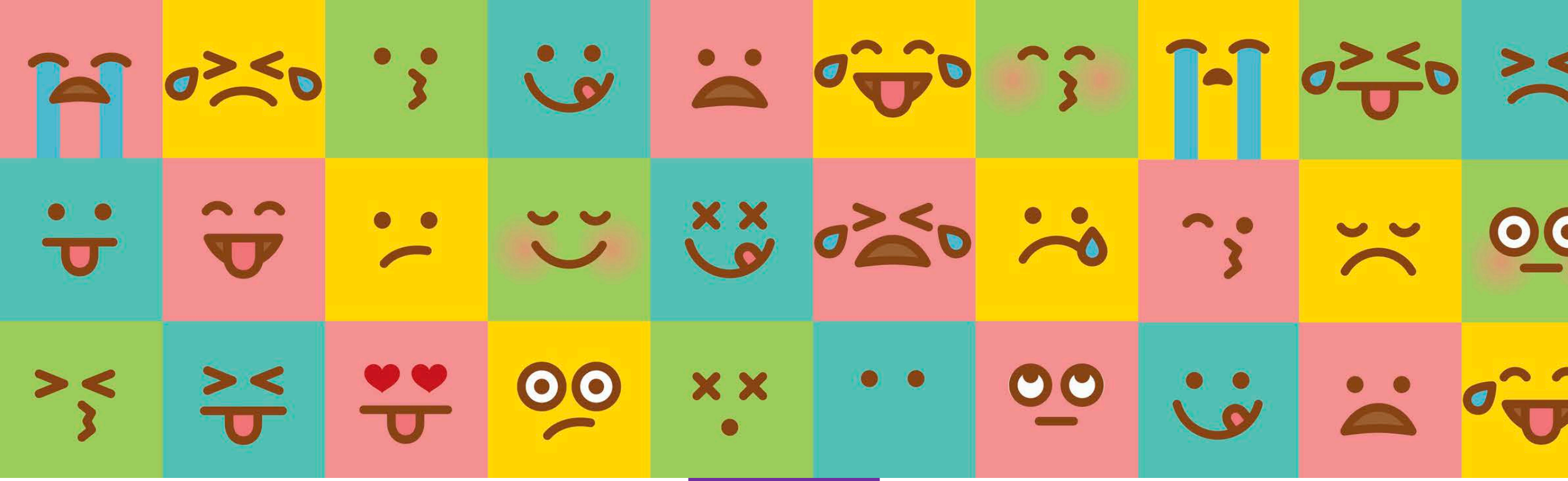


SPEECH EMOTION RECOGNITION

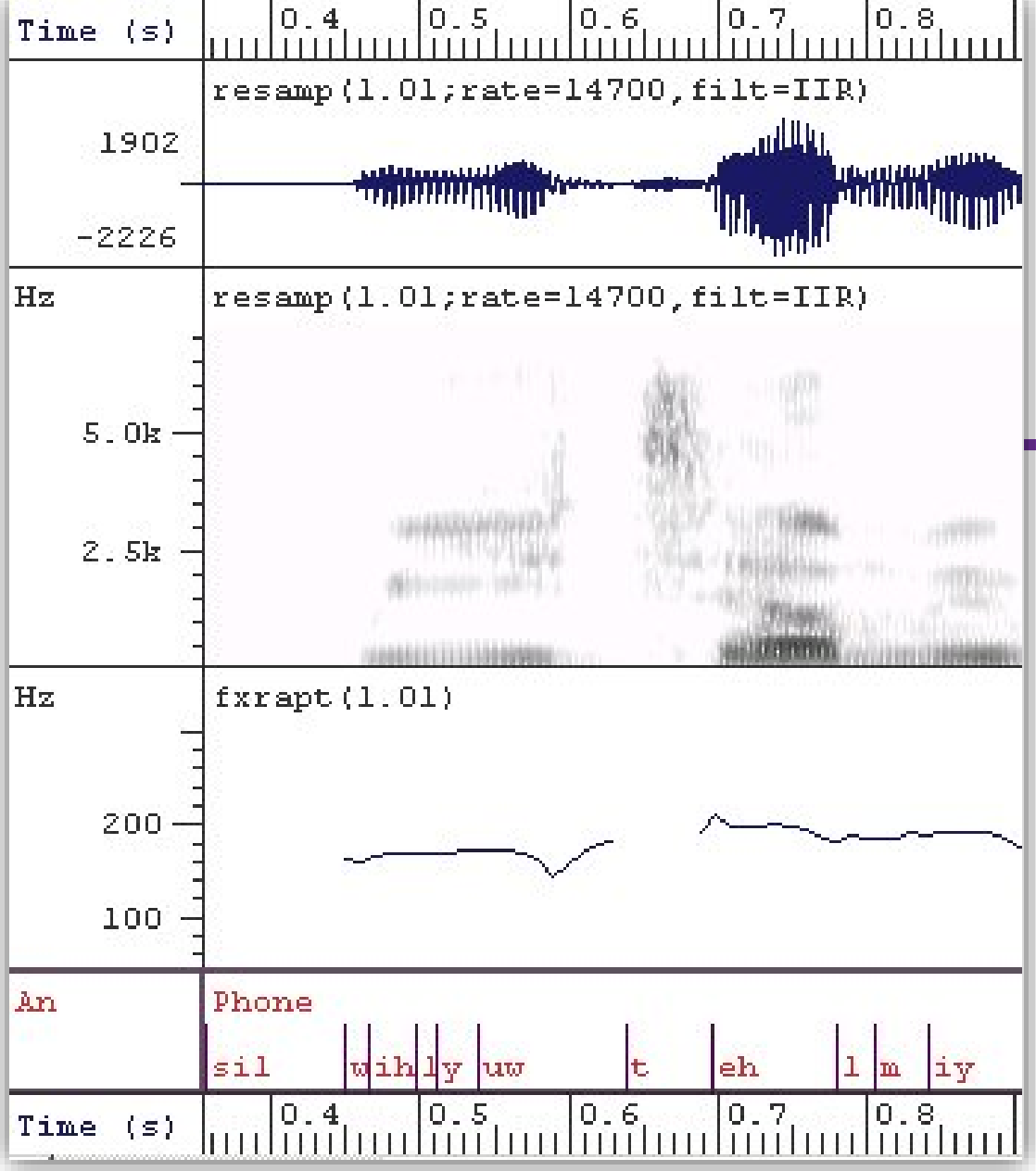
DEEP LEARNING ON PYTHON



OBJECTIVE

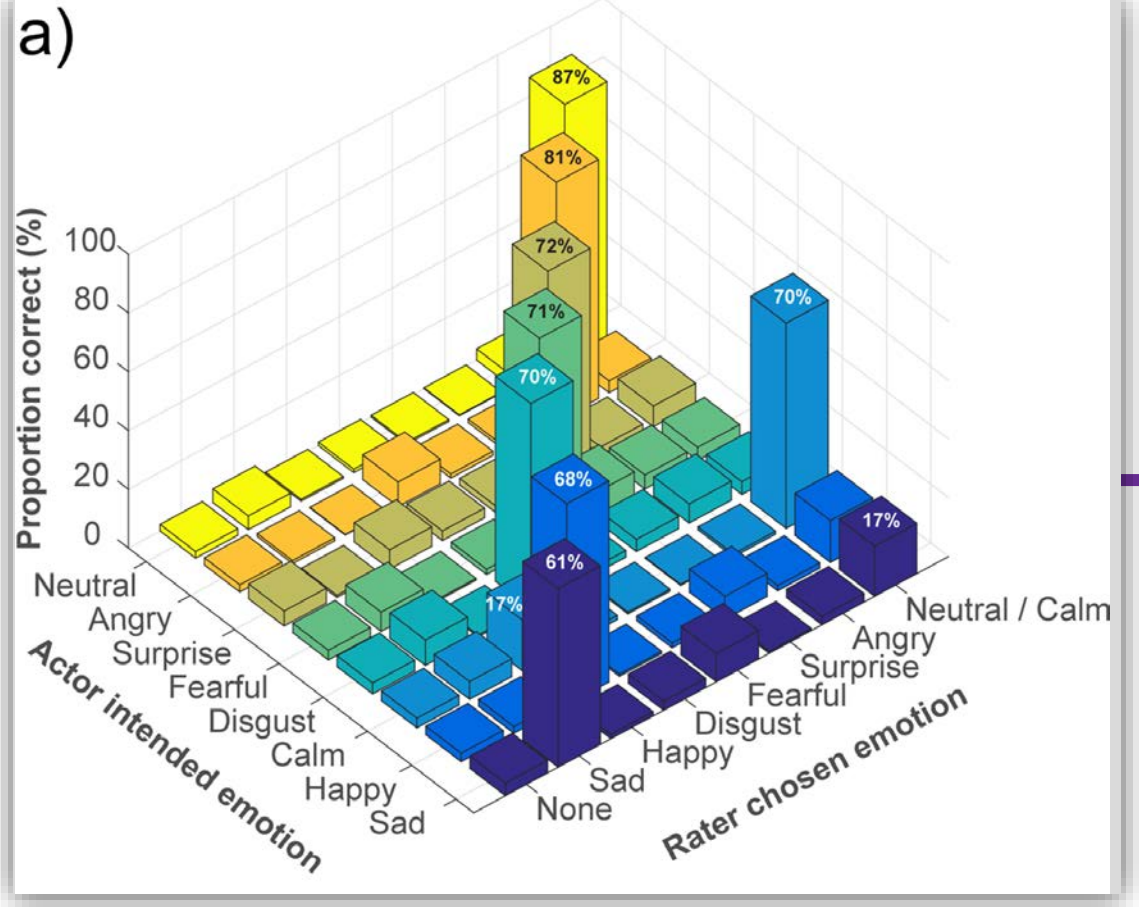
To create an Emotion recognition engine to identify mood of a person through their speech recording.

DATABASES USED



SAVEE

Surrey Audio-Visual Expressed Emotion (SAVEE) database has been recorded as a pre-requisite for the development of an automatic emotion recognition system. The database consists of recordings from 4 male actors in 7 different emotions, 480 British English utterances in total. The sentences were chosen from the standard TIMIT corpus and phonetically-balanced for each emotion. The data were recorded in a visual media lab with high quality audio-visual equipment, processed and labeled.

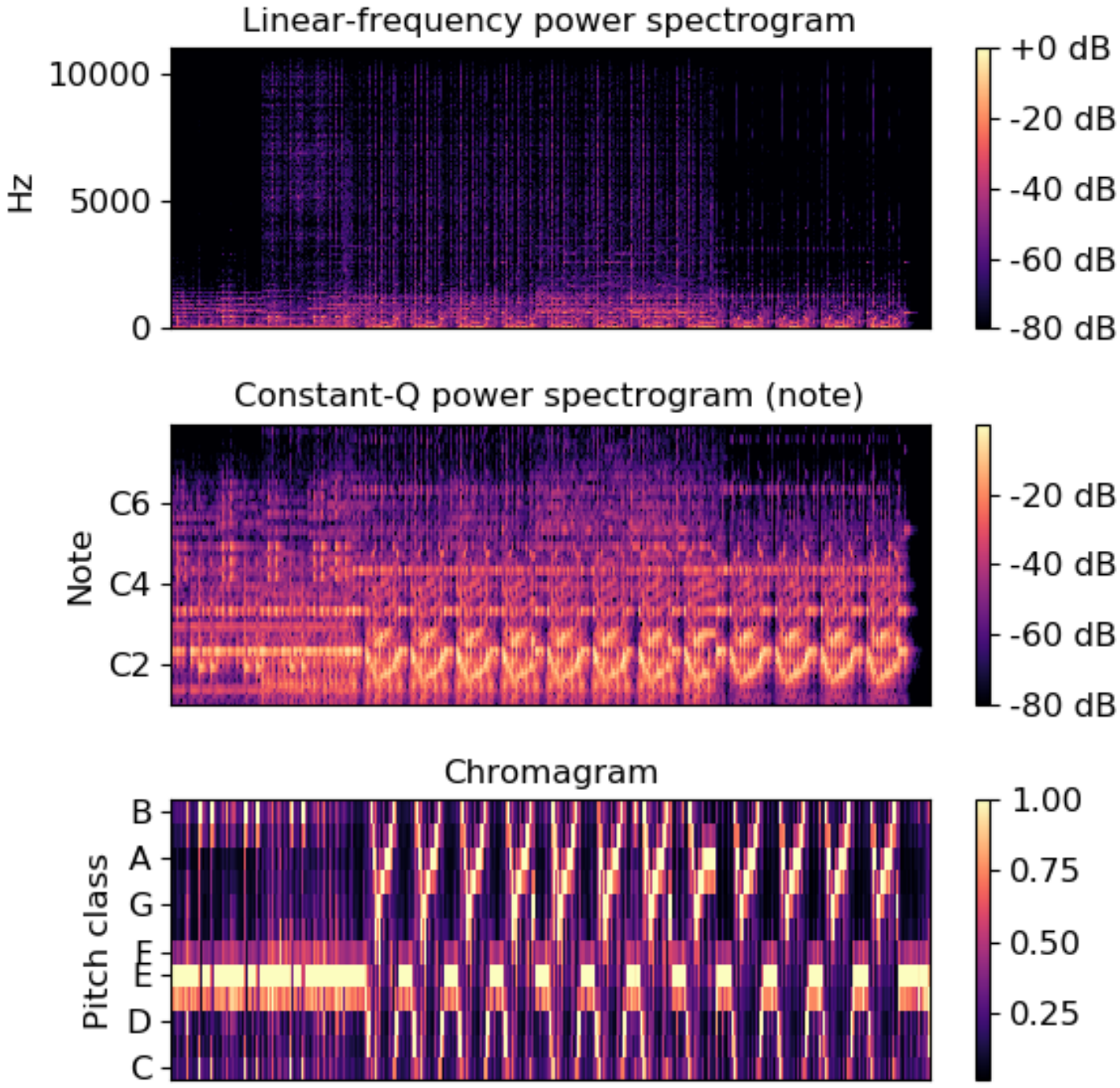


RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. The RAVDESS is a validated multimodal database of emotional speech and song. The database is gender balanced consisting of 24 professional actors, vocalizing lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity, with an additional neutral expression. All conditions are available in face-and-voice, face-only, and voice-only formats. The set of 7356 recordings were each rated 10 times on emotional validity, intensity, and genuineness.

FEATURES EXTRACTED

Using the LibROSA Python library. It provides the building blocks necessary to create music information retrieval systems.

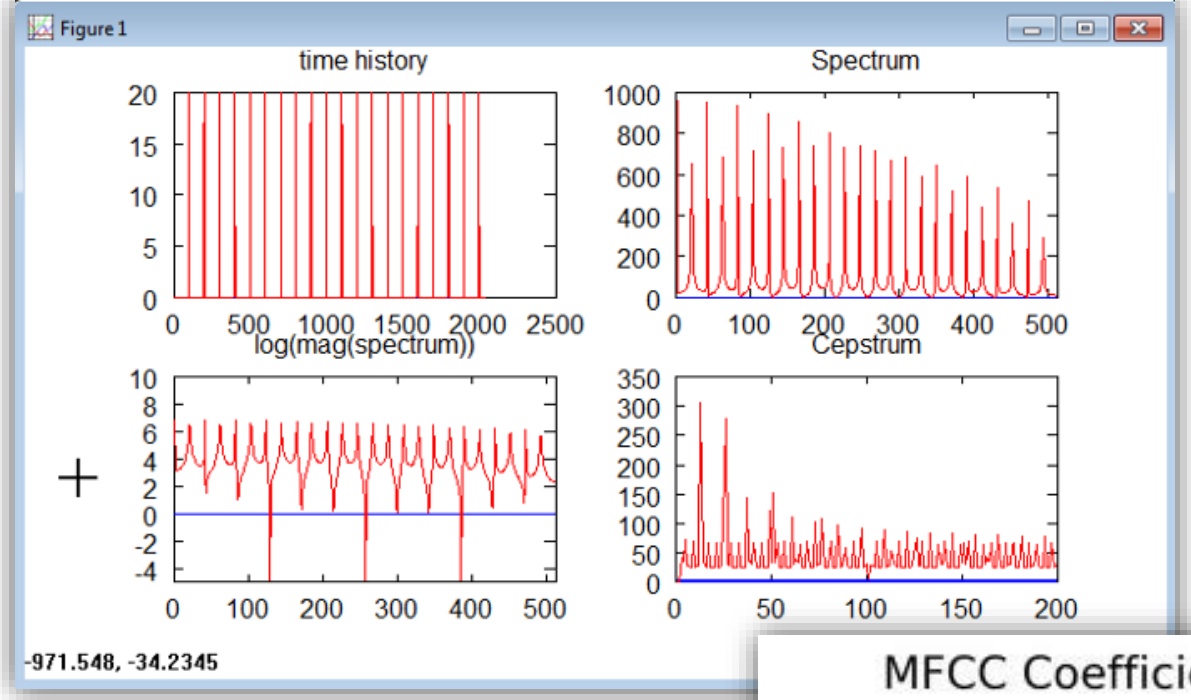


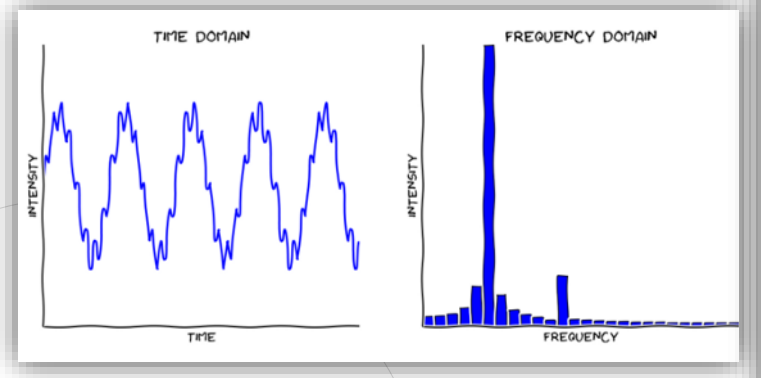
KUNAL GEHLOT

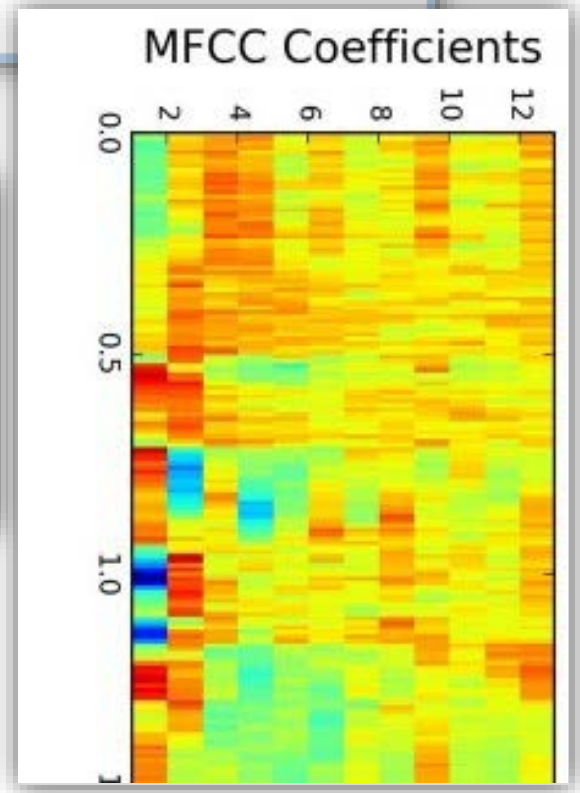
MEL-FREQUENCY CEPSTRAL COEFFICIENTS (MFCCS)

Ever heard the word cepstral before? Probably not. It's spectral with the spec reversed! Why though? For a very basic understanding, cepstrum is the information of rate of change in spectral bands. In the conventional analysis of time signals, any periodic component (for eg, echoes) shows up as sharp peaks in the corresponding frequency spectrum (i.e., Fourier spectrum. This is obtained by applying a Fourier transform on the time signal).

On taking the log of the magnitude of this Fourier spectrum, and then again taking the spectrum of this log by a cosine transformation (I know it sounds complicated, but bear with me please!), we observe a peak wherever there is a periodic element in the original time signal. Since we apply a transform on the frequency spectrum itself, the resulting spectrum is neither in the frequency domain nor in the time domain and hence Bogert et al. decided to call it the *quefrency domain*. And this spectrum of the log of the spectrum of the time signal was named *cepstrum* (ta-da!).

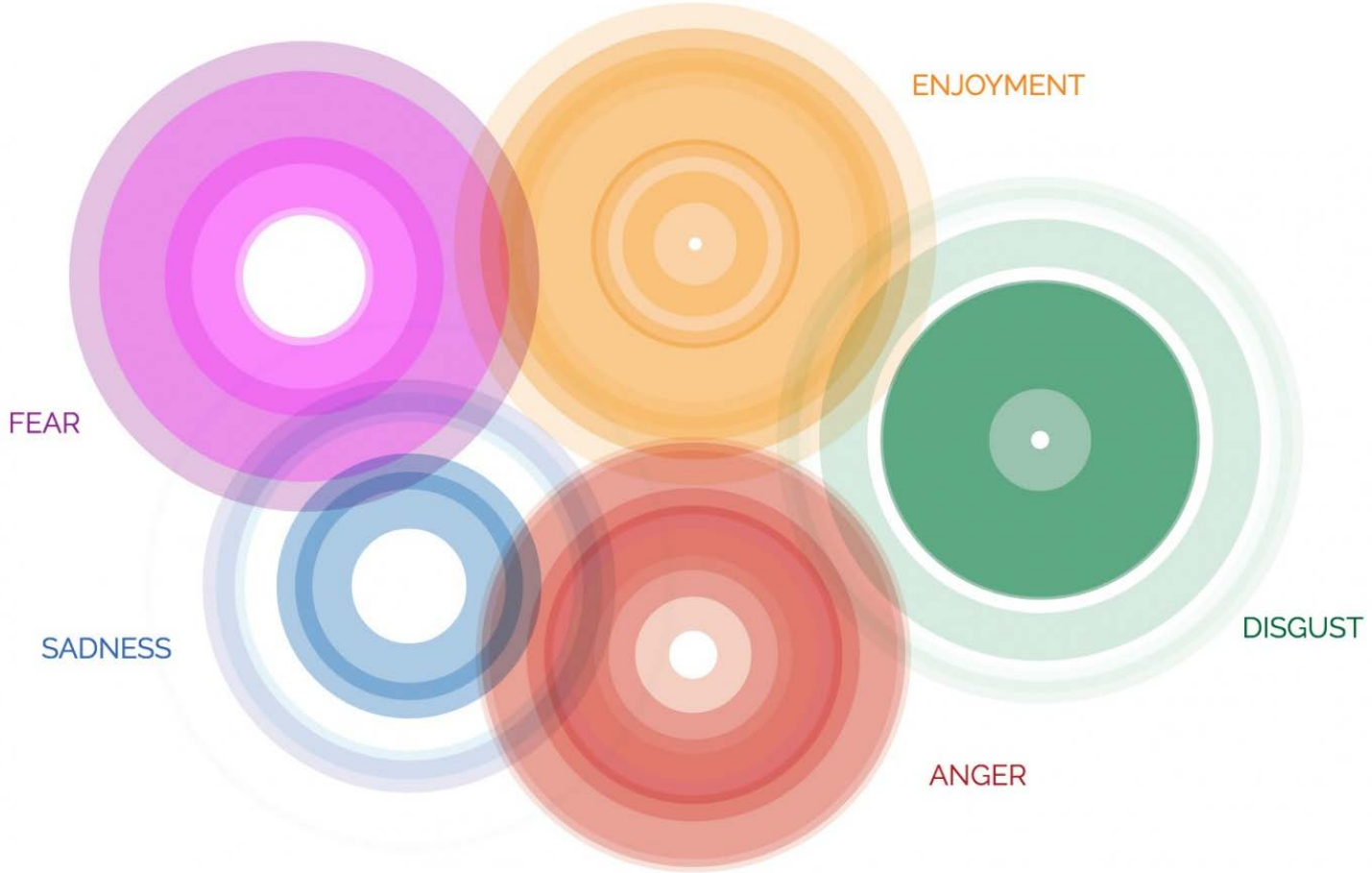






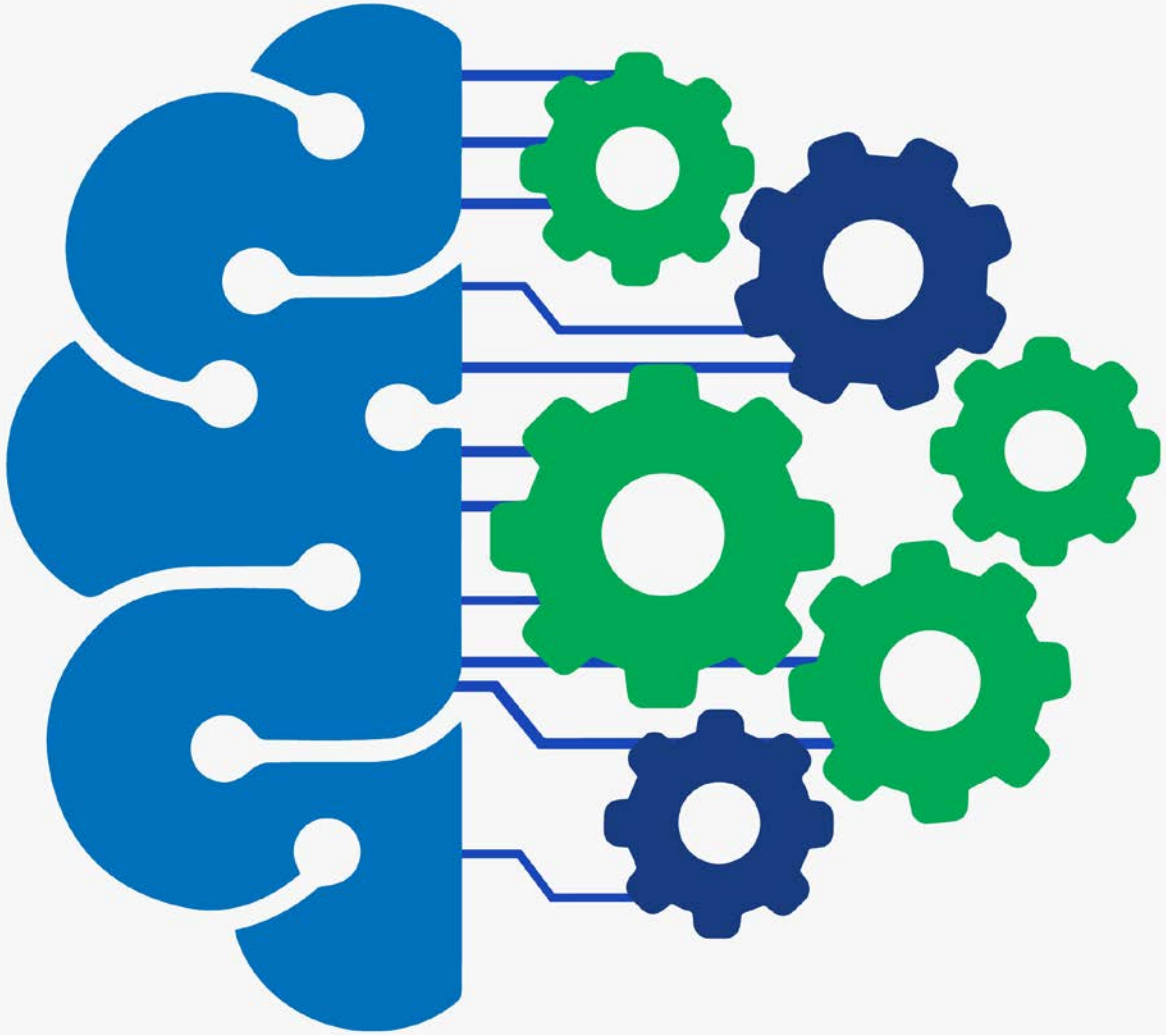
NUMBER OF EMOTIONS

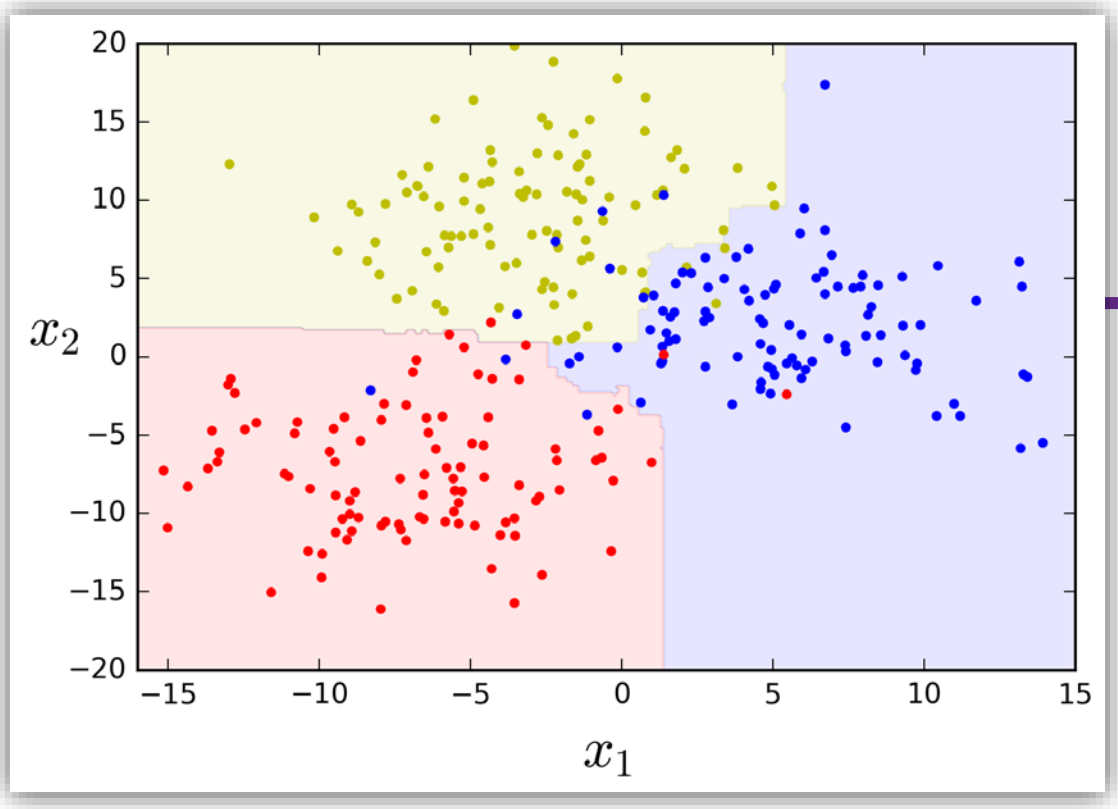
The output, according to the trained database was classified into 7 emotions, i.e. : **Anger, Fear, Disgust, Happy, Sad, Surprised, Neutral**



MODELS USED

We took two approaches, one of Machine Learning and other of Deep Learning with model and accuracy mentioned.





MACHINE LEARNING

Random Forest

Accuracy: 38.89%

KNeighborsClassifier(9 neighbors)

Accuracy: 35.93%

Concluding that machine learning would not work at all, I took the Deep Learning approach.

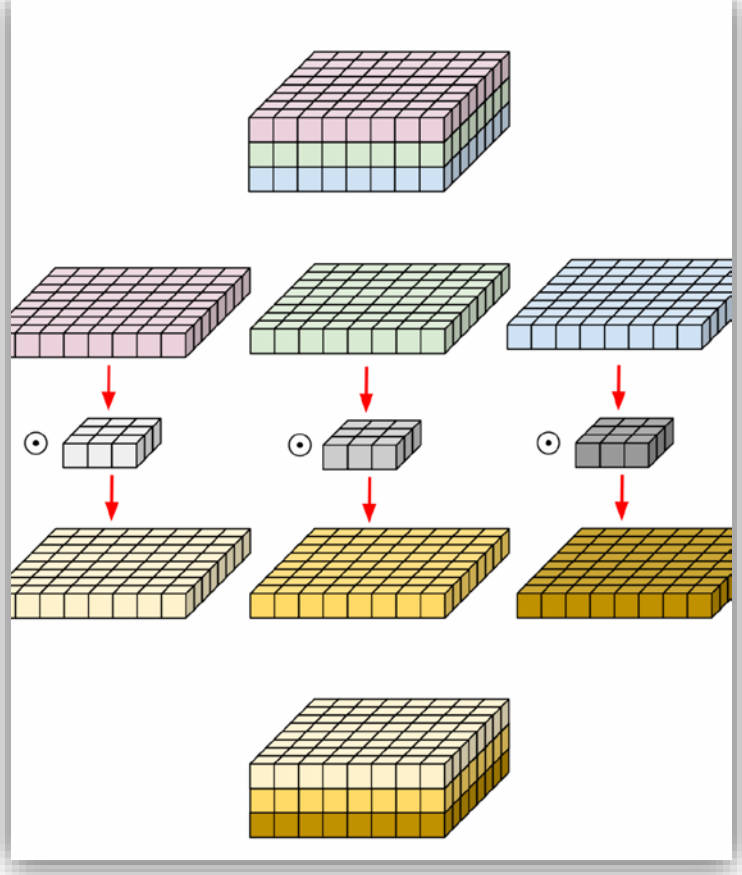
DEEP LEARNING

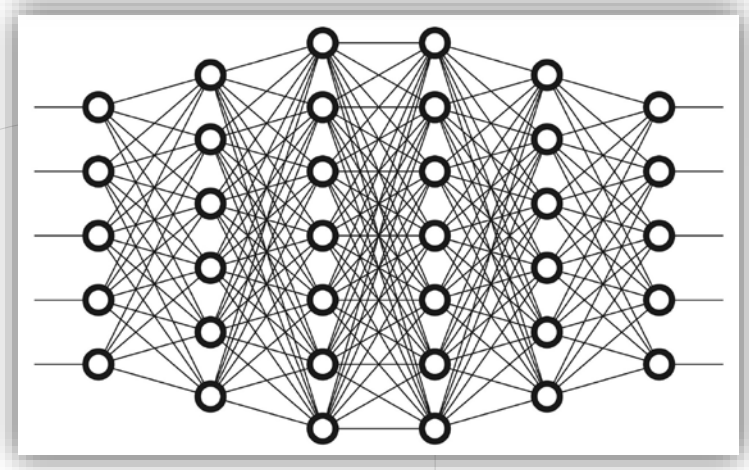
Using the GPU accelerated Keras, running on top of Tensorflow, I used 2-dimentional Convolutional Neural Network (Conv2D).

Conv2D

Accuracy: 86.43%

Thus making the Conv2D the most useful Model to predict emotion.





Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 27, 204, 8)	1360
batch_normalization_1 (Batch	(None, 27, 204, 8)	32
activation_1 (Activation)	(None, 27, 204, 8)	0
conv2d_2 (Conv2D)	(None, 15, 192, 8)	10824
batch_normalization_2 (Batch	(None, 15, 192, 8)	32
activation_2 (Activation)	(None, 15, 192, 8)	0
max_pooling2d_1 (MaxPooling2	(None, 7, 192, 8)	0
conv2d_3 (Conv2D)	(None, 5, 190, 8)	584
batch_normalization_3 (Batch	(None, 5, 190, 8)	32
activation_3 (Activation)	(None, 5, 190, 8)	0
conv2d_4 (Conv2D)	(None, 5, 190, 8)	72
batch_normalization_4 (Batch	(None, 5, 190, 8)	32
activation_4 (Activation)	(None, 5, 190, 8)	0
max_pooling2d_2 (MaxPooling2	(None, 2, 190, 8)	0
flatten_1 (Flatten)	(None, 3040)	0
dense_1 (Dense)	(None, 64)	194624
batch_normalization_5 (Batch	(None, 64)	256
activation_5 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455

Total params: 208,303
Trainable params: 208,111
Non-trainable params: 192

TESTING

Two separate programs are made, one to load the saved models and audio file which is recorded from another program.

```
model = Sequential()
```

```
model.add(Conv2D(8, (13, 13), input_shape=(in_shape[0], in_shape[1], 1)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(Conv2D(8, (13, 13)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 1)))
model.add(Conv2D(8, (3, 3)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(Conv2D(8, (1, 1)))
model.add(BatchNormalization(axis=-1))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 1)))
model.add(Flatten())
model.add(Dense(64))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(numLabels, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam',
                        metrics=['accuracy'])
print(model.summary(), file=sys.stderr)
```

```
•••
```

```
model.load_weights('VoiceEmo.h5')
```

```
file = 'output1.wav'
```

```
X, sample_rate = librosa.load(file, res_type='kaiser_best',duration=2.5,sr=22050*2,offset=0.5)
sample_rate = np.array(sample_rate)
mfccs = librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=39)
feature = mfccs
```

```
feature = feature.reshape(39, 216, 1)
```

```
classLabels[np.argmax(model.predict(np.array([feature])))]
```

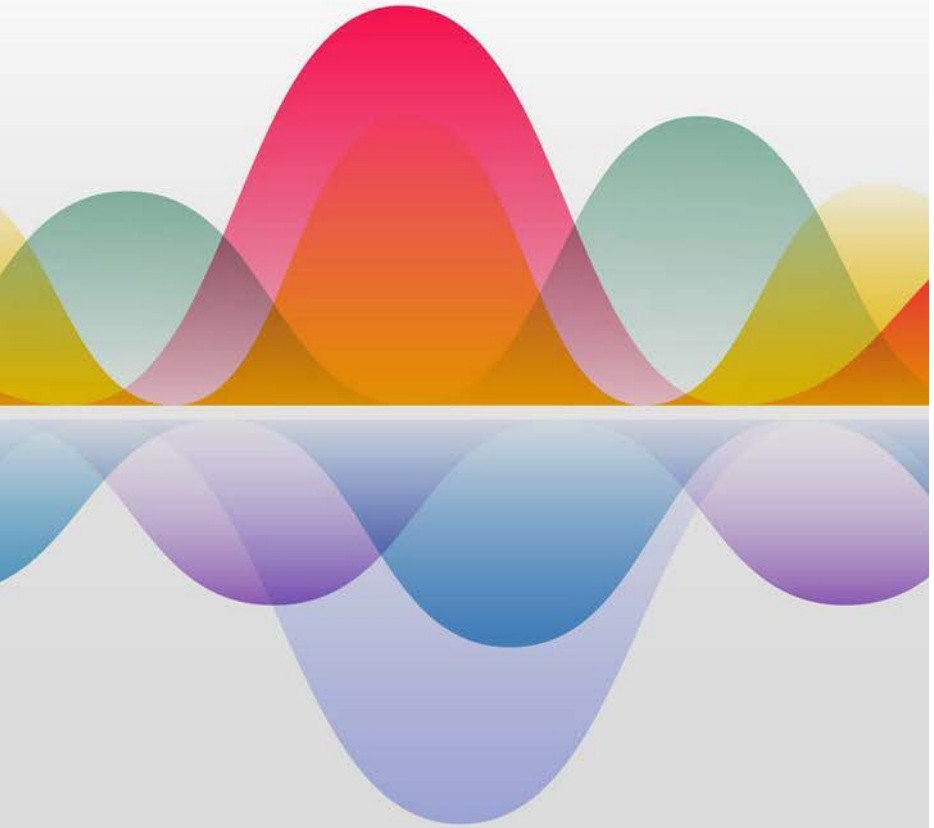
```
'Neutral'
```

```
import pyaudio
import wave

CHUNK = 1024
FORMAT = pyaudio.paInt16 #paInt8
CHANNELS = 2
RATE = 44100 #sample rate
RECORD_SECONDS = 4
WAVE_OUTPUT_FILENAME = "output1.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK) #buffer

print("* recording")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data) # 2 bytes(16 bits) per channel

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

```
* recording
* done recording
```

References

- https://github.com/harry-7/speech-emotion-recognition
- https://github.com/MITESHPUTHRANNEU/Speech-Emotion-Analyzer
- https://github.com/eesungkim/Speech_Emotion_Recognition_DNN-ELM
- https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
- https://www.researchgate.net/publication/326723032_EMOTION_DETECTION_FROM_VOICE_BASED_CLASSIFIED_FRAME-ENERGY_SIGNAL_USING_K-MEANS_CLUSTERING?enrichId=rgreq-38fe84f6038b43556c4b23501430a538-XXX&enrichSource=Y292ZXJQYWdlOzMyNjcyMzAzMjtBUzo2NTY2OTk2MTg3ODMyMzJAMTUzMzU4MDc5ODA0OA%3D%3D&el=1_x_2&_esc=publicationCoverPdf
- https://github.com/RayanWang/Speech_emotion_recognition_BLSTM
- https://www.researchgate.net/publication/331399972_Improving_Speech_Emotion_Recognition_with_Unsupervised_Representation_Learning_on_Unlabeled_Speech
- https://www.researchgate.net/publication/328681470_EmoVoice_Finding_My_Mood_from_My_Voice_Signal
- https://www.researchgate.net/publication/292378134_Voice_and_Mood



THANK YOU

gehlotkunal@outlook.com

www.linkedin.com/in/GehlotK