A Project Report

on

SPEECH EMOTION RECOGNITION

Submitted to Manipal University, Jaipur

Towards the partial fulfilment for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In Data Science

2022-2023

By

Zeel Tanna

209309048

MANIPAL UNIVERSITY
JAIPUR

Under the Guidance of

Dr. Shweta Sharma

Department of Information Technology

School of Information Technology

Manipal University Jaipur

Jaipur, Rajasthan

(Nov-2022)

CERTIFICATE

Date: 20/11/2022

This is to certify that the project work entitled **"Speech Emotion Recognition"** submitted by **Zeel Tanna(209309048)** in fulfilment for the requirements of the award of Bachelor of Technology Degree in Data Science at Manipal University Jaipur is an authentic work carried out by them under my supervision and guidance. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Dr.Shweta Sharma
Associate Professor
& Project Guide
Department of IT
Manipal University Jaipur

Dr. Pankaj Vyas

HOD
Department of IT
Manipal University Jaipur

ABSTRACT

- Speech is human vocal communication using language. Human speech has evolved into various forms of communication like facial expressions, gestures, postures, speech etc. The form of communication depends on the context of interaction. It is also often accompanied by various sorts of reactions. All forms of communications carry information at two levels, the message itself and the underlying emotional state.

- Speech often carries information not only about the lexical content, but also about the age, gender, signature and the emotional state of the speaker. Speech in different emotional states is accompanied by distinct changes in the production mechanism. Humans have the natural ability to grasp the underlying emotional state as well as the lexical content. For example, the perception of angry speech is that there is increase in voice intensity, raised pitch and faster speaking rate. Whereas the perception of calm speech is that the voice intensity is quiet, pitch is normal, and the person speaks anxiety free.

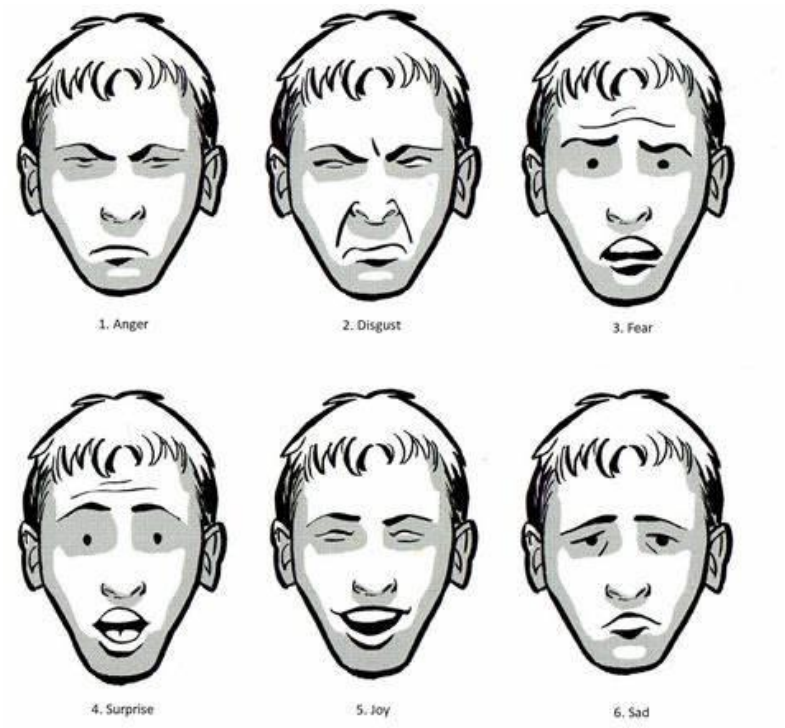

Figure 1: Facial Expressions

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Speech Emotion Recognition deals with the part of research in which the machine or the computer is able to detect and recognize emotions and lexical content from the speech just like humans can. The objective is to derive an emotion specific feature representation by focusing on the deviations in the components of speech production mechanism.

The Speech Emotion Recognition System has a three-layered architecture approach:

(i) A speech processing system that extracts some appropriate characteristics from the input such as input, pitch, energy etc.

(ii) These characteristics should be summarized into a reduced set of features with the help of feature extraction.

(iii) A classifier that learns in a supervised manner with sample data on how to associate with the features related to the emotions.

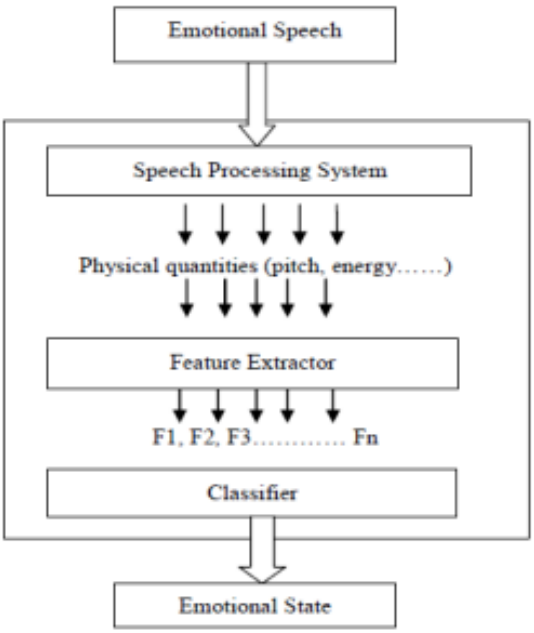

Figure 2

1.1 Motivation

Speech is complicated due to the fact that vocal expressions are evolving right as we speak. The vocal expression database is so extensive as a matter of fact that researchers still debate the extent to which verbal and non-verbal aspects can be neatly separated hence this serves as a motivation to further work on improving the accuracy to which vocal expressions can be neatly classified and categorized into a simpler and recognizable form.

1.2 Objective

- In the past decade a lot of research and effort has been put into Automatic Speech Emotion Recognition. The primary objective of Speech Emotion Recognition is to improve the man-machine interaction.

2. Literature Review

Natural Language processing refers to the branch of Computer Science that deals with giving computers the ability to interpret and decode text and spoken words the same way human beings can. Human language is filed with ambiguities that make it incredibly difficult for a software to decode and interpret accurately.

Several NLP tasks break down the human text and the vocal data into pieces that help the computer interpret. One of these tasks is Speech Recognition. Speech Recognition also called speech-to-text, is the task of converting the vocal data into textual data.

Speech Recognition is required for applications that follow voice commands or answer spoken questions. What makes Speech Recognition challenging is that the machine should be able to keep up and recognise the different accents, different speed of speaking, emphasis and intonation of words.

My project deals with Speech Emotion Recognition and its analysis. The programming language used for analysis and model formation is Python. For dataset purpose I have used Toronto Emotion Speech Set from Kaggle. The data set contains 200 target words that were spoken by two actresses and recordings were made of the set portraying seven emotions(anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral. There are 2800 audio files in total and the audio file format used is WAV format.

RNN fails to do its job when used in the given data set, whereas LSTM perfectly overcomes the drawbacks of RNN and works perfectly in sync with the data set. Hence, I have used the LSTM model to handle the dataset.

3. Design and Implementation

- Neural Networks are one of the most popular Machine Learning and Deep Learning algorithms. They outperform the other algorithms in both speed as well as accuracy. A Neural Network consists of different layers connected to each other. It works similar to that of a human brain. It uses large volumes of data and uses complex algorithms to train a neural network.

- Recurrent Neural Network of RNN is the type of neural network that works on the principle of saving the output of a particular layer and feeding the same output back into the network as the input to predict the output of the layer. Sequential information that is stored in the memory of the RNNs is used for predictions. The idea of using RNN instead of Traditional Neural Network, it is assumed that every input and output aren't dependent on each other. Hence using traditional neural network is a bad idea. The Recurrent Neural Network is used for Speech Recognition, Voice Recognition, Time Series Production and Natural Language Processing.

- Long Term Short-Term Memory Network is a special kind of RNN that is capable of learning long dependencies by remembering information for longer periods of time this helps combat the Vanishing Gradient issue of the RNN model. The LSTM has a chain like structure but instead of having a single neural network layer it has four interacting layers.

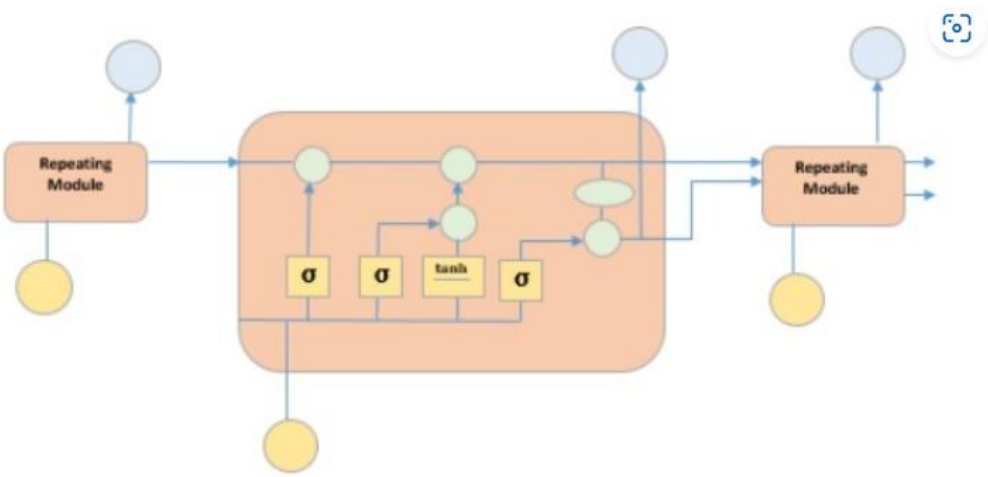

Figure 3

- The figure above depicts the four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and point wise operators in blue circles. The module has a cell state and three gates which provides the with the power to selectively learn, unlearn or retain information from each of the units.

Step 1: Loading and pre-process data

1.1 Load the data:

The Toronto Emotion Speech Set dataset used for pre-processing is upload in the form of zip file via Kaggle itself. Contents of the dataset include a set of 200 words that were spoken by two actresses (aged 26 & 64 years) and recordings were made of the set portraying each seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness and neutral). There are about 2800 data points in total.

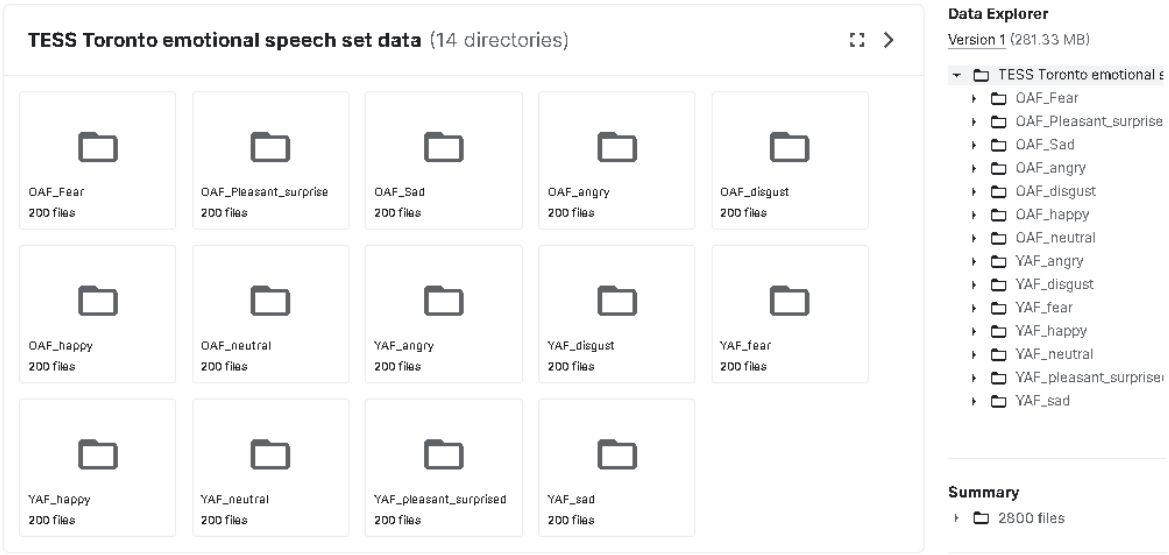

Figure 4

1.2 Pre-process the data:

Data Pre-processing is the data mining technique that turns the raw data gathered from diverse sources into cleaner information that's more suitable for work. Raw data can have missing or inconsistent values as well as a lot of redundant information. If we don't take care of these issues then the final output would be plagued with faulty insights. This is especially true for more sensitive analysis that can be more affected by small mistakes like when its used in new fields where minimal variations in raw data can lead to wrong assumptions. Hence the information needs to be organized, sorted, and merged.

Pre-processing of the uploaded data is done via following steps:

- **(i)** The dataset consists of multiple files having filenames that needed to be trimmed down to just display them as the 7 emotion names.
- **(ii)** The new names of the files were appended into the original dataset.
- **(iii)** Next step was to display the paths and the labels for the first 5 items of the appended data set.
- **(iv)** Before moving into exploratory analysis, the dataset is to be put into a data frame after creating a new data frame.

```
#Splitting the file names from the various files using labels
paths = []
labels = []
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
    if len(paths) == 2800:
        break
print('Dataset is Loaded')
```

+ Code + Markdown

```
#Paths of first 5 items
paths[:5]
```

```
#Labels of first 5 items
labels[:5]
```

```
#Create a Dataframe
df = pd.DataFrame()
df['speech'] = paths #input 1
df['label'] = labels #input 2
df.head()
```

Figure 5

Step 2: Exploratory Data Analysis: -

1.1 . Customizing the data:

After loading and customizing the data set, we perform Exploratory Analysis so as to identify the outliers and features of the data:

- The data set contains audio files hence we create and use the Waveplot function to form the basis for deeper analysis of sound data.
- We also use Spectrogram function for visualising the signals of the audio into images which is preferable while performing data analysis.

```
#Displaying the Waveplot
def waveplot(data, sr, emotion): #defining the waveplot
    plt.figure(figsize=(10,4)) #define the size of the figure
    plt.title(emotion, size=20) #giving title to the figure as emotion
    librosa.display.waveshow(data, sr=sr) #entering the data into figure
    plt.show()

#Displaying the Spectogram
def spectrogram(data, sr, emotion):
    x = librosa.stft(data)
    xdb = librosa.amplitude_to_db(abs(x))
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(xdb, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar()
```

Figure 6

- We use the Waveplot and the Spectrogram functions to project the graphs of 7 different emotions as well as their sample audio files. (Fear, Anger, Disgust, etc)

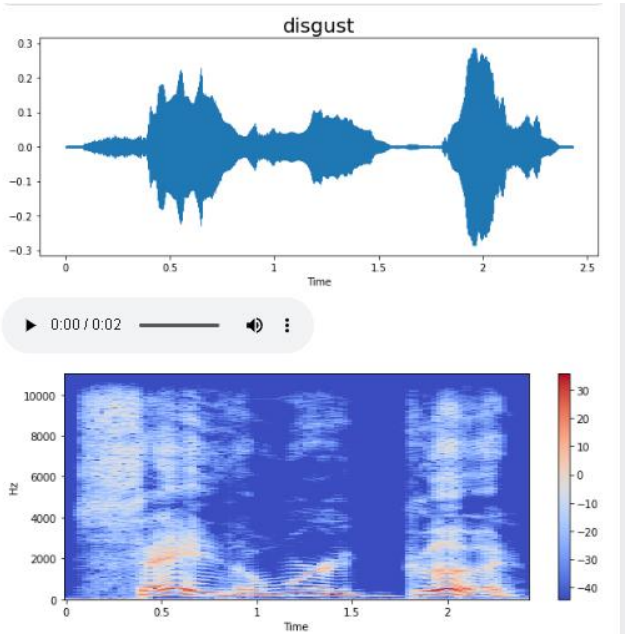

Figure 7

Step 3: Feature Extraction: -

1.1 Extracting Features using MFCC method:

The MFCC method is used to develop the features from the audio signals which can be used to detect the phones from the speech:

- Using raw signal as input to our model will cause lots of problems due to presence of noise in the audio signal thus I extracted features from the audio signal to use as input to the base model to produce much better results.

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration=3, offset=0.5) #loading the dataset and setti
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40).T, axis=0) #extracting the
    return mfcc
```

+ Code + Markdown

```
extract_mfcc(df['speech'][0])
```

```
array([-287.13037  ,   87.756935 ,   -4.139178 ,   24.081968 ,
        -16.696726 ,   12.9706335,   10.522443 ,   -1.1463207,
         -0.7333776,   12.855535 ,  -19.147291 ,   -6.418062 ,
          4.9657674,   -2.6571147,  -10.655446 ,    4.9578824,
        -14.555861 ,   15.37587  ,   18.444933 ,   23.878317 ,
         31.495148 ,   17.326372 ,   -4.764838 ,    1.7432449,
        -12.009848 ,    7.34574  ,   -3.205127 ,   -7.1714525,
        -11.410635 ,   -2.0019934,   -5.6109643,    4.532194 ,
        -11.396626 ,   -8.892363 ,   -3.7391381,    4.8819685,
         -1.5599906,    2.4654472,   11.599151 ,   11.042193 ],
      dtype=float32)
```

Figure 8

- We then split the data that we need as input and model it as a three-dimensional data set so as to it is compatible with the LSTM model.

```
X = [x for x in X_mfcc]
X = np.array(X)
X.shape
```

```
(2800, 40)
```

```
## input split
X = np.expand_dims(X, -1)
X.shape
```

Figure 9

Step 4: Model Creation and Training: -

1.1 LSTM Model:

The LSTM model is basically the advanced version of RNN model. RNN models face the vanishing and exploding gradient problems, and since the LSTM model overcame these difficulties it's the best suited model for my Dataset. The model is an excellent choice for classifying the speech sample because it effectively exploits the temporal dependencies in the acoustic data.

- Importing libraries and creating the LSTM model using Keras. The model contains four layers respectively; the first layer being the LSTM layer, the second third and fourth layers being the Dense Layers.
- Activation functions used in the model are Sigmoid, RELU and SoftMax. Sigmoid and RELU are used to achieve nonlinear transformation of the data and SoftMax is used for Categorical classification.
- The model is then compiled, and the model contains an accuracy matrix so as to determine the accuracy of the model.

SIGMOID FUNCTION:

```
#Importing the libraries
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

#Creating the model
model = Sequential([
    LSTM(256, return_sequences=False, input_shape=(40,1)), #first layer of the model
    Dropout(0.2),
    Dense(128, activation='sigmoid'), #second layer of the model
    Dropout(0.2),
    Dense(64, activation='sigmoid'), #third layer of the model
    Dropout(0.2),
    Dense(7, activation='softmax') #fourth layer of the model #
])

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

Figure 10

- Now we load the dataset into the model and train it setting the following parameters:
  - Validation Split = 20%
  - Number of Epochs = 50
  - Batch Size = 64
- Same procedure is followed for the RELU function

```
history = model.fit(X, y, validation_split=0.20, epochs=50, batch_size=64)
```

```
Epoch 1/50
35/35 [==============================] - 4s 82ms/step - loss: 0.0592 - accuracy: 0.9857 - val_loss: 2.3127 - val_accuracy: 0.5048
Epoch 2/50
35/35 [==============================] - 3s 80ms/step - loss: 0.0511 - accuracy: 0.9902 - val_loss: 2.5756 - val_accuracy: 0.4786
Epoch 3/50
35/35 [==============================] - 3s 79ms/step - loss: 0.0560 - accuracy: 0.9871 - val_loss: 1.7946 - val_accuracy: 0.5905
Epoch 4/50
35/35 [==============================] - 3s 80ms/step - loss: 0.0373 - accuracy: 0.9924 - val_loss: 2.8124 - val_accuracy: 0.4952
Epoch 5/50
35/35 [==============================] - 3s 82ms/step - loss: 0.0389 - accuracy: 0.9929 - val_loss: 3.2196 - val_accuracy: 0.3595
Epoch 6/50
35/35 [==============================] - 3s 82ms/step - loss: 0.0342 - accuracy: 0.9920 - val_loss: 4.0610 - val_accuracy: 0.2714
Epoch 7/50
35/35 [==============================] - 3s 79ms/step - loss: 0.0305 - accuracy: 0.9942 - val_loss: 3.3071 - val_accuracy: 0.3714
Epoch 8/50
35/35 [==============================] - 3s 81ms/step - loss: 0.0237 - accuracy: 0.9960 - val_loss: 3.3545 - val_accuracy: 0.3810
Epoch 9/50
35/35 [==============================] - 3s 96ms/step - loss: 0.0292 - accuracy: 0.9951 - val_loss: 3.0093 - val_accuracy: 0.4214
Epoch 10/50
35/35 [==============================] - 3s 81ms/step - loss: 0.0280 - accuracy: 0.9942 - val_loss: 3.9294 - val_accuracy: 0.3786
Epoch 11/50
35/35 [==============================] - 3s 79ms/step - loss: 0.0269 - accuracy: 0.9946 - val_loss: 2.7505 - val_accuracy: 0.4929
Epoch 12/50
35/35 [==============================] - 3s 81ms/step - loss: 0.0470 - accuracy: 0.9884 - val_loss: 2.6203 - val_accuracy: 0.5667
Epoch 13/50
35/35 [==============================] - 3s 80ms/step - loss: 0.0525 - accuracy: 0.9875 - val_loss: 3.0513 - val_accuracy: 0.5452
Epoch 14/50
```

Figure 11

#RELU FUNCTION:

```
#Importing the libraries
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout

#Creating the model
model = Sequential([
    LSTM(256, return_sequences=False, input_shape=(40,1)), #first layer of the model
    Dropout(0.2),
    Dense(128, activation='relu'), #second layer of the model
    Dropout(0.2),
    Dense(64, activation='relu'), #third layer of the model
    Dropout(0.2),
    Dense(7, activation='softmax') #fourth layer of the model #
])

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

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 256)               264192
_________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_19 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_20 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_20 (Dense)             (None, 7)                 455
=================================================================
Total params: 305,799
Trainable params: 305,799
Non-trainable params: 0
_________________________________________________________________
```

Figure 12

```
Epoch 13/50
35/35 [==============================] - 4s 101ms/step - loss: 0.0710 - accuracy: 0.9786 - val_loss: 2.8826 - val_accuracy: 0.3875
Epoch 14/50
35/35 [==============================] - 4s 101ms/step - loss: 0.0757 - accuracy: 0.9750 - val_loss: 1.6044 - val_accuracy: 0.6000
Epoch 15/50
35/35 [==============================] - 4s 101ms/step - loss: 0.0661 - accuracy: 0.9817 - val_loss: 2.8485 - val_accuracy: 0.4036
Epoch 16/50
35/35 [==============================] - 4s 100ms/step - loss: 0.0325 - accuracy: 0.9915 - val_loss: 2.3683 - val_accuracy: 0.5518
Epoch 17/50
35/35 [==============================] - 4s 124ms/step - loss: 0.0327 - accuracy: 0.9920 - val_loss: 3.3539 - val_accuracy: 0.3893
Epoch 18/50
35/35 [==============================] - 4s 103ms/step - loss: 0.0270 - accuracy: 0.9911 - val_loss: 4.6709 - val_accuracy: 0.3304
Epoch 19/50
35/35 [==============================] - 3s 98ms/step - loss: 0.0549 - accuracy: 0.9853 - val_loss: 3.1640 - val_accuracy: 0.5018
Epoch 20/50
35/35 [==============================] - 3s 98ms/step - loss: 0.0260 - accuracy: 0.9937 - val_loss: 3.3719 - val_accuracy: 0.3857
Epoch 21/50
35/35 [==============================] - 3s 100ms/step - loss: 0.0243 - accuracy: 0.9902 - val_loss: 3.5022 - val_accuracy: 0.4732
Epoch 22/50
35/35 [==============================] - 4s 102ms/step - loss: 0.0509 - accuracy: 0.9853 - val_loss: 2.6158 - val_accuracy: 0.5179
Epoch 23/50
35/35 [==============================] - 4s 102ms/step - loss: 0.0215 - accuracy: 0.9920 - val_loss: 3.5363 - val_accuracy: 0.4286
Epoch 24/50
35/35 [==============================] - 4s 102ms/step - loss: 0.0204 - accuracy: 0.9937 - val_loss: 2.8550 - val_accuracy: 0.4482
Epoch 25/50
35/35 [==============================] - 4s 126ms/step - loss: 0.0190 - accuracy: 0.9946 - val_loss: 3.0711 - val_accuracy: 0.4946
Epoch 26/50
35/35 [==============================] - 3s 100ms/step - loss: 0.0235 - accuracy: 0.9924 - val_loss: 2.6028 - val_accuracy: 0.4821
Epoch 27/50
35/35 [==============================] - 3s 100ms/step - loss: 0.0197 - accuracy: 0.9955 - val_loss: 4.3076 - val_accuracy: 0.4679
Epoch 28/50
35/35 [==============================] - 4s 105ms/step - loss: 0.0217 - accuracy: 0.9933 - val_loss: 4.9115 - val_accuracy: 0.3554
Epoch 29/50
35/35 [==============================] - 4s 102ms/step - loss: 0.0093 - accuracy: 0.9973 - val_loss: 2.9637 - val_accuracy: 0.6196
Epoch 30/50
35/35 [==============================] - 3s 99ms/step - loss: 0.0043 - accuracy: 0.9991 - val_loss: 4.1487 - val_accuracy: 0.4857
Epoch 31/50
35/35 [==============================] - 4s 101ms/step - loss: 0.0131 - accuracy: 0.9951 - val_loss: 3.4314 - val_accuracy: 0.5536
Epoch 32/50
35/35 [==============================] - 3s 98ms/step - loss: 0.0112 - accuracy: 0.9960 - val_loss: 4.0082 - val_accuracy: 0.5179
Epoch 33/50
35/35 [==============================] - 3s 98ms/step - loss: 0.0059 - accuracy: 0.9982 - val_loss: 4.0369 - val_accuracy: 0.5000
Epoch 34/50
```

Figure 13

4. RESULT ANALYSIS

- Using the Trial and Error Method of Training and Testing Data with the LSTM model we can observe that the model with the RELU function produces better results.
- Using the LSTM model, I achieved a peak accuracy of 61.96%.
- Through this project we can see how we can utilize machine learning as well as deep learning to obtain the underlying emotions from the speech audio data and get some insight into the human expressions of emotions through voice.

```
epochs = list(range(50))
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, label='train accuracy')
plt.plot(epochs, val_acc, label='val accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

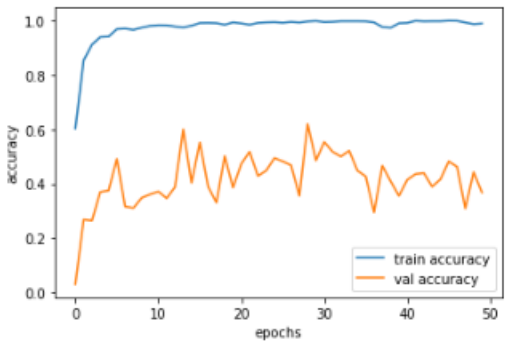

Figure 14

```
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(epochs, loss, label='train loss')
plt.plot(epochs, val_loss, label='val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

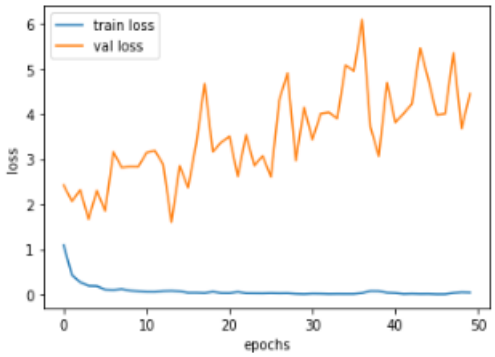

Figure 15

5. CONCLUSION

The overall system results show us how we can utilize Deep Learning and NLP to obtain the ultimate emotions from the audio data and get insights on human expressions of emotion through vocals.

Using python as a basis language for the project, the basic steps include loading the data, pre-processing them, using feature extraction for extracting important features and then validating the data for further analysis.

Further steps involve developing the LSTM model, which is an advanced RNN model, and training the model with the dataset that was adjusted to its compatibility. Finally, the model was trained with the dataset and produced accuracy results of 61.96%.

Although the model cannot be implemented since the accuracy being so low it still has multiple scope of improvements such as:

- Exploring acoustic properties of sound data to check their compatibility in domain of speech recognition
- Following Lexical feature-based approach towards SER to improve the accuracy of the system
- Applying applications for processing and accepting live inputs through multiple devices such as smartphones, laptops, tablets etc, and implementing them in day to day life.

References

[1] Speech Emotion Recognition (SER) through Machine Learning (analyticsinsight.net)

[2] Toronto emotional speech set (TESS) | Kaggle

[3] Speech Emotion Recognition [An applied project] - ExtruDesign

[4] Speech Emotion Recognition - an overview | ScienceDirect Topics

[5] K. Sreenivasa Rao, Shashidhar G. Koolagudi

Springer Science & Business Media, 07-Nov-2012 - Technology & Engineering