



# 50.038 Computation Data Science Project Report

Speech Emotion Recognition  
Term 6

*Team Members -*

Anshu Ghatе (1005268)

WangXilun (1004877)

Haozhi Lyu (1004888)

# Table of Contents

Objective :.....	3
Speech Emotion Recognition (SER) -.....	3
Use Cases -.....	3
Overview :.....	3
Dataset -.....	4
Graphical Representation of the data -.....	4
Data Visualisation -.....	4
Data Augmentation -.....	5
Feature Extraction.....	6
Data Pre-Processing.....	8
Problem and Algorithm Model.....	9
Evaluation Methodology.....	12
Results.....	14
Discussion.....	14
Model Limitation -.....	14
Future Direction -.....	15
Conclusion -.....	15

## Objective :

Using theories and concepts in class for speech emotion recognition. The goal is to classify emotion of each utterance in a conversation

### *Speech Emotion Recognition (SER) -*

SER is a field with the goal of identifying affective states and human emotions in speech. This is accomplished by examining a speaker's speech's varied characteristics, such as pitch, tone, and intensity, as these frequently represent underlying emotional states. The potential uses of SER include enhancing human-computer connection, identifying emotional disorders, and creating individualized goods and services. The key emotions present in the datasets used in SER studies often include Happy, Fear, Angry, Disgust, Surprise, Sad, or Neutral.

### *Use Cases -*

Speech emotion recognition has numerous potential applications across various industries. By building an AI model that analyzes recorded audio clips to predict the emotional state of a speaker, we can unlock valuable insights for marketing, healthcare, customer service, gaming, and more. Other fields, such as social media analysis and stress monitoring, can also benefit from this technology.

## Overview :

One method is to simulate voice emotion recognition using convolutional neural networks (CNNs). CNNs are a subclass of deep learning neural networks that are particularly effective at evaluating data having a grid-like architecture, such as spectrograms of audio signals or picture data. CNNs may be used to extract pertinent information from an utterance's spectrogram, such as the shape and distribution of different frequencies, for speech emotion identification. These features can then be fed into a classification model to predict the emotional state conveyed in the utterance. CNNs have been shown to be effective for speech emotion recognition, achieving high levels of accuracy in classifying different emotional states. By leveraging the power of deep learning, CNN-based models can learn to recognize complex patterns in the acoustic features of speech, making them a valuable tool for understanding human emotions in speech.

We successfully predicted the test data set with accuracy of 0.57.

## Dataset and Collection

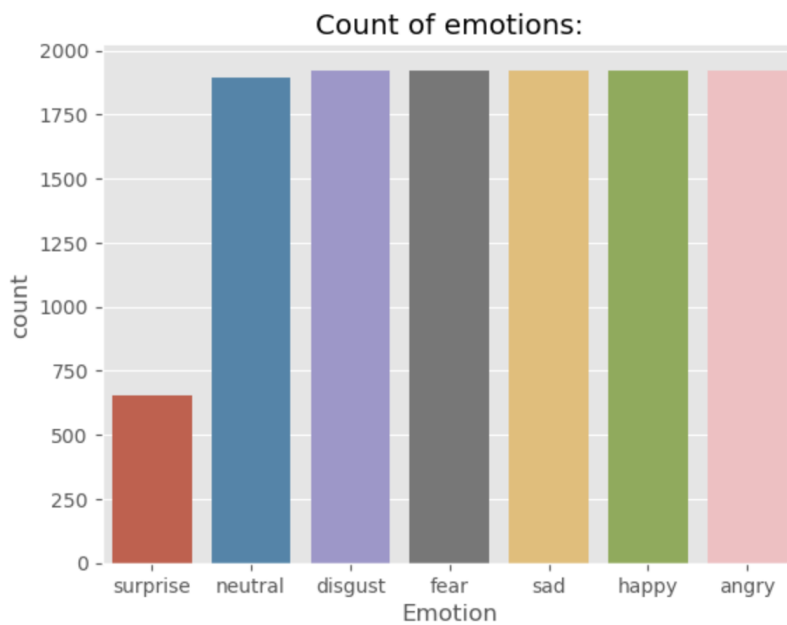
For our speech emotion recognition project, we utilized an audio dataset that was obtained from the Kaggle website.

The audio dataset we used contained speech recordings of several speakers conveying different emotional states, such as happiness, sadness, anger, and others. Here ,4 most popular datasets in English: Crema, Ravdess, Savee and Tess. Each of them contains audio in .wav format with some main labels.

### *Dataset -*

<https://www.kaggle.com/dmitrybabko/speech-emotion-recognition-en>

### *Graphical Representation of the data -*



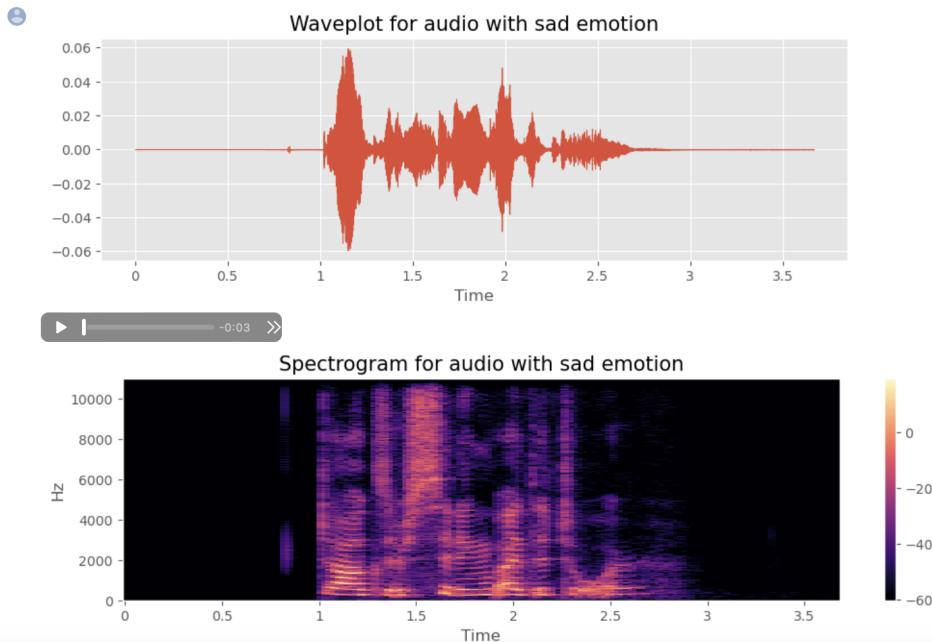
### *Data Visualisation -*

We tried to visualize the data for the different emotions using wave plot and spectrogram. The diagrams are shown below for sad emotion.

```

emotion='sad'
path = np.array(df.Path[df.Emotion==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

```



## *Data Augmentation -*

Using data augmentation approaches, more sound data can be produced, exposing the speech emotion detection model to a wider range of input variables. The training data can be made more diverse by using methods like noise injection, stretching, shifting, and pitching. This helps the model be better at differentiating between different kinds of sounds. These methods enable us to develop a more reliable and adaptable speech emotion recognition model.

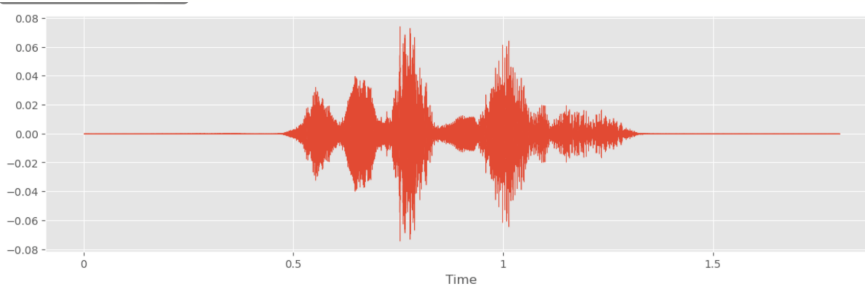
For instance, below is the code we used for data augmentation where we stretched the audio file -

```

def stretch(data, rate=0.8):
    """Stretching data with some rate."""
    return librosa.effects.time_stretch(data, rate=1.0/rate)

stretched_data = stretch(data, rate=0.5)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=stretched_data, sr=sampling_rate)
Audio(stretched_data, rate=sampling_rate)

```



## Feature Extraction

These are the steps used for extracting features from a signal. The process of extracting significant characteristics from a raw signal through analysis and transformation is known as feature extraction. Computing numerous signal parameters, such as frequency content, amplitude, and energy, which might provide information about the signal's attributes, is a frequent approach for feature extraction. These characteristics can be computed using mathematical operations and can be represented as a set of values that describe the signal. Here are the steps involved in feature extraction:

- Zero Crossing Rate: This is the frequency at which the signal changes sign inside a given frame. It can provide details about the signal's pitch and if periodicity is present.
- Energy : This is the squared sum of the signal values normalized by the frame duration. It can reveal details about the signal's total amplitude.
- Entropy of Energy: This measures the degree of randomness or uniformity in the energy distribution of sub-frames. It can provide information about the presence of sudden changes or transients in the signal.
- Spectral Centroid: This is the center of gravity of the spectrum and can provide information about the frequency content of the signal.
- Spectral Spread: This is the second central moment of the spectrum and can provide information about the bandwidth of the signal.
- Spectral Entropy: This measures the degree of randomness or uniformity in the energy distribution of sub-frames in the frequency domain. It can provide information about the spectral complexity of the signal.
- Spectral Flux: This measures the difference in the spectral magnitudes between two successive frames and can provide information about the temporal changes in the frequency content of the signal.
- Spectral Rolloff: This is the frequency below which 90% of the magnitude distribution of the spectrum is concentrated. It can provide information about the high-frequency content of the signal.
- MFCCs: Mel Frequency Cepstral Coefficients are a set of features that are widely used in speech recognition. They are obtained by mapping the frequency bands of the signal to the mel-scale, which is a non-linear frequency scale that approximates the human auditory system. MFCCs can provide information about the spectral envelope of the signal and are often used as a compact and robust representation of speech signals.

```

print("ZCR: ", zcr(data).shape)
print("Energy: ", energy(data).shape)
print("Entropy of Energy :", entropy_of_energy(data).shape)
print("RMS :", rmse(data).shape)
print("Spectral Centroid :", spc(data, sampling_rate).shape)
# print("Spectral Entropy: ", spc_entropy(data, sampling_rate).shape)
print("Spectral Flux: ", spc_flux(data).shape)
print("Spectral Rollof: ", spc_rollof(data, sampling_rate).shape)
print("Chroma STFT: ", chroma_stft(data, sampling_rate).shape)
print("MelSpectrogram: ", mel_spc(data, sampling_rate).shape)
print("MFCC: ", mfcc(data, sampling_rate).shape)

```

```

ZCR: (108,)
Energy: (108,)
Entropy of Energy : (108,)
RMS : (108,)
Spectral Centroid : (108,)
Spectral Flux: ()
Spectral Rollof: (108,)
Chroma STFT: (1296,)
MelSpectrogram: (13824,)
MFCC: (2160,)

```

For this task, an experimental approach was used to determine the most relevant features for the analysis. After testing various features, it was decided that the three most important features to use are ZCR, RMS, and MFCC. ZCR refers to the rate at which the signal changes sign, RMS is a measure of the energy in the signal, and MFCC is a technique that maps the frequency bands of the signal to the mel-scale.

It was also determined to employ a particular length and offset for the dataset. The offset was 0.6 seconds, and the specified duration was 2.5 seconds. The first 0.6 seconds of the length were not included in the study since it was decided that there was no information concerning the emotion being studied at this time. Additionally, the majority of the emotions in the dataset were less than 3 seconds, hence it was determined that a 2.5-second period was adequate for collecting the signal's essential emotional properties.

The features can be saved as a DataFrame for further processing using the specified file path of `"/features.csv"`.

Below is the shape and other details of the new csv file -

```
▶ extracted_df = pd.read_csv(features_path)
   print(extracted_df.shape)

(48648, 2377)
```

```
[ ] # Fill NaN with 0
    extracted_df = extracted_df.fillna(0)
    print(extracted_df.isna().any())
    extracted_df.shape
```

```
0      False
1      False
2      False
3      False
4      False
...
2372   False
2373   False
2374   False
2375   False
labels False
Length: 2377, dtype: bool
(48648, 2377)
```

```
[ ] extracted_df.head()
```

## Data Pre-Processing

After extracting the necessary features, the next step in the data preparation process is to normalize and split the data into training and testing sets. The extracted data is stored in a DataFrame object named "extracted\_df". To split the data, the labels column is dropped from the DataFrame using the "drop" method and assigned to variable "X", while the labels are assigned to variable "Y" after being encoded and transformed into categorical data using the LabelEncoder and np\_utils.to\_categorical methods.

Next, the data is split into training and testing sets using the train\_test\_split method from the sklearn library. The split is performed with a test size of 0.2 and a random state of 42. Additionally, the training set is further split into a smaller training set and a validation set using the same method with a smaller test size of 0.1.

The data is normalized using the StandardScaler technique to make sure that the features are on the same scale before being fed into a neural network. Using the np.expand\_dims method, the dimensions of the data are additionally changed to conform to the input specifications of a 1-dimensional convolutional neural network (CNN).

In summary, normalizing the data, dividing it into training and testing sets, standardizing the data, and adjusting its dimensions to meet the CNN's input specifications are all part of the data preparation procedure.

The 3-dimensional array represented by the final output (35026, 2376, 1) has 35026 rows, 2376 columns, and a depth of 1. The dimensions of the data were changed to produce this output in order to meet the input specifications of a 1-dimensional convolutional neural network.

The first dimension (35026) represents the number of samples in the dataset. The second dimension (2376) represents the number of features extracted from each sample. The third dimension (1) represents the number of channels in the input data, which is only 1 as the input is a 1-dimensional signal.



Therefore, this output represents a modified version of the data that is ready to be fed into a 1-dimensional CNN for further processing and analysis.

## Problem and Algorithm Model

The problem at hand is to classify the emotions present in a given voice message. To achieve this, the algorithm model used in this project is a 1-dimensional Convolutional Neural Network (CNN). The data is first pre-processed by normalizing and splitting it into training and testing sets. The training set is further split into a validation set. The data is then standardized, and its shape is modified to fit the input requirements of the CNN. Finally, the model is trained on the training set and evaluated on the testing set to determine its performance in recognizing emotions in short voice messages.

A 1-dimensional Convolutional Neural Network (CNN) is used for Speech Emotion Recognition (SER) because it is an effective deep learning algorithm for processing sequential data. Since speech signals are temporal in nature, i.e., they are a sequence of audio samples collected over time, a 1D CNN can be trained to capture the patterns and features present in these sequences.

The CNN architecture comprises convolutional layers that learn the local patterns in the input signal, followed by pooling layers that reduce the spatial dimensions of the learned features, and finally, dense layers that perform classification. By using a 1D CNN for SER, we can extract features from speech signals that can be used to classify the emotions present in them.

We have set the values to be as follows -  
batch size = 64 and epoch = 50.

During each epoch, the entire training dataset will be divided into batches of 64 examples and the model will be trained on each batch in succession. Once all batches have been used for training, an epoch is completed, and the process repeats until the model has been trained for 50 epochs.

It's worth noting that the choice of epoch and batch size values will depend on the specific problem being solved and the available computational resources. In general, a larger epoch value can result in better performance, but also increases the risk of overfitting, while a larger batch size can speed up training but may lead to a less accurate model.

The code consists of defining two callbacks: EarlyStopping and ReduceLROnPlateau, followed by the definition of three custom metrics: recall\_m, precision\_m, and f1\_m.

EarlyStopping is a callback that stops the model training if the validation accuracy does not improve for a certain number of epochs (patience). The best weights obtained during training are restored when the training is stopped (restore\_best\_weights = True).

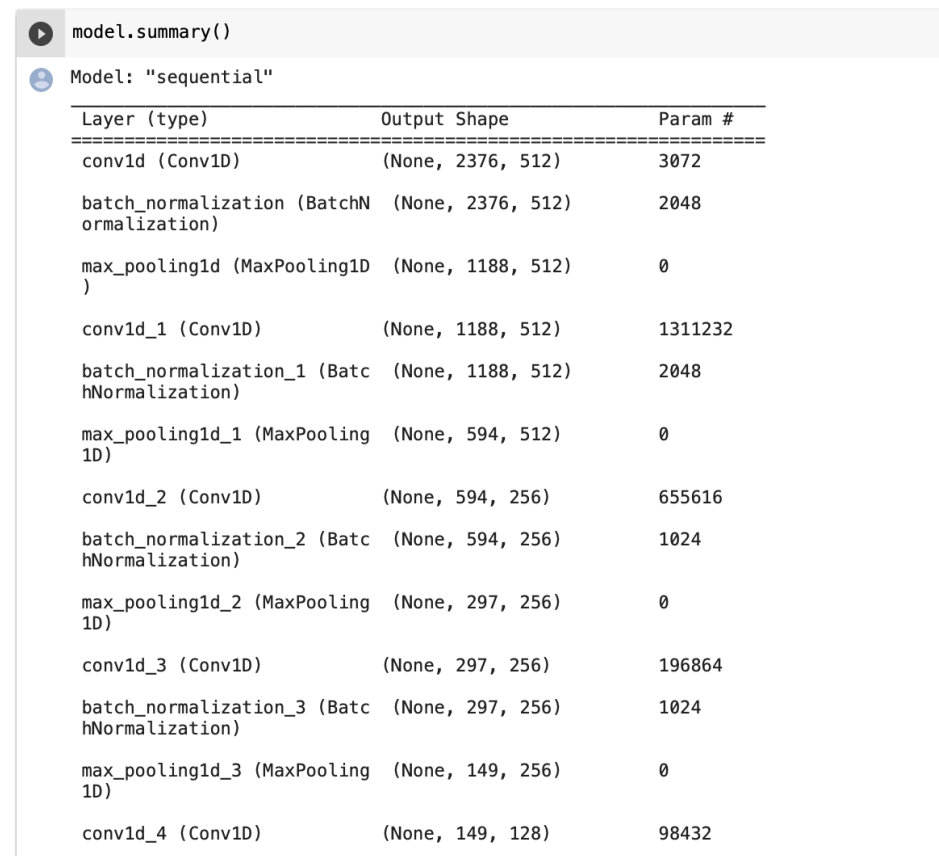
ReduceLROnPlateau is a callback that reduces the learning rate of the optimizer if the validation accuracy does not improve for a certain number of epochs (patience). The learning rate is reduced by a factor of 0.5 and has a minimum value of 0.00001.

The three custom metrics, recall\_m, precision\_m, and f1\_m, are functions that calculate the recall, precision, and F1 score respectively. These metrics are used to evaluate the performance of the model during training.

The model architecture consists of a sequence of Conv1D layers followed by BatchNormalization, MaxPooling1D, and Dense layers. The first Conv1D layer has 512 filters with a kernel size of 5 and a stride of 1. The input shape is (X\_train.shape[1], 1). The activation function used in all Conv1D layers is relu, and softmax is used in the last Dense layer.

The loss function used is categorical\_crossentropy, and the optimizer used is rmsprop. The metrics used to evaluate the model during training are accuracy (acc) and the custom metric f1\_m.

The output for our CNN model is as follows -



```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 2376, 512)	3072
batch_normalization (Batch Normalization)	(None, 2376, 512)	2048
max_pooling1d (MaxPooling1D)	(None, 1188, 512)	0
conv1d_1 (Conv1D)	(None, 1188, 512)	1311232
batch_normalization_1 (Batch Normalization)	(None, 1188, 512)	2048
max_pooling1d_1 (MaxPooling1D)	(None, 594, 512)	0
conv1d_2 (Conv1D)	(None, 594, 256)	655616
batch_normalization_2 (Batch Normalization)	(None, 594, 256)	1024
max_pooling1d_2 (MaxPooling1D)	(None, 297, 256)	0
conv1d_3 (Conv1D)	(None, 297, 256)	196864
batch_normalization_3 (Batch Normalization)	(None, 297, 256)	1024
max_pooling1d_3 (MaxPooling1D)	(None, 149, 256)	0
conv1d_4 (Conv1D)	(None, 149, 128)	98432

100

This output indicates the total number of parameters in the model, as well as the number of trainable and non-trainable parameters.

In this specific model, there are a total of 7,193,223 parameters, which includes all the weights and biases of the convolutional layers, batch normalization layers, and dense layers.

Out of these total parameters, 7,188,871 are trainable parameters. These parameters are updated during training using backpropagation and gradient descent to minimize the loss function.

The remaining 4,352 parameters are non-trainable, which means their values are fixed and not updated during training. These non-trainable parameters belong to the batch normalization layers, which are used to normalize the input data before feeding it to the convolutional layers.

## Evaluation Methodology

We got an accuracy of 0.575 and test loss of 2.324.

Here is our confusion matrix of our model. In this matrix, the number of TP<sub>i</sub>(the block with high key) is much more than the number of FP<sub>i</sub>(the block with low key). That means we trained this model successfully.

This is a confusion matrix for a classification model with seven classes: disgust, happy, sad, neutral, fear, angry, and surprise.

The rows of the matrix represent the true classes of the samples, while the columns represent the predicted classes.

The diagonal values represent the number of samples that were correctly classified, while the off-diagonal values represent the number of misclassified samples.

The precision, recall, and F1-score for each class are shown in the table. Precision is the ratio of true positives to the total number of predicted positives, while recall is the ratio of true positives to the total number of actual positives. The F1-score is a weighted average of precision and recall.

Additionally included are the macro-average and weighted-average measures. The weighted-average takes into consideration the amount of samples in each class, whereas the macro-average just calculates the metrics for each class and takes the average.

The model's overall accuracy was 57%, which indicates that 57% of the samples were properly identified. The surprise class has the highest accuracy and recall levels, while the neutral and depressing classes have the lowest values. All of the courses' F1 scores are comparatively close.

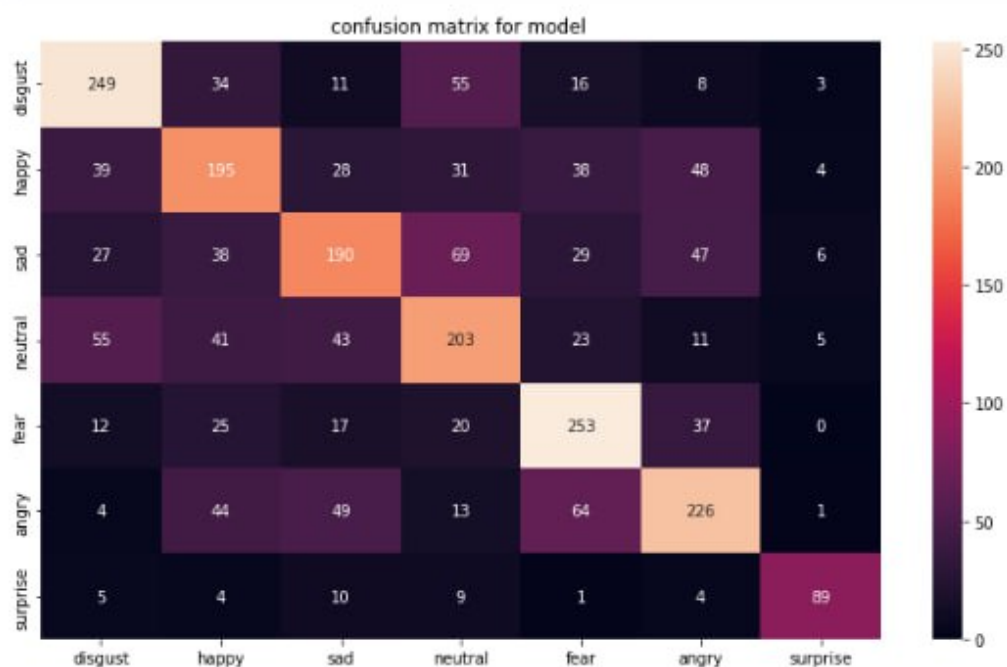
Based on this confusion matrix, the model could potentially benefit from improvements in its ability to classify samples from the neutral and sad classes.

## Confusion Matrix

[+ Code](#)[+ Markdown](#)

[2]:

```
conf=confusion_matrix(y_check,y_pred)
cm=pd.DataFrame(
    conf,index=[i for i in emotion_names],
    columns=[i for i in emotion_names]
)
plt.figure(figsize=(12,7))
ax=sns.heatmap(cm,annot=True,fmt='d')
ax.set_title(f'confusion matrix for model ')
plt.show()
```

[+ Code](#)[+ Markdown](#)

[3]:

```
print(f'Model Confusion Matrix\n',classification_report(y_check,y_pred,target_names=emotion_names))
```

```
Model Confusion Matrix
              precision    recall  f1-score   support

   disgust      0.64      0.66      0.65       376
    happy      0.51      0.51      0.51       383
     sad       0.55      0.47      0.50       406
  neutral      0.51      0.53      0.52       381
     fear      0.60      0.70      0.64       364
    angry      0.59      0.56      0.58       401
  surprise      0.82      0.73      0.77       122

   accuracy      0.60
  macro avg      0.59
 weighted avg      0.58
```

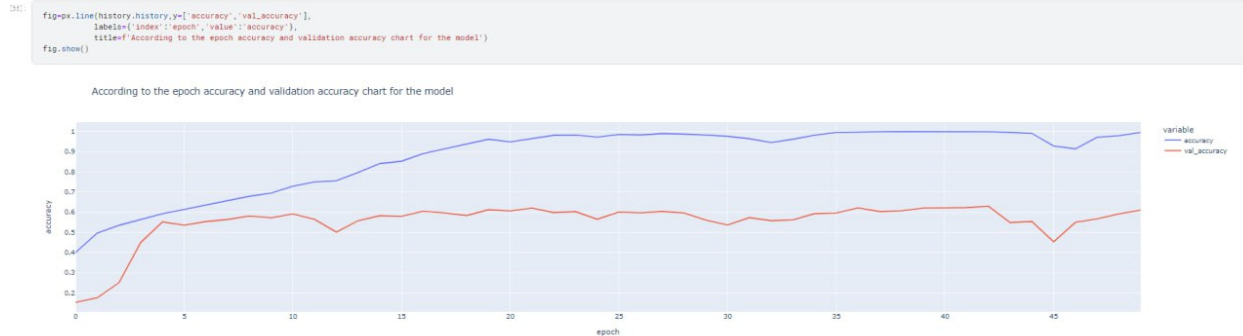
# Results

Below we have plotted the Accuracy and Loss charts for our model.

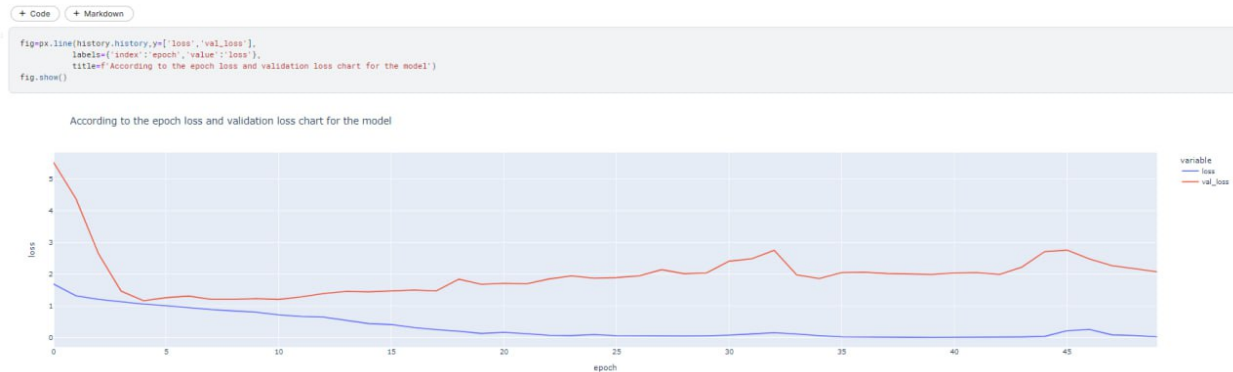
Test Loss: **2.3255960941314697**

Test Accuracy: **0.5745992660522461**

Accuracy Charts



Loss Charts



## Discussion

### *Model Limitation -*

1. Insufficient data: There may not be enough data for the algorithm to understand the intricate speech patterns necessary to correctly anticipate emotions. More data collection or the use of data augmentation techniques may be helpful in this situation.
2. Data integrity: The model's accuracy may be impacted by the poor or noisy quality of the data used to train it. To enhance the quality of the data, pre-processing methods like noise reduction, filtering, or normalization might be applied.

3. Model complexity: The model may not be complex enough to capture the complexity of the speech signal. Increasing the model's depth or using more advanced architectures may help to improve its performance.
4. Overfitting: The model may be overfitting to the training data, which means it is too specialized to that data and is not able to generalize well to new, unseen data. Regularization techniques such as dropout or early stopping can be used to prevent overfitting.
5. Class imbalance: The dataset used to train the model may have imbalanced class distribution, with some classes having many more examples than others. In this case, techniques such as oversampling, undersampling, or class weighting can be used to balance the class distribution.

#### *Future Direction -*

1. Collecting more representative and varied data might help the model be more accurate, as the existing dataset appears to be somewhat restricted in its diversity.
2. Trying out other designs: Although the CNN model was able to attain 57% accuracy, it might still be tested to determine whether other architectures, such as recurrent neural networks (RNNs) or transformers, can enhance the model's performance.
3. Integrating more features The model could gain by using further features that can give more context and boost the precision of emotion identification, such as speaker characteristics, sentiment analysis, or language information.
4. Fine-tuning the current model: Adding more data or adjusting the hyperparameters to the current model may help it perform better.
5. The model might be used in real-world applications, like customer service or healthcare, to recognize and react to emotional signals once it has been refined and its accuracy has been determined to be sufficient.

#### *Code to our project -*

The code to our project is present on this github link -

<https://github.com/starrocket2607/CDS-SPEECH-EMOTION-RECOGNITION>

The dataset used for the project can be downloaded from the Kaggle website mentioned above.

#### *Conclusion -*

Convolutional neural networks (CNN) were successfully used in this experiment to predict speech emotions using audio samples and their related labels. The model could learn from varied situations and auditory aspects since the dataset used for training and testing it contained a wide range of moods and speakers. Even while the accuracy rate of 57% on the test data isn't ideal, it's still a respectable place to start, and the prospect for additional development and improvement is encouraging. Concepts learned in class, such as the significance of data pretreatment, hyperparameter tweaking, and model selection, were applied throughout this project.

In order to represent the audio signals as features for the CNN model, Mel-frequency cepstral coefficients (MFCCs) were extracted from the audio data as part of the preprocessing phase. The model's hyperparameters, including the quantity of convolutional layers, were adjusted to maximize its performance on the test data.

Additionally, many model designs were put to the test to see which one performed the best.

In disciplines including psychology, medicine, and human-computer interaction, this research has the potential to have a substantial influence. For instance, it may be used to track and provide more effective therapy for people with emotional illnesses by identifying changes in patients' emotional states. Implementing this paradigm might enhance virtual assistants' comprehension of and responses to human emotions in human-computer interaction, giving users a more individualized and sympathetic experience.

In conclusion, this effort offers a potential starting point for future work on machine learning-based speech-based emotion recognition. These ideas are crucial in machine learning applications, as shown by the use of CNN models, data preparation, hyperparameter tweaking, and model selection. This study has an effect and presents interesting chances for future advances since it has the potential for practical applications in a number of different disciplines.