

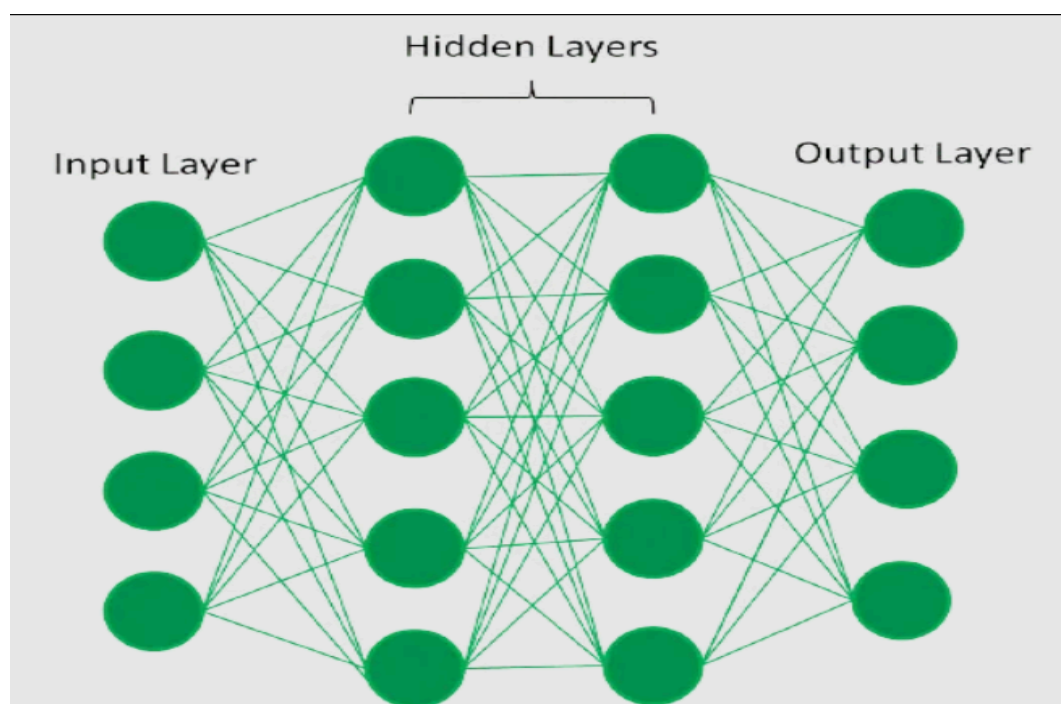
Speech Emotion Recognition

Final Report

Aditya Kumar
210020005

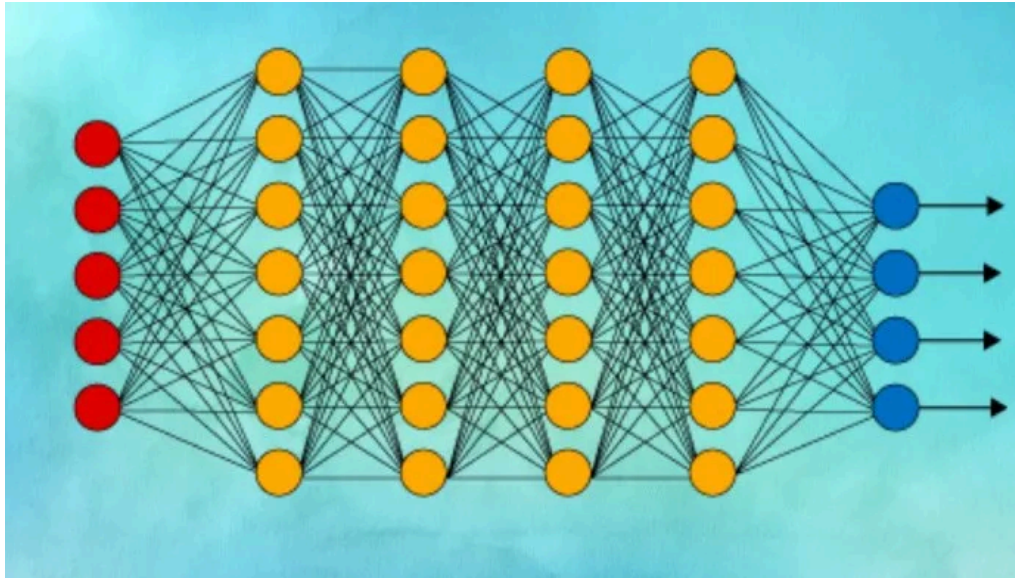
Neural Networks

Neural networks are computational models inspired by the human brain's structure and function. They consist of interconnected nodes or "neurons" that process information in layers. These networks are widely used in machine learning for tasks like classification, regression, and pattern recognition, leveraging their ability to learn from data through a process called training.

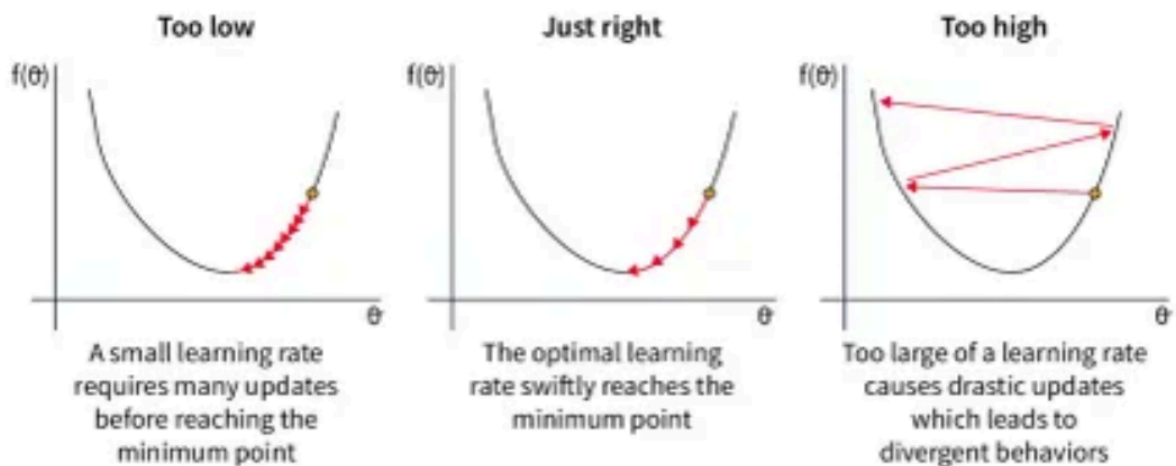


Deep Neural Networks

Deep neural networks (DNNs) feature multiple hidden layers, enabling them to model highly complex and abstract patterns in data. This depth allows DNNs to excel in a wide range of applications, including image and speech recognition, natural language processing, and autonomous systems. Despite their powerful capabilities, DNNs require significant computational resources and large datasets for effective training.



Optimisation



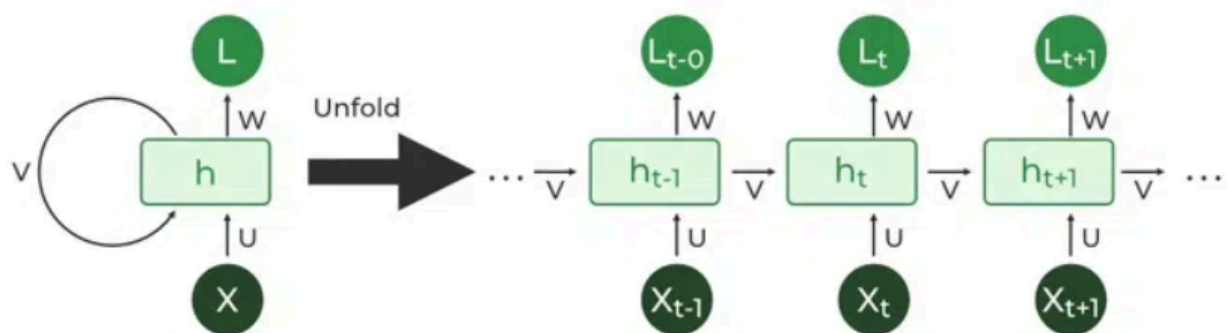
Optimization is the process of selecting the best solution out of the various feasible solutions that are available. In other words, optimization can be defined as a way of getting the best or the least value of a given function. In the majority of problems, the objective function $f(x)$ is constrained and the purpose is to identify the values of x which minimize or maximize $f(x)$.

There are many types of optimizers:

- Gradient Descent
 1. Stochastic Gradient Descent (SGD)
 2. Mini-batch Gradient Descent
 3. Batch Gradient Descent
- Adam (Adaptive Moment Estimation)
- AdaGrad (Adaptive Gradient Algorithm)
- RMSProp (Root Mean Square Propagation)
- SGD with Momentum

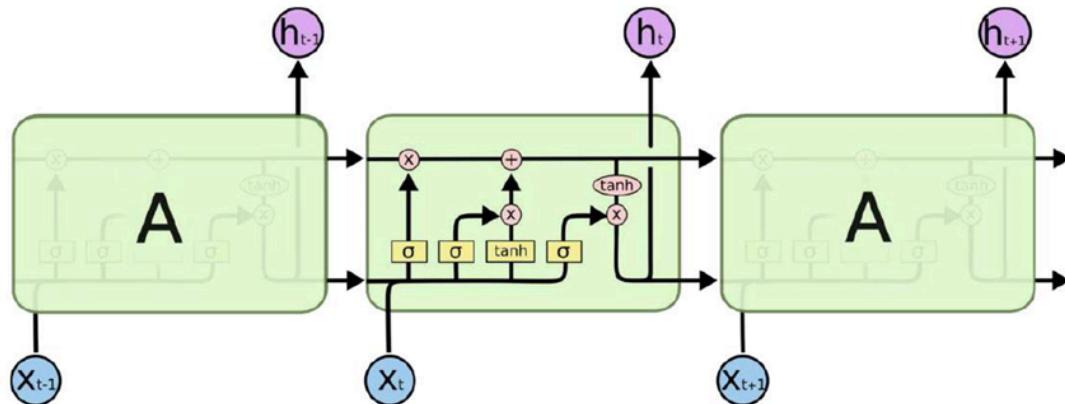
Recurrent Neural Networks

Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still, in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer.



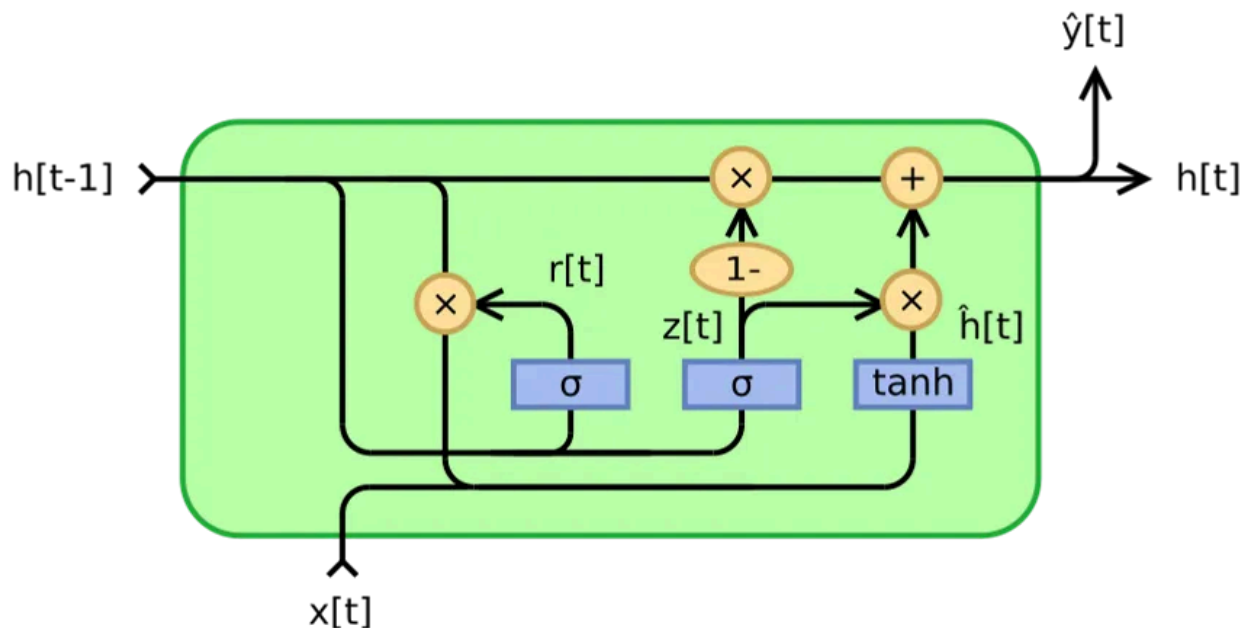
LSTM (Long- Short Term Memory)

Long Short-Term Memory is an improved version of recurrent neural network designed to capture long-term dependencies in sequence data. They address the vanishing gradient problem faced by standard RNNs through a sophisticated architecture that includes cell states and three types of gates: input, forget, and output gates.



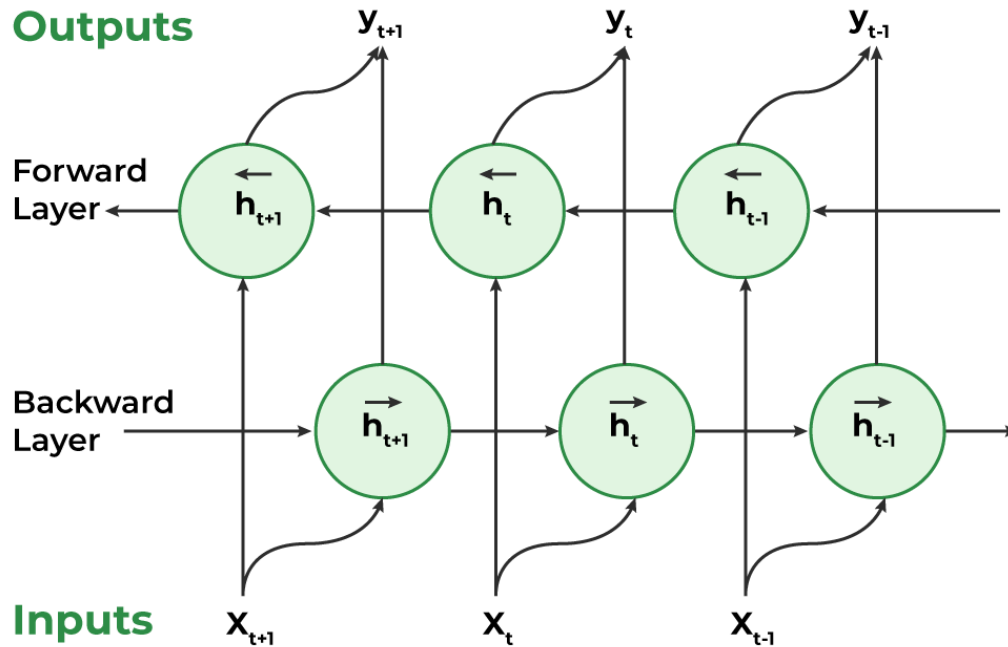
GRU (Gated Recurrent Units)

Gated Recurrent Units (GRUs) are a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem and improve the learning of long-term dependencies. GRUs simplify the LSTM architecture by combining the forget and input gates into a single update gate and merging the cell state and hidden state.



Bi Directional RNN

Bi-Directional Recurrent Neural Networks (Bi-RNNs) are an extension of traditional RNNs designed to capture context from both past and future states. This is particularly useful for tasks where understanding the full context of a sequence is crucial.



NLP

Natural language processing (NLP) is a field of computer science and a subfield of artificial intelligence that aims to make computers understand human language. NLP uses computational linguistics, which is the study of how language works, and various models based on statistics, machine learning, and deep learning. These technologies allow computers to analyze and process text or voice data, and to grasp their full meaning, including the speaker's or writer's intentions and emotions.

NLP Techniques

- Text Processing and Preprocessing In NLP
- Syntax and Parsing In NLP
- Semantic Analysis
- Information Extraction

- Text Classification in NLP
- Language Generation
- Speech Processing

Word Embeddings in NLP

Word Embedding is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meanings to have a similar representation.

Need for Word Embedding?

- To reduce dimensionality
- To use a word to predict the words around it.
- Inter-word semantics must be capture

How are Word Embeddings used?

- They are used as input to machine learning models.
- Take the words —> Give their numeric representation —> Use in training or inference.
- To represent or visualize any underlying patterns of usage in the corpus that was used to train them.

Approaches for Text Representation

One-Hot Encoding

One-hot encoding is a simple method for representing words in natural language processing (NLP). In this encoding scheme, each word in the vocabulary is represented as a unique vector, where the dimensionality of the vector is equal to the size of the vocabulary. The vector has all elements set to 0, except for the element corresponding to the index of the word in the vocabulary, which is set to 1.

While one-hot encoding is a simple and intuitive method for representing words in NLP, it has several disadvantages, which may limit its effectiveness in certain applications.

- One-hot encoding results in high-dimensional vectors, making it computationally expensive and memory-intensive, especially with large vocabularies.
- It does not capture semantic relationships between words; each word is treated as an isolated entity without considering its meaning or context.

- It is restricted to the vocabulary seen during training, making it unsuitable for handling out-of-vocabulary words.

Index	Animal	One-Hot code →	Index	Dog	Cat	Sheep	Lion	Horse
0	Dog		0	1	0	0	0	0
1	Cat		1	0	1	0	0	0
2	Sheep		2	0	0	1	0	0
3	Horse		3	0	0	0	0	1
4	Lion		4	0	0	0	1	0

Bag of Word (Bow)

Bag-of-Words (BoW) is a text representation technique that represents a document as an unordered set of words and their respective frequencies. It discards the word order and captures the frequency of each word in the document, creating a vector representation.

While BoW is a simple and interpretable representation, below disadvantages highlight its limitations in capturing certain aspects of language structure and semantics:

- BoW ignores the order of words in the document, leading to a loss of sequential information and context making it less effective for tasks where word order is crucial, such as in natural language understanding.
- BoW representations are often sparse, with many elements being zero resulting in increased memory requirements and computational inefficiency, especially when dealing with large datasets.

TF-IDF

Term Frequency-Inverse Document Frequency, commonly known as TF-IDF, is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents (corpus). It is widely used in natural language processing and

information retrieval to evaluate the significance of a term within a specific document in a larger corpus. TF-IDF consists of two components:

- **Term Frequency (TF):** Term Frequency measures how often a term (word) appears in a document. It is calculated using the formula:

$$TF(t, d) = \frac{\text{Total number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

- **Inverse Document Frequency (IDF):** Inverse Document Frequency measures the importance of a term across a collection of documents. It is calculated using the formula:

$$IDF(t, D) = \log \left(\frac{\text{Total documents}}{\text{Number of documents containing term } t} \right)$$

- The TF-IDF score for a term t in a document d is then given by multiplying the TF and IDF values:

$$TF - IDF(t, d, D) = TF(t, d)IDF(t, D)$$

The higher the TF-IDF score for a term in a document, the more important that term is to that document within the context of the entire corpus. This weighting scheme helps in identifying and extracting relevant information from a large collection of documents, and it is commonly used in text mining, information retrieval, and document clustering.

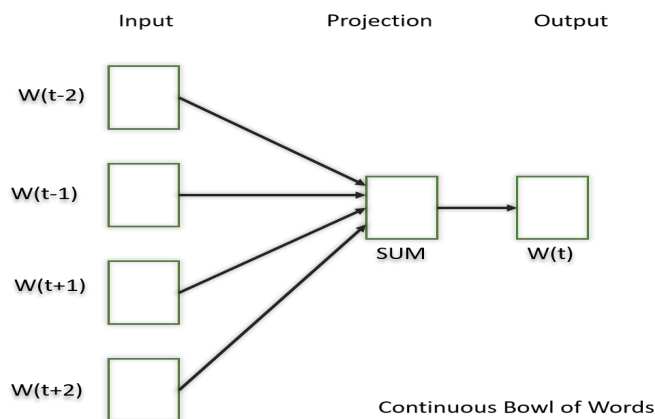
Word2Vec

Word2Vec is a neural approach for generating word embeddings. It belongs to the family of neural word embedding techniques and specifically falls under the category of distributed representation models. It is a popular technique in natural language processing (NLP) that is used to represent words as continuous vector spaces. Developed by a team at Google, Word2Vec aims to capture the semantic relationships between words by mapping them to high-dimensional vectors. The underlying idea is that words with similar meanings should have similar vector representations. In Word2Vec every word is assigned a vector. We start with either a random vector or one-hot vector.

Continuous Bag of Words(CBOW)

Continuous Bag of Words (CBOW) is a type of neural network architecture used in the Word2Vec model. The primary objective of CBOW is to predict a target word based on its context, which consists of the surrounding words in a given window. Given a sequence of words in a context window, the model is trained to predict the target word at the center of the window.

CBOW is a feedforward neural network with a single hidden layer. The input layer represents the context words, and the output layer represents the target word. The hidden layer contains the learned continuous vector representations (word embeddings) of the input words.



BERT

BERT is a transformer-based model that learns contextualized embeddings for words. It considers the entire context of a word by considering both left and right contexts, resulting in embeddings that capture rich contextual information.

Advantages and Disadvantage of Word Embeddings

Advantages

- It is much faster to train than hand build models like WordNet (which uses graph embeddings).
- Almost all modern NLP applications start with an embedding layer.
- It Stores an approximation of meaning.

Disadvantages

- It can be memory intensive.
- It is corpus dependent. Any underlying bias will have an effect on your model.
- It cannot distinguish between homophones. Eg: brake/break, cell/sell, weather/whether etc.

Project

Our aim is to make a model which will analyze the input speech data, and then express the emotion present in the speech audio file.

We will be given a raw audio file, on which we have to implement our model.

Data Pre-processing steps:

Some data pre-processing steps include:

- **Data collection:** We need a diverse set of audio recordings, representing various emotions.
- **Data cleaning:**
 - Remove Silence: We need to remove the silence or non-speech segments from the file.
 - Noise Reduction: To minimize the background noise and improve audio quality.
- **Emotion Representation:** Need to label all the emotions in numbers or in strings, uniformly. E.g.: '01' for 'angry' etc.
- **Normalization:**
 - Amplitude Normalization: Need to normalize the amplitude to ensure consistency across all raw files.
 - Length Normalization: Pad or Trim to a fixed length
- **Resampling:** Need to Resample the raw data set for consistent sample rate. E.g. 10kHz for all data

Feature Extraction:

There are a lot of feature extraction steps which can be done on a speech data file. Some of them are:

- Temporal Features: ZCR (Zero Crossing Rate) , Short Term Energy
- Spectral Features: MFCCs (Mel Frequency Cepstral Coefficients), Spectral Contrast and Spectral Bandwidth
- Prosodic Features: Pitch, Intensity and Jitter

Train, test and Validation sets

We need to split the data into training, testing and validation.

Model Selection and training

- We need to make the model RNN/LSTM/GRU's whichever suits the best for our time-series dataset.
- The model parameters such as layers, initial weightings etc. also need to be decided.
- Hyperparameters need to be taken into consideration and a valid optimizer should be selected.
- We also need to select the batch size, depending on the model and GPU performance.

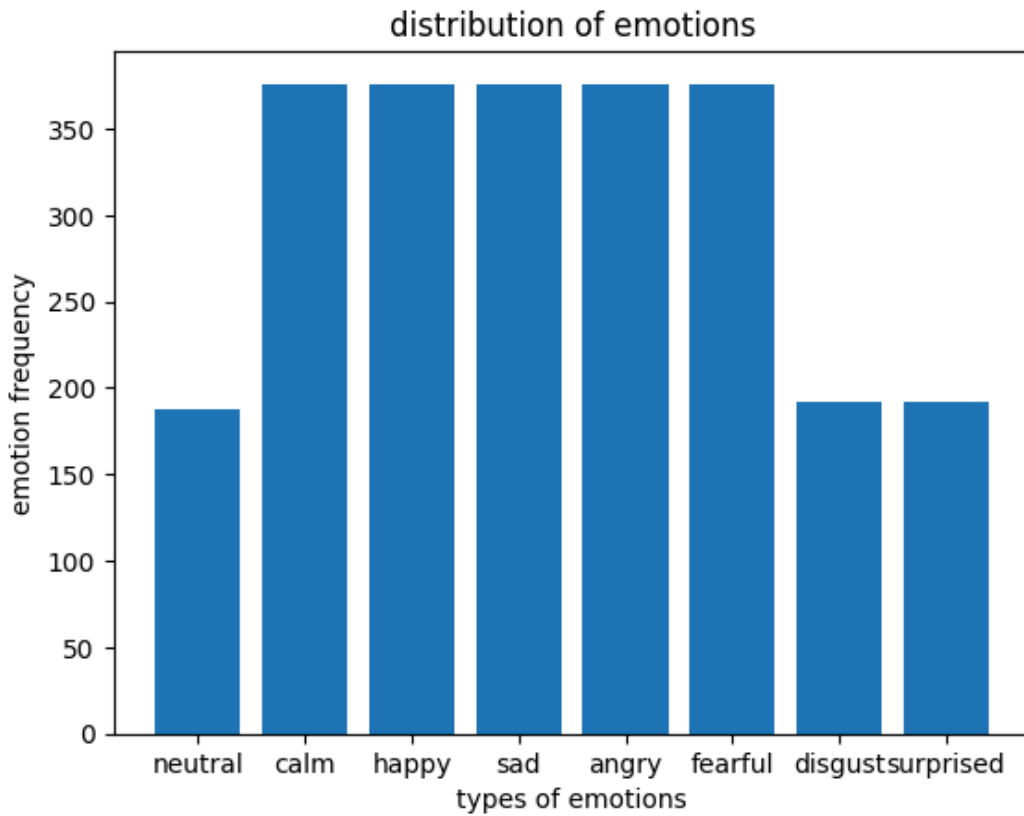
Model Evaluation

After training we need to evaluate our model on some basis like loss, accuracy score, confusion matrix for the validation and test sets.

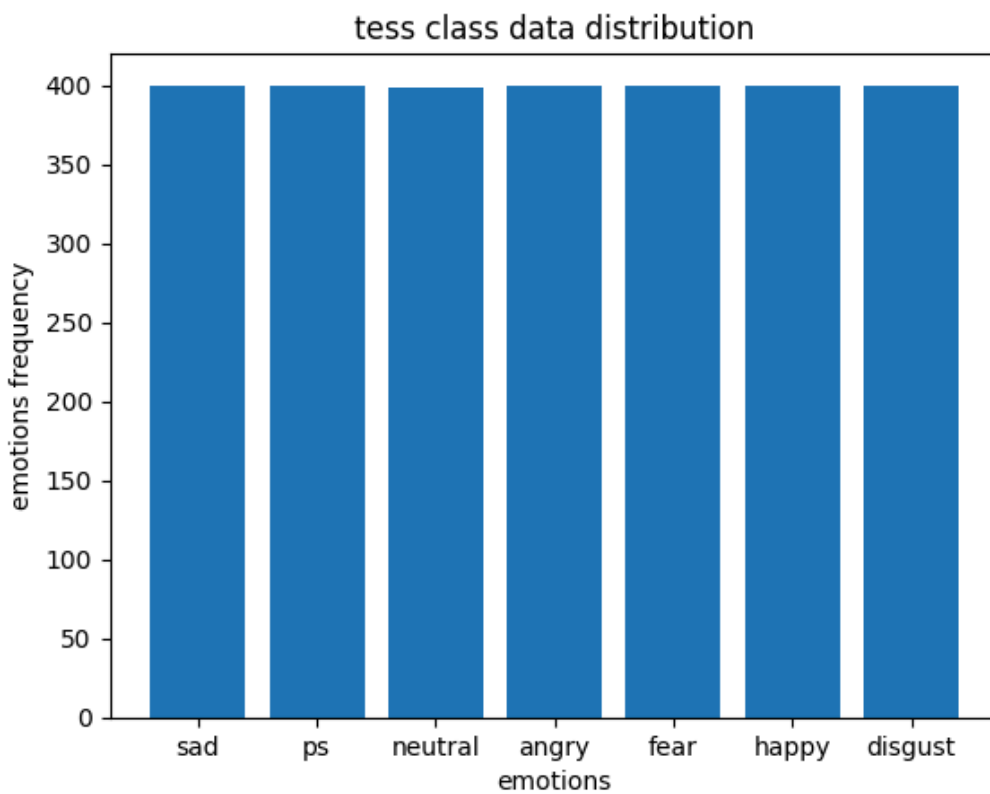
Data Visualisation

Distribution of emotions from RAVDESS dataset

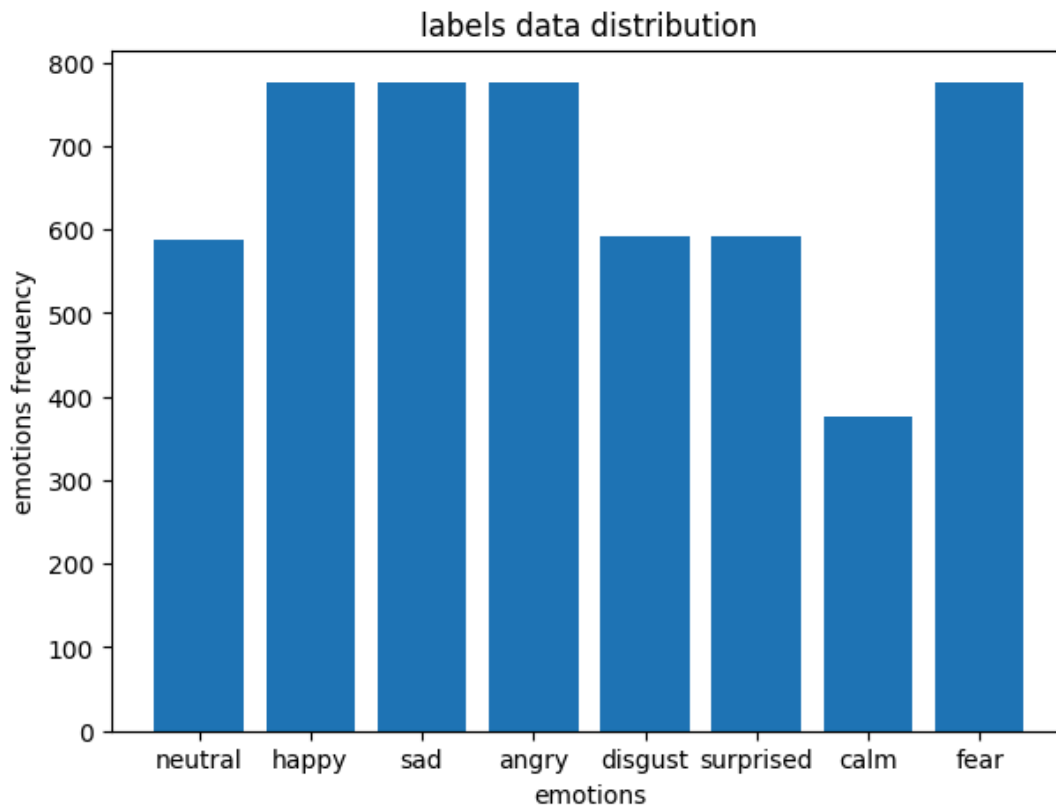
These are distributions from two dataset, RAVDESS SPEECH and RAVDESS SONG. 8 different types of emotions and their frequency can be seen in the following bar chart.



Distribution of emotions from TESS Dataset

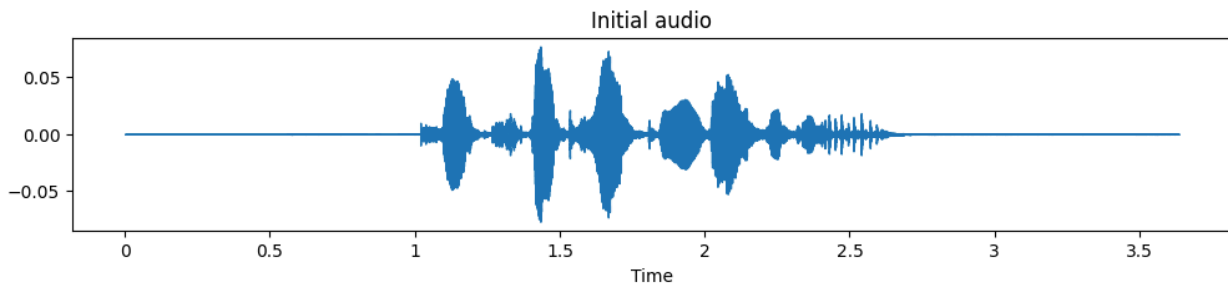


Total emotions distributions



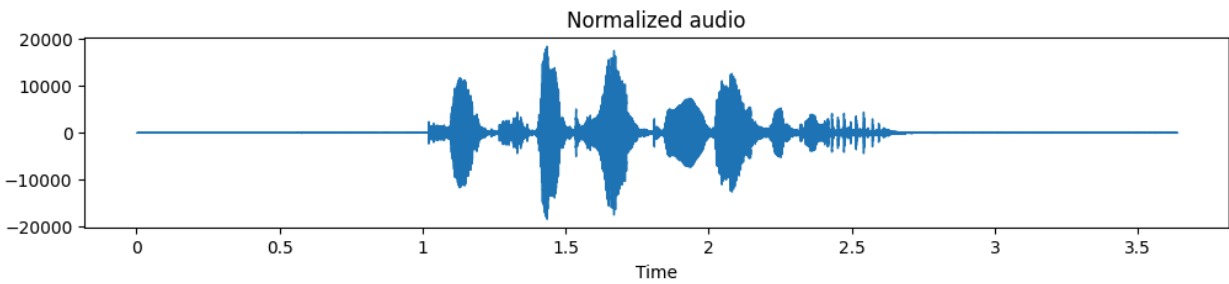
Data Pre-Processing

Audio Visualization



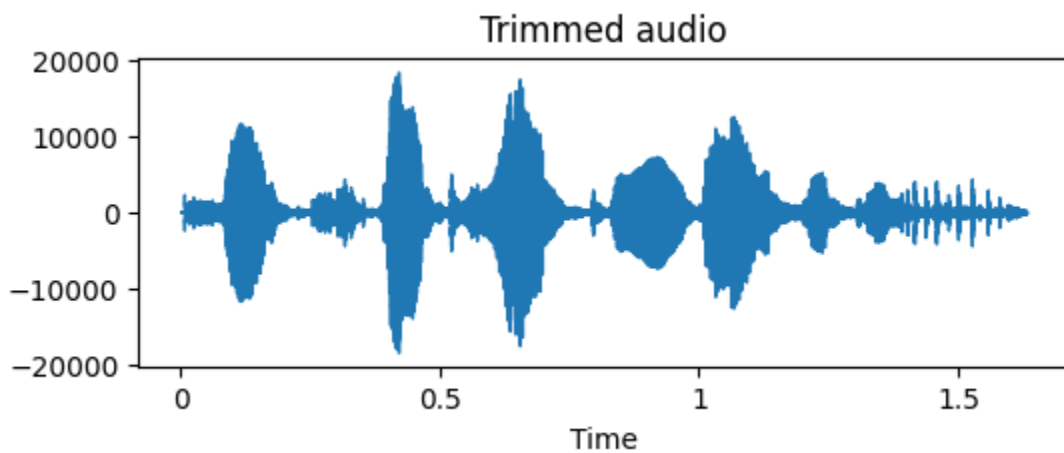
Normalization

Each 'AudioSegment' object is normalized to + 5.0 dBFS.



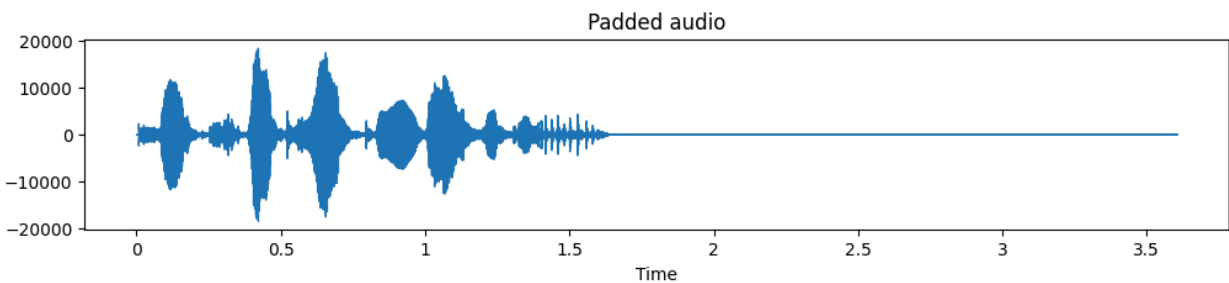
Trimmed Audio

Each audio is trimmed to remove the silence in the beginning and end.

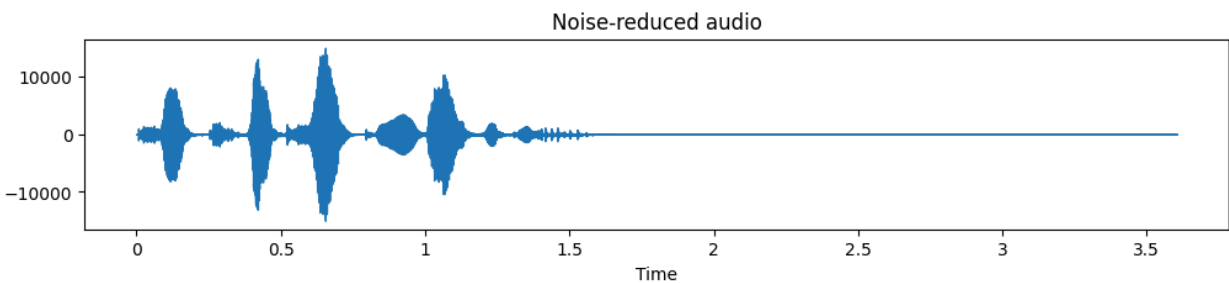


Padding

We pad every audio file to make it of the same length.



Noise reduction



Feature Extraction

RMS (root mean square): RMS is a measure of the energy or loudness of an audio signal.

ZCR (zero crossing rate): ZCR is the rate at which the audio signal changes sign from positive to negative or vice versa.

MFCC (Mel-Frequency Cepstral Coefficients): MFCCs are a representation of the short-term power spectrum of an audio signal, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

ZCR shape: (7708, 476, 1)

RMS shape: (7708, 476, 1)

MFCCs shape: (7708, 476, 13)

Training data shape: torch.Size([6744, 476, 15])

Training labels shape: torch.Size([6744, 1, 8])

Validation data shape: torch.Size([670, 476, 15])

Validation labels shape: torch.Size([670, 1, 8])

Convolutional LSTM

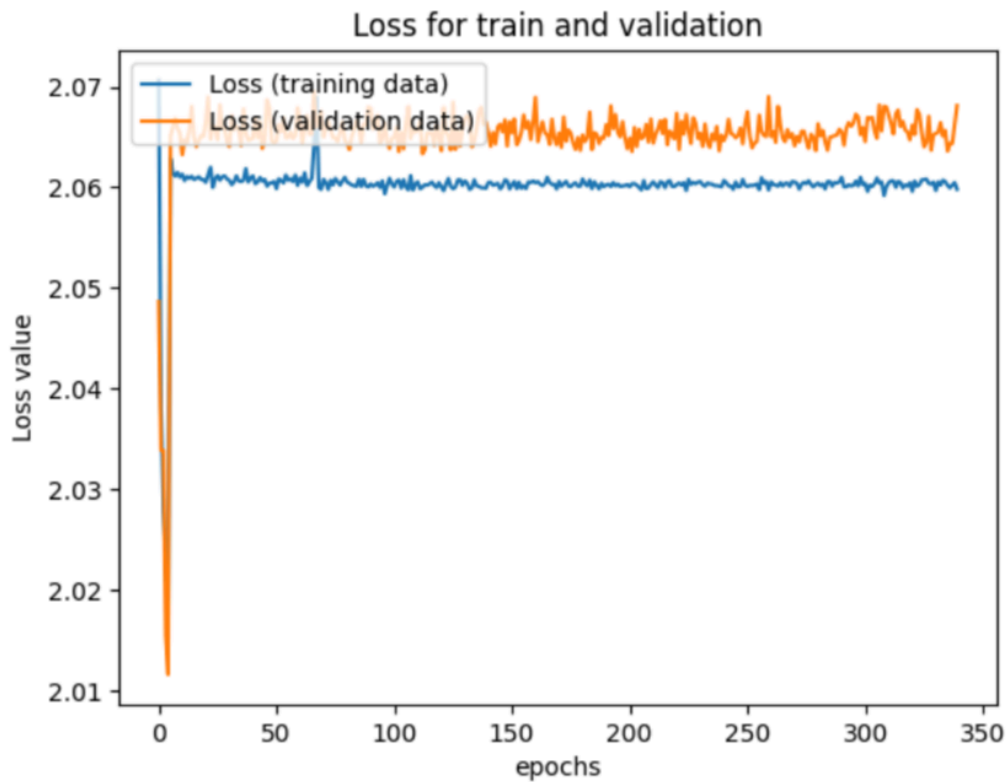
Training Accuracy → **0.1526**

Validation Accuracy → **0.1388**

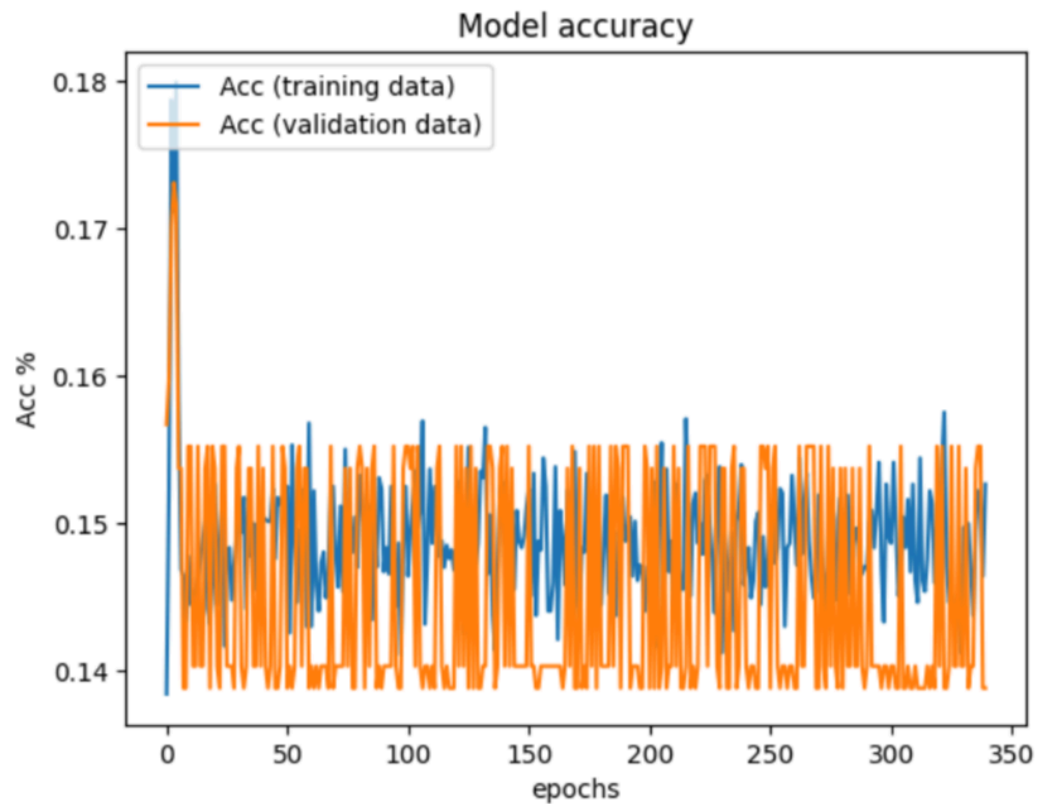
Testing Accuracy → **0.1365**

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 474, 64)	2944
max_pooling1d_1 (MaxPooling1D)	(None, 237, 64)	0
lstm_1 (LSTM)	(None, 64)	33024
dense_1 (Dense)	(None, 8)	520
Total params: 36488 (142.53 KB)		
Trainable params: 36488 (142.53 KB)		
Non-trainable params: 0 (0.00 Byte)		

Loss Graph for training and validation



Model Accuracy Graph



LSTM with Dropout

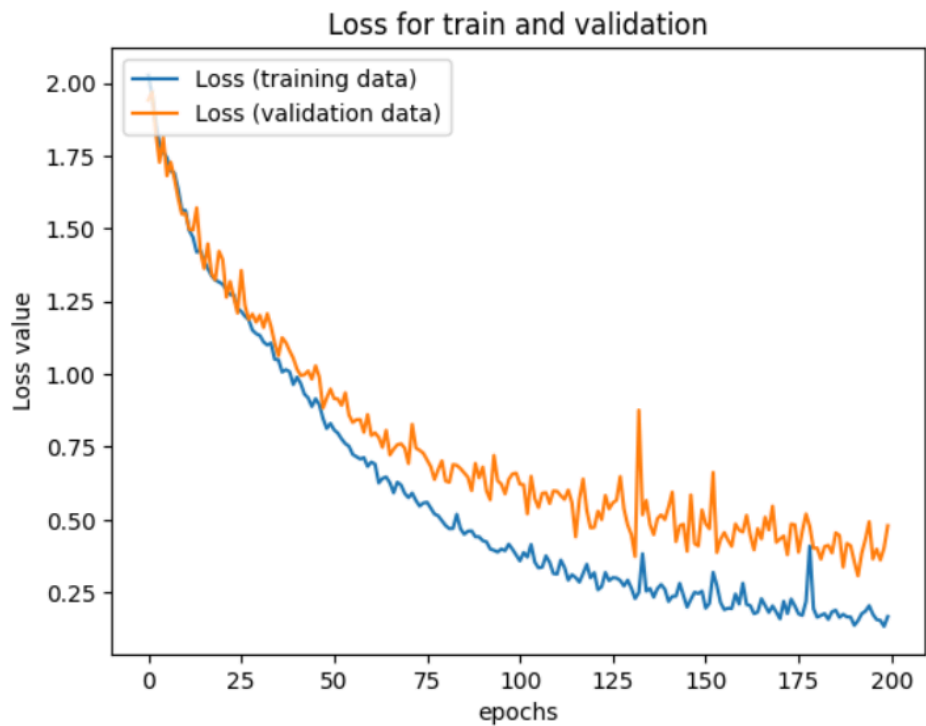
Training Accuracy - **0.9484**

Validation Accuracy - **0.8597**

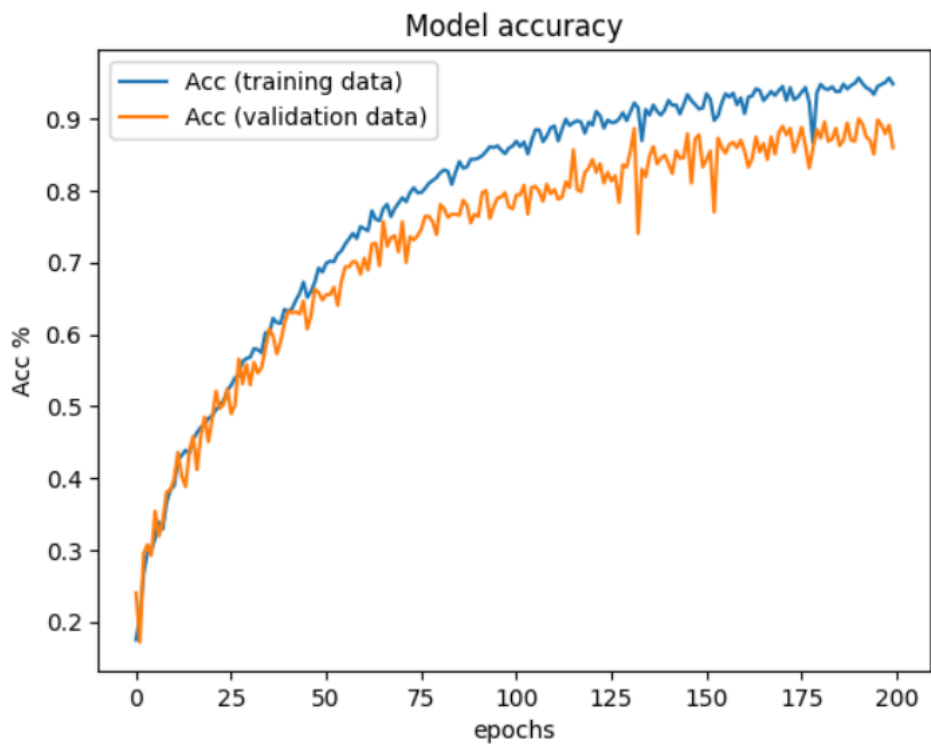
Testing Accuracy - **0.8597**

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 476, 128)	73728
dropout_2 (Dropout)	(None, 476, 128)	0
lstm_5 (LSTM)	(None, 64)	49408
dropout_3 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 8)	520
Total params: 123656 (483.03 KB)		
Trainable params: 123656 (483.03 KB)		
Non-trainable params: 0 (0.00 Byte)		

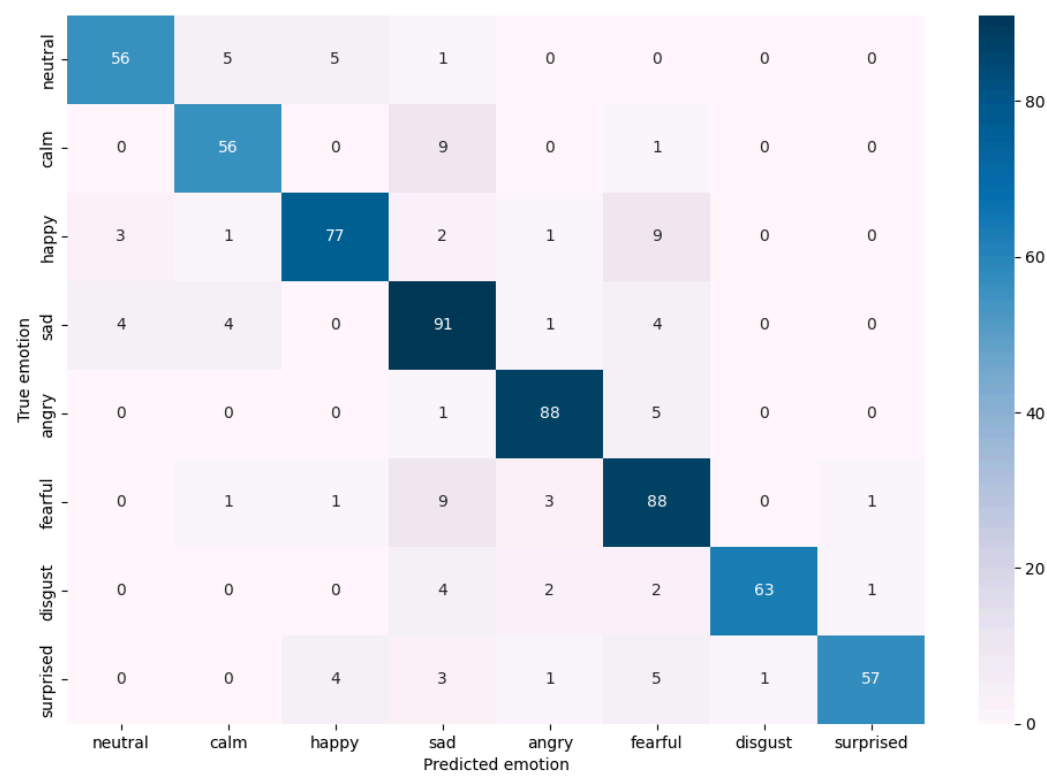
Loss Graph for training and validation



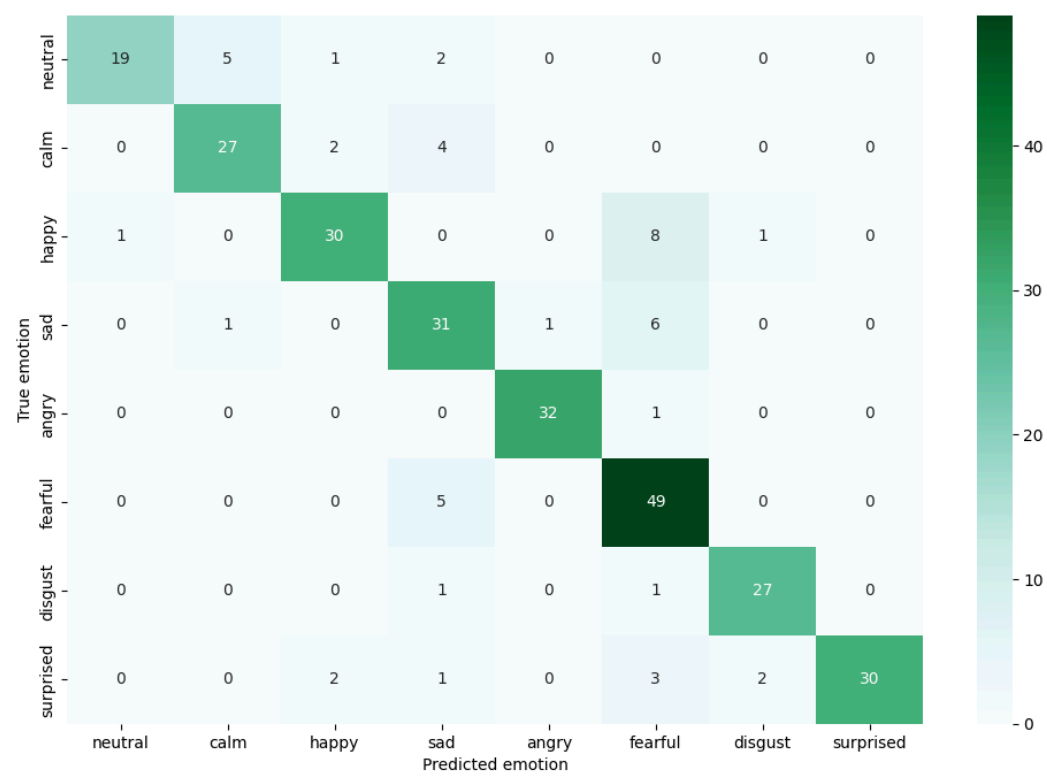
Model Accuracy Graph



Validation Confusion Matrix




Testing Confusion Matrix



Conclusion

Accuracy	LSTM with Dropout	Convolutional LSTM
Training	0.9484	0.15260.1365
Validation	0.8597	0.1388
Testing	0.8597	0.1365

References Used

 Speech Emotion Recognition [ID:30] Mentor's Resources

[Neural Networks and Dee Learning- Coursera](#)

[Improving Deep Neural Networks - Coursera](#)

[Sequence Models - Coursera](#)

[Krish Naik - NLP - Youtube Playlist](#)

<https://www.kaggle.com/code/muhammedaymanacs/speech-emotion-recognition-ser>

<https://www.kaggle.com/code/abduhrhmaneldeeb/speech-emotion-recognition>