ABSTRACT

The classification of emotional states of speakers through their speech signals is the primary objective of Speech Emotion Recognition, which is a developing research area. Due to their capability of automatically extracting relevant features from raw speech data, CNN have found extensive application in SER. In this paper, we propose a convolutional neural network model that helps in recognizing the emotion of the speech. The system that has been suggested comprises primarily of four elements: emotion classification, data augmentation, model training, and feature extraction. During the data augmentation phase, multiple signal processing methods, such as pitch shifting, time stretching, noise addition, and dynamic range alteration, are employed on the original speech signals. To capture the spectral information of speech signals, the feature extraction module employs Mel-Frequency Spectral Coefficients (MFCCs) in addition to their first and second-order derivatives. The proposed system undergoes training and evaluation processes utilizing two datasets, namely the RAVDESS and TESS. The CNN architecture is used by the emotion classification module to group feelings into one among the seven basic emotion categories: sad, fear, surprise, angry, happy, neutral and disgust.

Keywords: Speech Emotion Recognition, Data Augmentation, MFCC, CNN.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

1.	SER	Speech Emotion Recognition
2.	CNN	Convolutional Neural networks
3.	MFCC	Mel-Frequency Cepstral Coefficients
4.	RAVDESS	Ryerson Audio-Visual Database of Emotional Speech and Song
5.	TESS	Toronto Emotional Speech Set
6.	DNN	Deep Neural Networks
7.	RNN	Recurrent Neural Networks
8.	SVM	Support Vector Machine
9.	HMM	Hidden Markov Model
10.	DFT	Discrete Fourier Transform
11.	DCT	Discrete Cosine Transform
12.	ReLU	Rectified Linear Unit
13.	ADAM	Adaptive Moment Estimation
14.	UML	Unified Modeling language
15.	STFT	Short-Term Fourier Transform

1. INTRODUCTION

1.1 INTRODUCTION

Speech is a natural method for people to express themselves, and in the age of remote communication, being able to identify and interpret emotional expression in speech is essential. It can be difficult to identify emotions in speech because they are arbitrary and there is no consensus on how to quantify or categorize them. The SER system consists of different methods for classifying and processing speech signals in order to identify embedded emotions. It can be used for a variety of tasks, such as the analysis of caller-agent interactions or interactive voice assistants. This research aims to reveal the hidden emotions present in recorded speech by analyzing the resonance characteristics of the recorded audio. Speech conveys a lot of information, including speaker identification, age, gender, locality, and emotions. Emotion plays a vital role in expressing feelings, and speech characteristics alter depending on the speaker's emotional state. Although humans can recognize a speaker's emotional state without training, implementing this ability in machines is complicated. Detecting emotions in speech involves extracting acoustic features such as pitch, loudness, spectral characteristics, and duration from the audio signal. These features are then utilized to train a deep learning algorithm that classifies the expressed emotion. The efficacy of the deep learning algorithm, along with the quantity and quality of the extracted acoustic features, determine the overall system's efficiency.

Deep learning can be compared to the human nervous system in a single term. To learn from a dataset of images or audio, deep learning models are used, as they are for machine vision. Numerous deep learning models are developed to instruct a computer on how to visualize similarly to a human. Each node in a deep learning network model functions as a neuron in a bigger network, simulating the human nervous system. Deep learning models are essentially a subset of artificial neural networks where the algorithms analyze the input audio or image more deeply as it passes through each layer of the neural network.

1.2 MOTIVATION OF THE WORK

Speech emotion recognition is an important area of research that has the potential to greatly improve human-computer interaction and provide valuable insights into various fields such as psychology, medicine, and education. The motivation behind developing a speech emotion recognition system using a CNN algorithm is to create a reliable and accurate tool that can automatically identify and classify different emotions present in speech signals. By leveraging the power of deep learning, specifically convolutional neural networks (CNNs), researchers can extract complex features from speech signals and use them to classify emotions with high accuracy. This technology can be applied to a variety of real world applications such as sentiment analysis in social media, customer service chatbots, and virtual assistants.

Furthermore, the development of a CNN-based speech emotion recognition system has the potential to contribute to the field of affective computing, which aims to enable machines to understand and respond to human emotions. This can lead to improved user experiences in various contexts, including entertainment, healthcare, and education.

1.3 PROBLEM STATEMENT

This project uses a CNN model to attempt to build a speech emotion recognition system. The project will explore how a CNN model can be used to analyze the audio data and classify different emotions present in speech signals. The key goal of this research is to build a highly accurate SER system capable of real-time emotion detection and categorization in audio data.

The project will focus on developing a CNN model that can learn and extract useful features from audio signals, including pitch, spectral characteristics, and duration. The model will be trained on a large dataset of audio recordings with labelled emotions to improve its accuracy in recognizing emotions in speech.

2. LITERATURE SURVEY

2.1 INTRODUCTION

Due to the significance of speech emotion recognition (SER) in human-computer interaction and the advancement of artificial intelligence systems, there have been numerous recent publications and surveys on this topic. In this section, we will examine the latest studies relevant to the present research. Swain et al. conducted a review of SER system studies from 2000 to 2017, which covered databases, feature extraction, and classifiers. However, their research only considered traditional machine learning methods for classification and did not include neural networks and deep learning approaches. In a subsequent study, Khalil et al. reviewed deep learning techniques in SER, analyzing their strengths and limitations but did not focus on addressing their weaknesses. Anjali et al. recently published an overview of speech emotion detection methods, which provides a concise examination of the diverse features employed in recognizing speech emotions. Basu et al. conducted a study in 2020 focusing on the significance of speech emotion datasets and features, noise reduction techniques, and various classification approaches. Lastly, the study conducted by Akcay et al. provides a comprehensive overview of databases, features, classifiers, and emotion models, including machine learning techniques for improving classification, but did not report comparable results from various methods.

2.2 A REVIEW ON DATABASES, FEATURES AND CLASSIFIERS BY SWAIN IN 2018

Swain et al. conducted a review of SER system studies from 2000 to 2017, focusing on three perspectives: databases, feature extraction, and classifiers. While the research provided an in-depth analysis of databases and feature extraction, the authors only considered traditional machine learning methods for classification and expressed regret for not including neural networks and deep learning approaches.

2.3 A REVIEW ON SPEECH EMOTION RECOGNITION USING DEEP LEARNING TECNIQUES BY KHALIL

In a subsequent study, Khalil et al. conducted a review of discrete methods in SER utilizing deep learning techniques. The study discussed various deep learning approaches such as deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders, and analyzed their strengths and limitations. However, the researchers did not focus on the possible solutions for addressing the identified weaknesses of these approaches.

2.4 EXAMINING ON THE DIVERSE FEATURES OF THE SPEECH BY ANJALI

An overview of speech emotion detection methods was recently published by Anjali et al. The research offers a concise examination of the diverse features employed in recognizing speech emotions and evaluates the methods utilized for this purpose during the period between 2009 and 2018. Although the paper lacks in-depth analysis, it serves as a starting point for further research on this topic.

2.5 A STUDY ON THE IMPORTANCE OF SER SYSTEM BY BASU

Basu et al. conducted a concise study in 2020 focusing on the significance of speech emotion datasets and features, noise reduction techniques, and various classification approaches such as SVM and HMM. Although the research identified several relevant features related to speech emotion recognition, it lacked in-depth analysis of contemporary methods. Convolutional and recurrent neural networks were briefly mentioned as deep learning methods.

2.6 A COMPREHENSIVE STUDY ON DIFFERENT TECHNIQUES FOR SER SYSTEM BY AKCAY IN 2020

The study conducted by Akcay et al. provides a relatively comprehensive overview of databases, features, classifiers, and emotion models. The research also examines machine learning techniques for improving classification. However, the study did not report comparable results from various methods, except for initial findings from their original papers.

3. PROPOSED METHODOLOGY

3.1 INTRODUCTION

The CNN model is trained using the RAVDESS and TESS datasets. RAVDESS consist of 1440 audio samples whereas TESS dataset has 2800 audio files. So, there are 4240 audio samples in total to train the model. The silence is removed from the audio sample using the librosa.trim. Waveplots and Spectograms are plotted to visualize the audio samples. Waveplots represent the audio samples in time domain whereas spectrogram in frequency domain. In the process of data augmentation, various signal processing techniques like pitch shifting, time stretching, adding noise, changing dynamic range to the original speech signals are applied.

The key to speech emotion recognition is feature extraction process. The quality of the features directly influences the accuracy of classification results. The MFCC (Mel-Frequency Cepstral Coefficients) feature is extracted from the audio sample using librosa module. MFCC feature extraction technique basically includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT. The features extracted from the audio files by MFCC is splitted into training and testing samples. Now, the CNN model is trained using the training samples and it is tested on the testing samples to detect the emotion of the audio sample.

3.2 OBJECTIVES

The main objective of developing a speech emotion recognition system using a CNN algorithm is to accurately identify and classify different emotions present in speech signals. More specifically, the objectives of our project include:

- **Developing a robust and accurate CNN-based model for speech emotion recognition:**

  This involves designing and training a CNN architecture that can effectively extract relevant features from speech signals and accurately classify them into different emotion categories.

- **Collecting and preparing a dataset for training and testing the model:**

  The quality and quantity of the dataset used for training and testing the model plays a crucial role in the performance of the speech emotion recognition system. Therefore, the objective is to gather a comprehensive dataset and prepare it for use in the training process.

- **Evaluating the performance of the model:**

  This involves assessing the accuracy, precision, recall, and F1-score of the model in identifying and classifying emotions present in speech signals. This helps to determine the effectiveness of the system and identify areas for improvement.

- **Investigating the impact of various factors on the performance of the model:**

  The objective is to explore how various factors such as the size of the dataset, the type of features extracted, and the type of CNN architecture used affect the performance of the speech emotion recognition system.

3.3 SYSTEM ARCHITECTURE

This project uses the RAVDESS and TESS datasets, which contain a total of 4240 audio samples, to detect emotions such as happy, angry, sad, neutral, fear, disgust, and surprise. Unnecessary data is removed and silence is trimmed from each audio sample using librosa.effects.trim function. Data augmentation is also performed to improve the performance of deep learning models. The Mel-frequency cepstrum coefficient (MFCC) feature extraction technique is used to represent the spectral property of voice signals. The dataset is split into training and testing sets, and a CNN model is trained using the MFCC

features extracted from the audio files. The trained model is then used to predict the emotion of the input audio file.

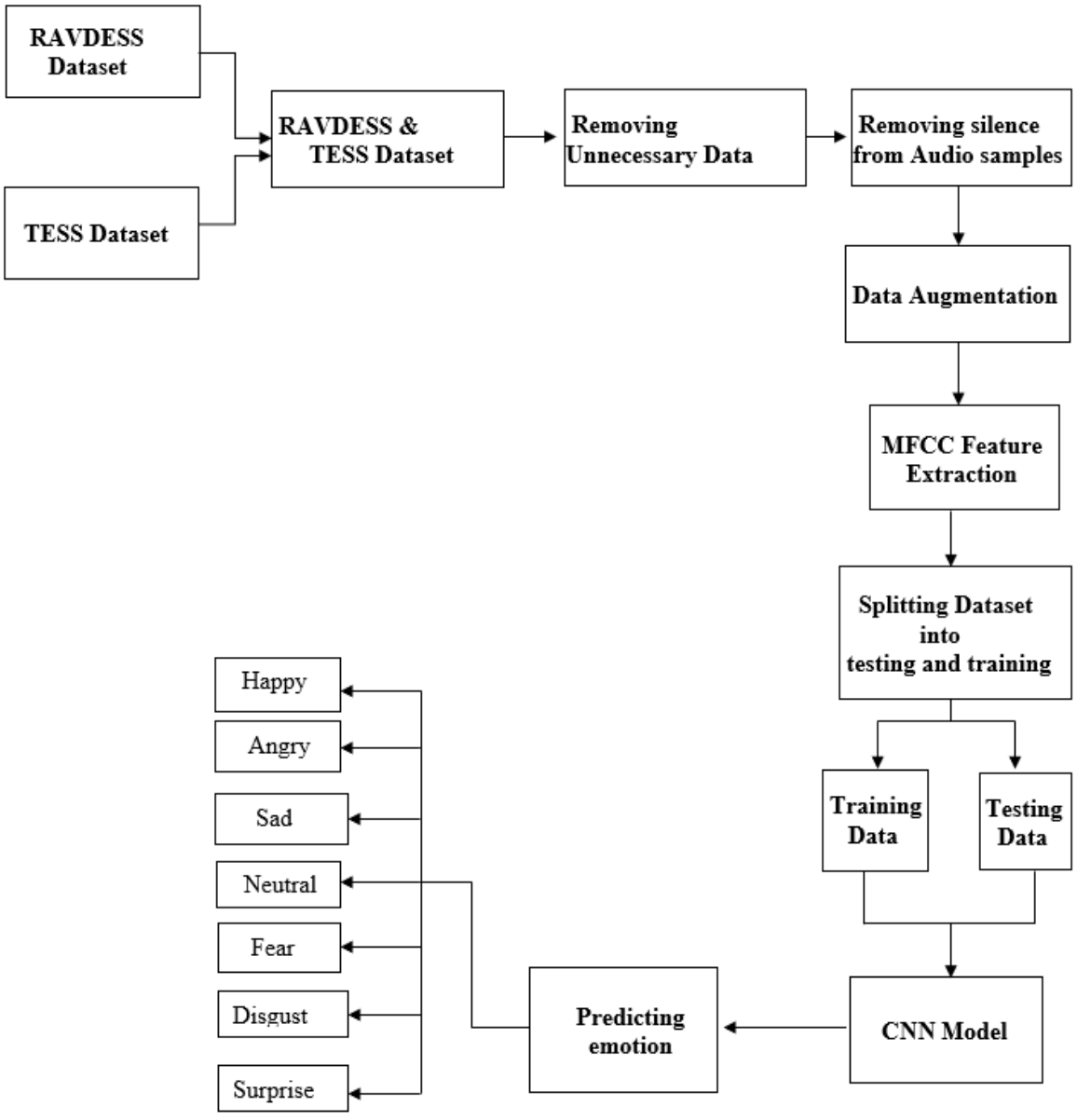


Fig 1 System Architecture

4. MODULES DIVISION

4.1 DATA COLLECTION

To initiate the SER system, the primary stage is to gather audio samples that fall into distinct emotional categories, which will be utilized to train the model. In this system, the audio samples are collected from the RAVDESS and TESS datasets. The unwanted audio samples are removed from the dataset. Therefore, it contains only the audio samples of emotions: neutral, happy, sad, angry, fearful, disgusted, and surprised. The silence is removed from every audio sample, so the unwanted data from each audio sample is decreased. The librosa.effects.trim function is used to remove the silence from the audio samples. If the sound is below 30 decibels, it is removed from the audio sample.

4.2 DATA AUGMENTATION

A common deep learning method is data augmentation, which creates new data from existing data to enlarge the training set. In SER, by increasing the quantity and variety of training data, data augmentation can aid in enhancing the efficiency of deep learning models. In the process of data augmentation, various signal processing techniques like pitch shifting, time stretching, adding noise and changing the initial speech signals in terms of their dynamic spectrum.

4.3 FEATURE EXTRACTION

The process of turning raw data into numerical traits that can be examined while preserving the specifics found in the original dataset is known as feature extraction. This process results in better outcomes than applying algorithms directly to the original data. MFCC is employed in this technique in order to extract features. The MFCC method requires applying a DFT onto every window, taking the logarithm of the amplitude, and converting

all frequencies to the Mel scale before performing an inverse DCT to extract the characteristics from the audio.

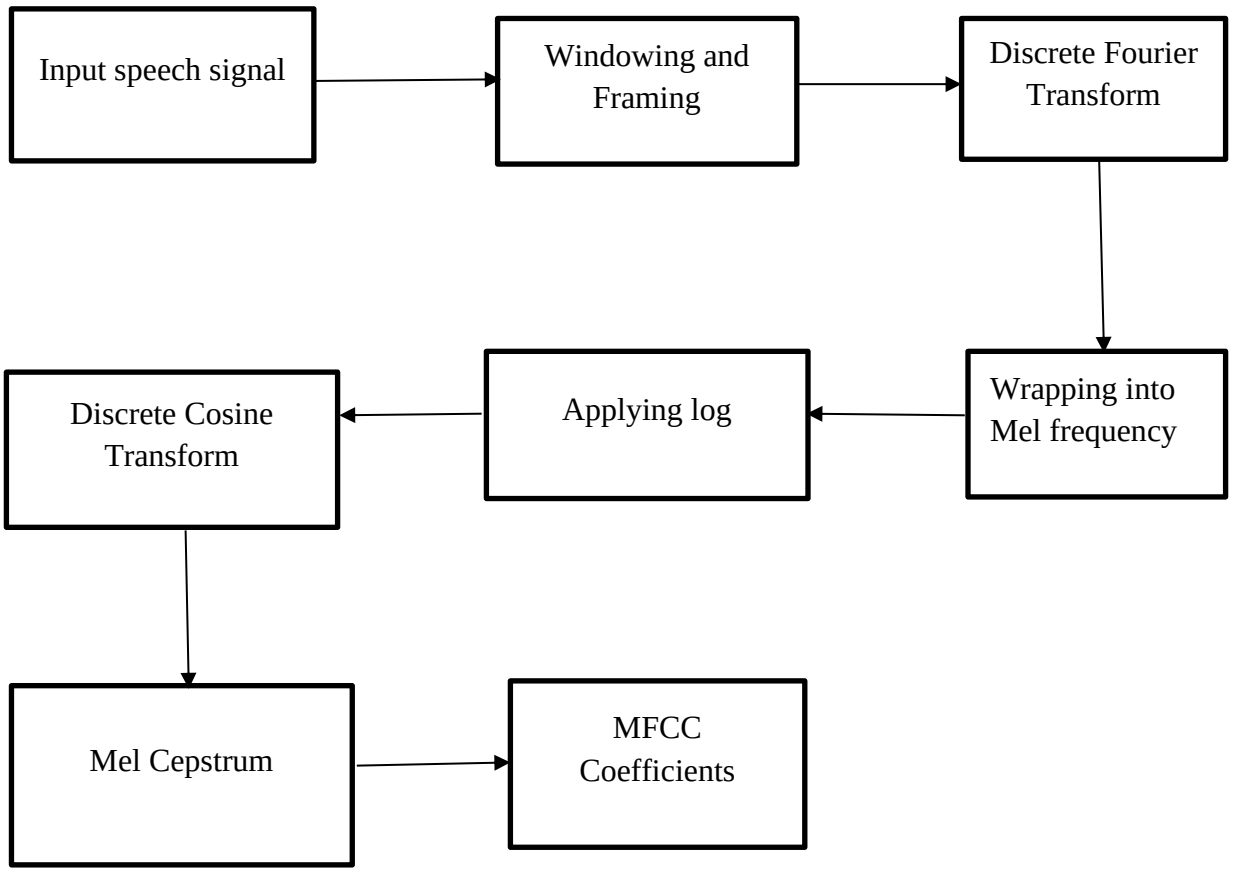


Fig 2 Steps in MFCC Feature Extraction

The function ibrosa.feature.mfcc computes 13 MFCC spectral coefficients for every audio file. The resulting coefficients then undergo augmentation which results in 39 spectral are utilized as input for the CNN model.

4.4 CNN MODEL

The CNN architecture is composed of several layers, each of which serves a distinct purpose. Convolutional layer, activation layer, pooling layer, dropout layer, fully connected

layer, and output layer are the traditional levels of a CNN. The convolutional layer helps to detect patterns from the input by applies filters. In the activation layer, activation functions are applied. The pooling layer helps in decreasing the feature map's dimensionality, and max pooling is the most widely used pooling operation that selects the maximum value within a local area of the feature map. To address the issue of overfitting, dropout layers are utilized. In dropout layers, it drops a fraction of neurons in the network. The layer that links every neuron from the layer before to the output layer, which handles classification, is known as the fully connected layer. The output layer, which usually results in a distribution of likelihood over every single class, is a softmax layer.

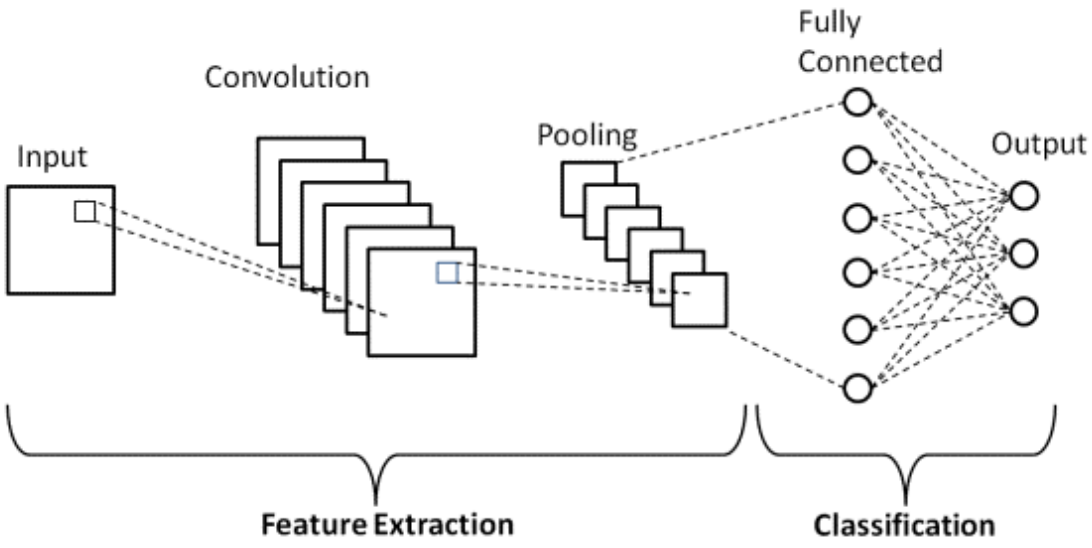


Fig 3 CNN Working

In our proposed framework, two convolutional layers with 64 feature maps each are followed by a maxing pooling layer. There are two dropout layers with a fraction of 0.6 to prevent the overfitting problem. Finally, there is a dense layer with 7 neurons that classifies the input into one among seven distinct emotions such as happy, angry, sad, disgust, neutral, fear and surprise.

4.5 TRAINING CNN MODEL

Datasets such as TESS and RAVDESS are used in the paper to demonstrate the use of a CNN for emotion classification from speech signals. With a learning rate of 0.0006 and the "categorical cross-entropy" loss function, the Adam optimizer was used to train the model. The model underwent 300 epochs of training. The training and validation loss and accuracy curves are depicted in the fig 18 and fig 19.

4.6 CLASSIFICATION OF SPEECH EMOTIONS

The From the audio file, the MFCC feature is extracted and given as an input to the proposed model to predict its emotion. The CNN model will classify the emotion into one of these categories: neutral, happy, sad, angry, disgust, fear and surprise.

5. REQUIREMENT ANALYSIS

5.1 INTRODUCTION

A requirement is a feature that the system must have or a constraint that it must to be accepted by the client. Requirement Engineering aims at defining the requirements of the system under construction. Requirement Engineering include two main activities requirement elicitation which results in the specification of the system that the client understands and analysis which in analysis model that the developer can unambiguously interpret. A requirement is a statement about what the proposed system will do.

Requirements can be divided into two major categories:

1. Functional Requirements
2. Non-Functional Requirements

5.2 FUNCTIONAL REQUIREMENTS

Speech emotion recognition systems have several functional requirements that are essential for accurately detecting and classifying different emotions from speech signals. Some of the key functional requirements for speech emotion recognition systems are:

- **Audio Input**: The system should be able to accept speech signals in different formats such as WAV, MP3, or other popular formats.

- **Feature Extraction**: The system should be able to extract relevant features from the pre-processed speech signal that can be used to identify different emotions.

- **Emotion Classification**: The system should be able to classify different emotions based on the extracted features, such as anger, joy, sadness, and surprise.

- **Real-time Processing**: The system should be able to provide real-time feedback on the detected emotions.

- **Accuracy**: The system should be able to accurately detect and classify different emotions from speech signals with a high degree of accuracy.

- **Training and Learning**: The system should be able to learn from user feedback and improve its accuracy over time.

- **Integration**: The system should be able to integrate with other applications and services, such as virtual assistants or customer service systems.

5.3 NON-FUNCTIONAL REQUIREMENTS

Speech emotion recognition systems have several non-functional requirements that are important for ensuring their reliability, usability, and effectiveness. Some of the key non-functional requirements for speech emotion recognition systems are:

- **Accuracy**: The system should be able to accurately detect and classify different emotions from speech signals.

- **Robustness**: The system should be able to perform well under different conditions, such as variations in background noise, speaker accents, and speech styles.

- **Speed**: The system should be able to process speech signals quickly and efficiently, so that it can provide real-time feedback in applications such as emotion-aware virtual assistants.

- **Scalability**: The system should be able to handle large volumes of speech data, and scale up as needed to accommodate growing demand.

- **Security**: The system should be secure and protect the privacy of users' speech data.

- **Usability**: The system should be user-friendly and easy to use, so that it can be effectively used by non-experts.

- **Reliability**: The system should be reliable and operate without errors or failures.

- **Maintainability**: The system should be easy to maintain and update, so that it can stay current with changing user needs and technology.

- **Compatibility**: The system should be compatible with different platforms and systems, so that it can be easily integrated into other applications and services.

- **Performance**: The system should perform well in terms of response time, throughput, and other relevant metrics, so that it can meet the needs of users and stakeholders.

6 ALGORITHM

6.1 CONVOLUTION NEURAL NETWORK

A Convolutional Neural Network (CNN) is a type of Deep Learning architecture commonly used for image classification and recognition tasks. It consists of multiple layers, including Convolutional layers, Pooling layers, and fully connected layers. The Convolutional layer applies filters to the input image to extract features, the Pooling layer down-samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent. services, such as virtual assistants or customer service systems.

6.1.1 CONVOLUTION LAYER

The convolution layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a colour image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimension (a height, width, and depth) which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model.

As we mentioned earlier, another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within the receptive fields of prior layers. As an example, let's assume that we're trying to determine if an image contains a bicycle. You can think of the bicycle as a sum of parts. It is comprised of a frame, handlebars, wheels, pedals, et cetera. Each individual part of the bicycle makes up a lower-level pattern in the neural net, and the combination of its parts represents a higher-level pattern, creating a feature hierarchy within the CNN. such as virtual assistants or customer service systems.

6.1.2 POOLING LAYER

Pooling layers, also known as down-sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array.

There are many types of pooling:

- **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.

- **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency and limit risk of overfitting.

6.1.3 FULLY CONNECTED LAYER

The name of the full-connected layer aptly describe itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

6.2 ACTIVATION FUNCTION

In a neural network, an activation function is used to introduce non-linearity in the output of a neuron. It takes the weighted sum of inputs to a neuron and applies a transformation to produce the output of the neuron. The purpose of the activation function is to determine whether or not the neuron should be activated based on the input received. Some commonly used activation functions include the sigmoid, ReLU, and tanh functions. The choice of activation function can have a significant impact on the performance of the neural network, and different activation functions may be more suitable for different types of problems.

6.2.1 RELU

It Stands for Rectified linear unit. ReLU is the most commonly used activation function in neural networks and The mathematical equation for ReLU is:

$$\mathbf{ReLU(x) = max(0,x)}$$

So if the input is negative, the output of ReLU is 0 and for positive values, it is x.

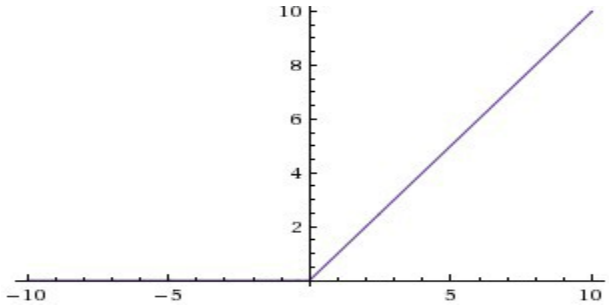


Fig 4 RELU Activation Function Graph

The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem. The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better. The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.

Though it looks like a linear function, it's not. ReLU has a derivative function and allows for backpropagation. Perceptron and convolutional neural networks.

There is one problem with ReLU. Let's suppose most of the input values are negative or 0, the ReLU produces the output as 0 and the neural network can't perform the back propagation. This is called the Dying ReLU problem. Also, ReLU is an unbounded function which means there is no maximum value.

6.2.2 SOFTMAX

The softmax function is shown below, where z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

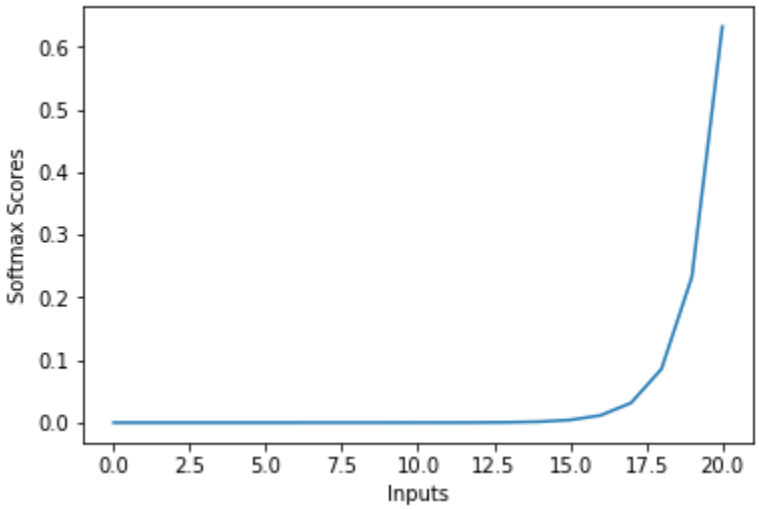


Fig 5 Softmax Activation Function Graph

The softmax function is sometimes called the soft argmax function, or multi-class logistic regression. This is because the softmax is a generalization of logistic regression that can be used for multi-class classification, and its formula is very similar to the sigmoid function which is used for logistic regression. The softmax function can be used in a classifier only when the classes are mutually exclusive.

Softmax activation function is non-linear in nature. The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input. It would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs. If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

6.3 OPTIMIZER

While training the deep learning optimizers model, we need to modify each epoch's weights and minimize the loss function. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rates. Thus, it helps in reducing the overall loss and improving accuracy. The problem of choosing the right weights for the model is a daunting task, as a deep learning model generally consists of millions of parameters. It raises the need to choose a suitable optimization algorithm for your application.

6.3.1 ADAM OPTIMIZER

Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used in deep learning for updating the parameters of the neural network during training. It combines the benefits of two other popular optimization algorithms, namely Adagrad and RMSprop.

Adam optimizer maintains an exponentially decaying average of past gradients and their squared values, and uses them to update the parameters. It also applies bias correction to the moving averages to reduce bias towards zero, especially during the early stages of training when the number of updates is small. Adam optimizer is known for its fast convergence and good performance on a wide range of deep learning tasks, making it a popular choice in the field.

To use the Adam optimizer in a CNN, it can be implemented using libraries such as TensorFlow or PyTorch, and the optimizer can be configured with parameters such as the learning rate, momentum, and decay rates.

7 UML DIAGRAMS

7.1 WHAT IS A UML DIAGRAM?

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

7.1.1 WHAT IS MEANT BY UML

UML stands for Unified Modeling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

- Improved integration between structural models like class diagrams and behavior models like activity diagrams.

- Added the ability to define a hierarchy and decompose a software system into components and sub-components.

- The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed statechart diagrams to state machine diagrams, also known as state diagrams.

7.1.2 TYPES OF UML DIAGRAMS

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing and deployment.

These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML diagrams

- Class diagram
- Package diagram
- Object diagram
- Component diagram
- Composite structure diagram
- Deployment diagram

Behavioral UML diagrams

- Activity diagram
- Sequence diagram
- Use case diagram
- State diagram
- Communication diagram
- Interaction overview diagram
- Timing diagram

7.2 CLASS DIAGRAM

The Unified Modeling Language (UML) can help you model systems in various ways. One of the more popular types in UML is the class diagram. Popular among software engineers to document software architecture, class diagrams are a type of structure diagram because they describe what must be present in the system being modeled. No matter your level of familiarity with UML or class diagrams, our UML software is designed to be simple and easy to use.

UML was set up as a standardized model to describe an object-oriented programming approach. Since classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

7.2.1 BENEFITS OF CLASS DIAGRAM

Class diagrams offer a number of benefits for any organization. Use UML class diagrams to: Illustrate data models for information systems, no matter how simple or complex. Better understand the general overview of the schematics of an application. Visually express any specific needs of a system and disseminate that information throughout the business. Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure. Provide an implementation-independent description of types used in a system that are later passed between its components.

7.2.2 BASIC COMPONENTS OF CLASS DIAGRAM

The standard class diagram is composed of three sections:

- **Upper section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.

- **Middle section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.

- **Bottom section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

Member access modifiers

All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:

- Public (+)
- Private (-)
- Protected (#)
- Package (~)
- Derived (/)
- Static (underlined)

Components:

- **Classes:**
  A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior.

They're represented by a rectangle that includes rows of the class name, its attributes, and its operations. When you draw a class in a class diagram, you're only required to fill out the top row—the others are optional if you'd like to provide more detail.

- **Name:**
  The first row in a class shape.

- **Attributes:**
  The second row in a class shape. Each attribute of the class is displayed on a separate line.

- **Methods:**
  The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.

- **Signals:**
  Symbols that represent one-way, asynchronous communications between active objects.

- **Data types:**
  Classifiers that define data values. Data types can model both primitive types and enumeration

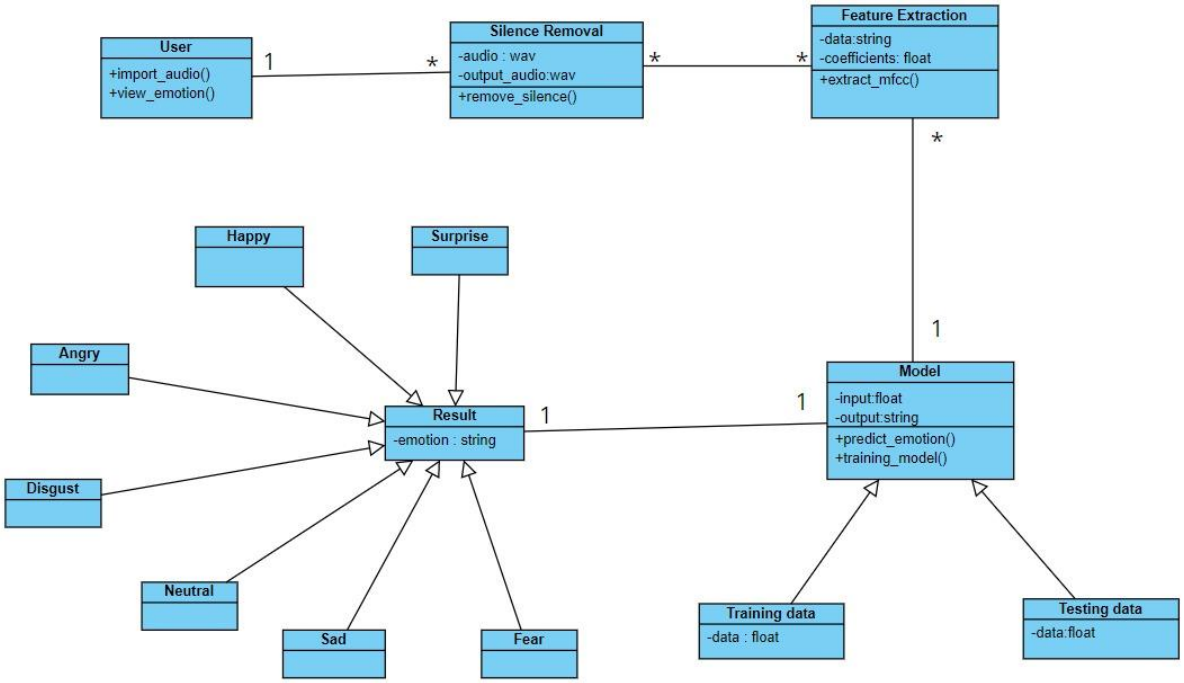


Fig 6 Class diagram of our model

7.3 USECASE DIAGRAM

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent: Scenarios in which your system or application interacts with people, organizations, or external systems. Goals that your system or application helps those entities (known as actors) achieve.

7.3.1 WHEN TO APPLY USECASE DIAGRAM

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend

that use case diagrams be used to supplement a more descriptive textual use case.UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

7.3.2 UML USECASE DIAGRAM ARE IDEAL FOR

- Representing the goals of system-user interactions.
- Defining and organizing functional requirements in a system.
- Specifying the context and requirements of a system.
- Modeling the basic flow of events in a use case.

7.3.3 COMPONENTS

- **Actors**: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.

- **System**: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.

- **Usecases:** Horizontally shaped ovals that represent the different uses that a user might have.

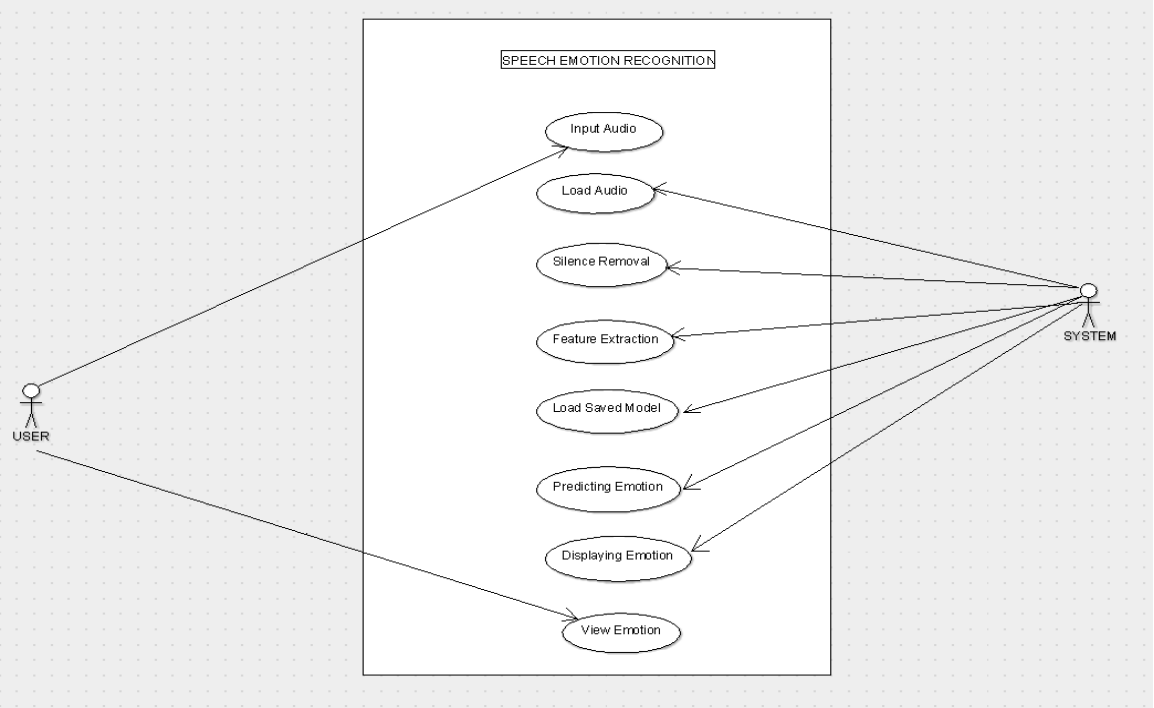


Fig 7 Use case diagram of our model

7.4 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

7.4.1 BENEFITS OF SEQUENCE DIAGRAM

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.

- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

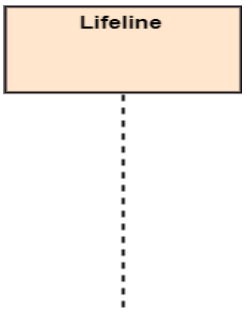
7.4.2 PURPOSE OF SEQUENCE DIAGRAM

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction.

7.4.3 NOTATIONS OF A SEQUENCE DIAGRAM

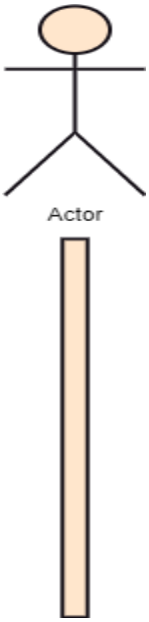
- **Lifeline:**

  An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram.



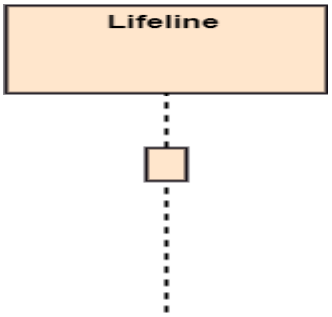
- **Actor:**

  A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.



- **Activation:**

  It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.



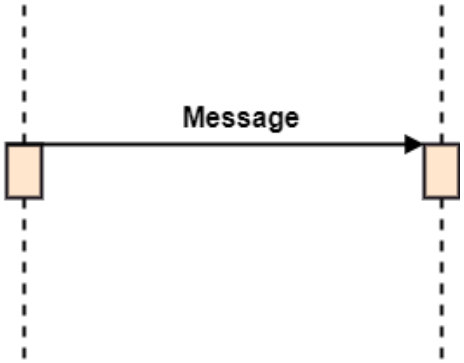
- **Messages:**

  The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

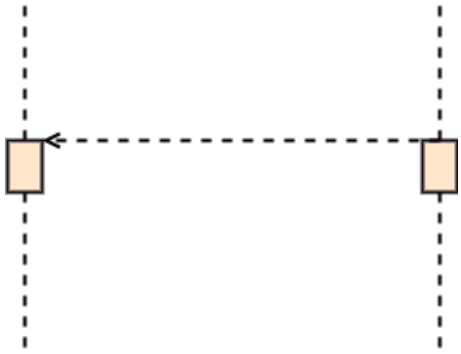
- **Call Message:**

  It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.



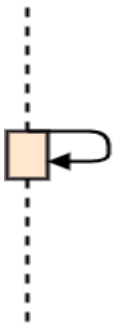
- **Return Message:**

  It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.



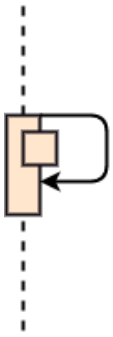
- **Self Message:**

  It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.



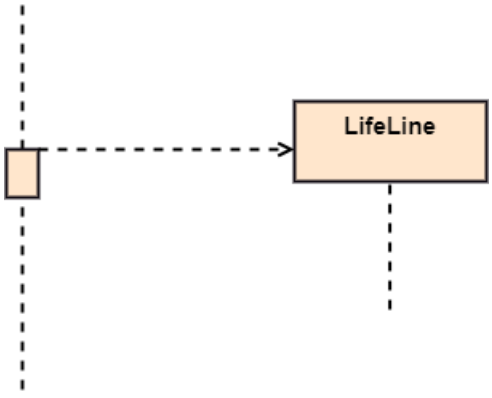
- **Recursive Message:**

  A self message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self message as it represents the recursive calls.



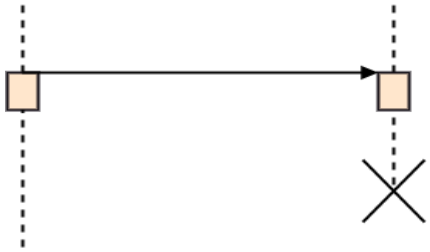
- **Create Message:**

  It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.



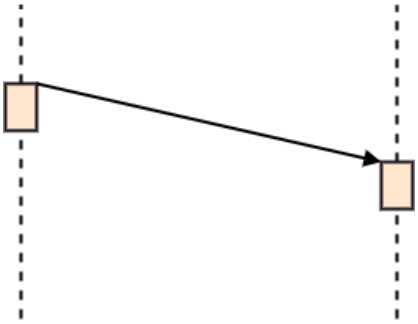
- **Destroy Message:**

  It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.



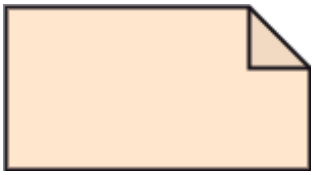
- **Duration Message:**

  It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modeling a system.



- **Note**

  A note is the capability of attaching several remarks to the element. It basically carries useful information for the modelers.



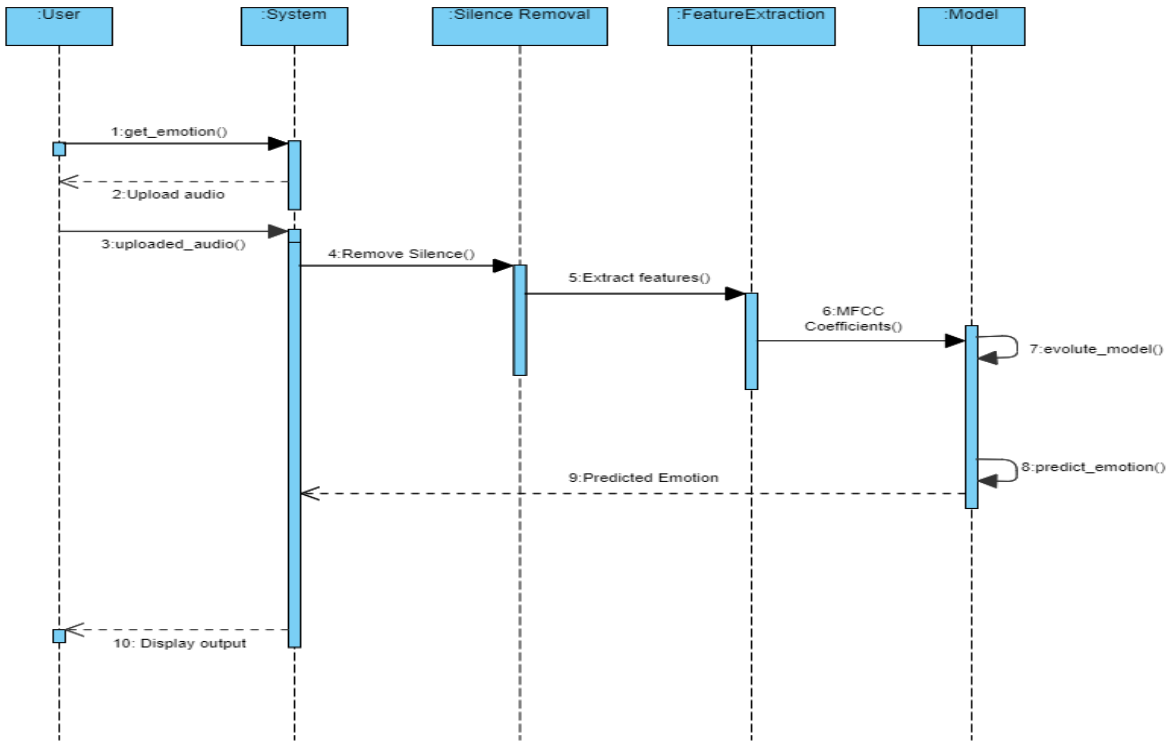


Fig 8 Sequence Diagram Of Our Model

7.5 ACTIVITY DIAGRAM

The Unified Modeling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behavior diagrams. Activity diagrams, along with use case and state machine diagrams, are considered behavior diagrams because they describe what must happen in the system being modeled.

Stakeholders have many issues to manage, so it's important to communicate with clarity and brevity. Activity diagrams help people on the business and development sides of an organization come together to understand the same process and behavior. You'll use a set of specialized symbols—including those used for starting, ending, merging, or receiving steps in the flow—to make an activity diagram, which we'll cover in more depth within this activity diagram guide.

7.5.1 BENEFITS OF ACTIVITY DIAGRAM

Activity diagrams present a number of benefits to users. Consider creating an activity diagram to:

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation.

7.5.2 BASIC COMPONENTS OF ACTIVITY DIAGRAM

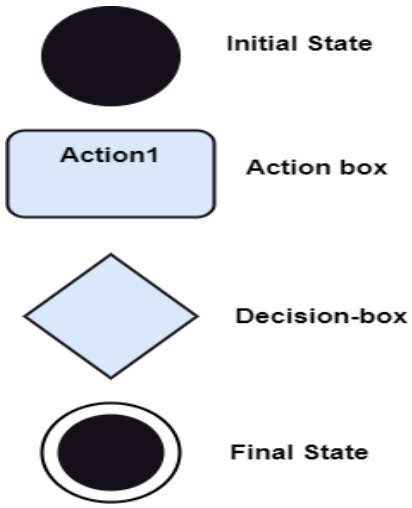
Before you begin making an activity diagram, you should first understand its makeup. Some of the most common components of an activity diagram include:

- **Action:** A step in the activity wherein the users or software perform a given task.
- **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.

- **Control flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

7.5.3 NOTATION OF AN ACTIVITY DIAGRAM:

- **Initial State:** It depicts the initial stage or beginning of the set of actions.
- **Final State:** It is the stage where all the control flows and object flows end.
- **Decision Box:** It makes sure that the control flow or object flow will follow only one path.
- **Action Box:** It represents the set of actions that are to be performed.



7.5.4 WHY WE USE ACTIVIVTY DIAGRAM

An event is created as an activity diagram encompassing a group of nodes associated with edges. To model the behavior of activities, they can be attached to any modeling element. It can model use cases, classes, interfaces, components, and collaborations.

It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering. It does not include the message part, which means message flow is not represented in an activity diagram. It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities.

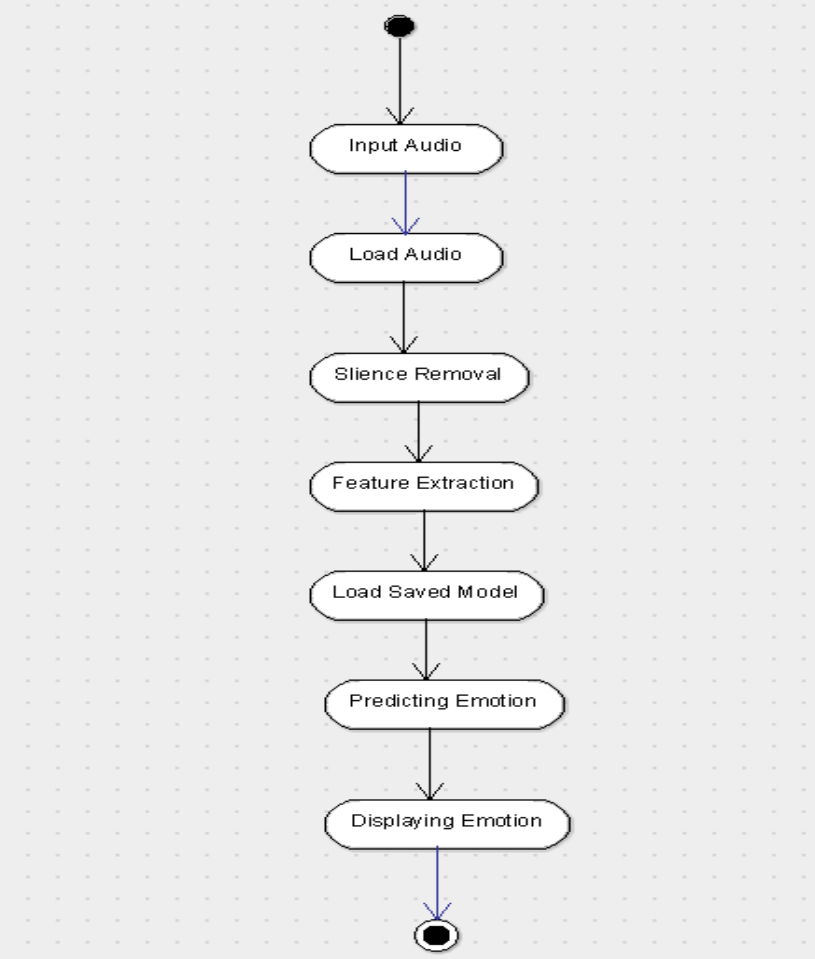


Fig 9 Activity diagram of our model

7.6 DEPLOYEMNT DIAGRAM

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.

Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

7.6.1 WHEN TO USE DEPLOYMENT DIAGRAM

- What existing systems will the newly added system need to interact or integrate with?
- How robust does system need to be (e.g., redundant hardware in case of a system failure)?
- What and who will connect to or interact with system, and how will they do it
- What middleware, including the operating system and communications approaches and protocols, will system use?
- What hardware and software will users directly interact with (PCs, network computers, browsers, etc.)?
- How will you monitor the system once deployed?
- How secure does the system needs to be (needs a firewall, physically secure hardware, etc.)?

7.6.2 PURPOSE OF DEPLOYMENT DIAGRAM

- They show the structure of the run-time system
- They capture the hardware that will be used to implement the system and the links between different items of hardware.

- They model physical hardware elements and the communication paths between them
- They can be used to plan the architecture of a system.
- They are also useful for Document the deployment of software components or nodes

7.6.3 ELEMENTS OF DEPLOYMENT DIAGRAM

A variety of shapes make up deployment diagrams. This list offers an overview of the basic elements you may encounter, and you can see most of these items illustrated in the image below.

- **Artifact:** A product developed by the software, symbolized by a rectangle with the name and the word “artifact” enclosed by double arrows.

- **Association**: A line that indicates a message or other type of communication between nodes.

- **Component:** A rectangle with two tabs that indicates a software element.

- **Dependency:** A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.

- **Interface:** A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.

- **Node:** A hardware or software object, shown by a three-dimensional box.

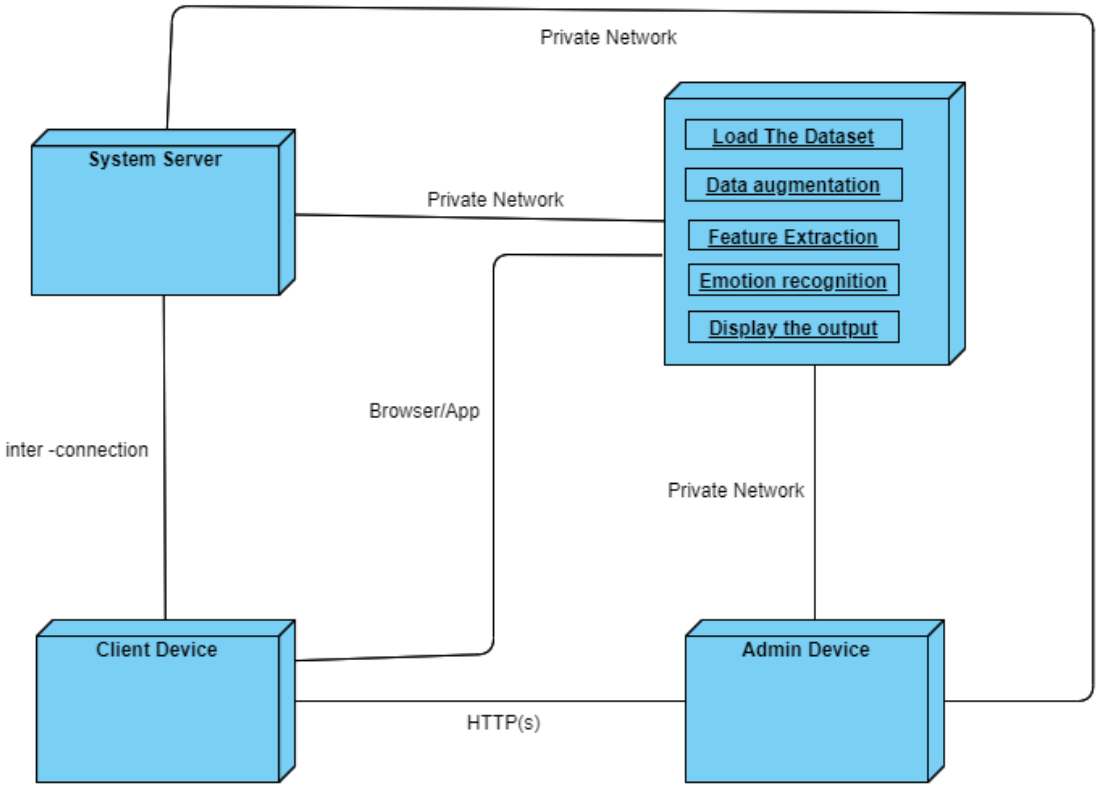


Fig 10 Deployment Diagram of our model

8 IMPLEMENTATION AND CODE

8.1 PYTHON LIBRARIES

- **LIBROSA**

  Librosa is a python package for music and audio analysis. librosa.display is used to show various spectrograms. Librosa.feature is used to extract different types of features from audio such as Mel-Frequency Cepstral Coefficients (MFCC), Mel spectrogram, chromograms etc.

- **MATPLOTLIB**

  Matplotlib is an amazing visualization llibrary in Python for 2D plots of arrays. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- **SEABORN**

  Seaborn is a python data visualization library based on matplotlib used to plot the graphs. It provides beautiful default styles and color palettes to make statistical plots more attractive. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **OS**

  OS module in python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

- **NUMPY**

  Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **PANDAS**

  Pandas is a software library written for the Python programming language for data manipulation and analysis.

- **IPYTHON**

  IPython means interactive Python. It is an interactive command-line terminal for Python. It will provide an IPython terminal and web-based (Notebook) platform for Python computing. It has more advanced features than the Python standard interpreter and will quickly execute a single line of Python code.

8.1.1 CODE FOR IMPORTING LIBRARIES

```
import librosa
import librosa.display
import pandas as pd
import numpy as np
import os
import IPython.display as ipd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

8.2 DATASETS

8.2.1 RAVDESS DATASET

RAVDESS, which stands for Ryerson Audio-Visual Database of Emotional Speech and Song, comprises 1440 audio files featuring 24 professional actors with an equal gender split. The actors speak two statements in a neutral North American accent, and the statements are matched lexically. The speech includes expressions of calm, happiness,

sadness, anger, fear, surprise, and disgust. Each expression is recorded at two different emotional intensities (normal and strong), and there is also an additional neutral expression included in the database.

Each file in the RAVDESS database is assigned a unique filename consisting of a seven-part numerical identifier.

These seven identifiers are as follows:

- Modality of the file (01 → full-AV, 02 → video-only, 03 → audio-only)
- Vocal channel (01 → speech, 02 → song)
- Emotion of the audio (01 → neutral, 02 → calm, 03 → happy, 04 → sad, 05 → angry, 06 → fearful, 07 → disgust, 08 → surprised)
- Intensity of the emotion (01 → normal, 02 → strong)
- Statement of the speech (01 → "Kids are talking by the door", 02 → "Dogs are sitting by the door")
- Repetition (01 → 1st repetition, 02 → 2nd repetition)
- Actors (from 01 to 24, male for odd and female for even).

8.2.1.1 CODE

```
rav = 'C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data/'
dir_rav = os.listdir(rav)
emotion=[]
gender=[]
path=[]

for i in dir_rav:
    fname = os.listdir(rav+i)
    print(fname)
    for f in fname:
```

```
            part = f.split('.')[0].split('-')
            emotion.append(int(part[2]))
            temp = int(part[6])
            if temp%2 == 0:
               temp="female"
            else:
               temp="male"
         gender.append(temp)
         path.append(rav+i+'/'+f)
rav_df = pd.DataFrame(emotion)

     rav_df  =
rav_df.replace({1:'neutral',2:'calm',3:'happy',4:'sad',5:'angry',6:'fear',7:'disgust',8:’surprise’
})
rav_df = pd.concat([pd.DataFrame(gender),rav_df],axis=1)
rav_df.columns = ['gender','emotion']
rav_df["labels"] = rav_df.gender + "_" + rav_df.emotion
rav_df = pd.concat([rav_df, pd.DataFrame(path, columns=['path'])],axis=1)
rav_df.head()
```

	gender	emotion	labels	path
0	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
2	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
3	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
4	male	calm	male_calm	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
...	...	...	...	...
1435	female	surprise	female_surprise	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1436	female	surprise	female_surprise	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1437	female	surprise	female_surprise	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1438	female	surprise	female_surprise	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1439	female	surprise	female_surprise	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...

1440 rows × 4 columns

8.2.2 TESS DATASET

The Toronto Emotional Speech Set, abbreviated as TESS, is a compilation of 2800 audio recordings saved in the WAV format. The dataset includes 200 objective words uttered by two actresses, aged 26 and 64, in the carrier phrase "Say the word _" depicting seven emotions: anger, disgust, fear, happiness, pleasant surprise, sadness, and neutrality. The data is arranged in folders specific to each female actor, and within each folder are subfolders that contain the audio files for all 200 target words.

8.2.2.1 CODE

```
dir_tess = os.listdir(tess)
path=[]
emotion=[]
e=[]
for i in dir_tess:
    fname=os.listdir(tess+i)
    for f in fname:
        if i=='OAF_angry' or i=='YAF_angry':
            emotion.append('female_angry')
            e.append("angry")
        elif i=='OAF_disgust' or i=='YAF_disgust':
            emotion.append('female_disgust')
            e.append('disgust')
        elif i=='OAF_Fear' or i=='YAF_fear':
            emotion.append('female_fear')
            e.append("fear")
        elif i=='OAF_happy' or i=='YAF_happy':
            emotion.append('female_happy')
            e.append("happy")
        elif i=='OAF_neutral' or i=='YAF_neutral':
```

```
        emotion.append('female_neutral')
        e.append("neutral")
    elif i=='OAF_Pleasant_surprise' or i=='YAF_pleasant_surprised':
        emotion.append('female_surprise')
        e.append('surprise')
    elif i=='OAF_Sad' or i=='YAF_sad':
        emotion.append('female_sad')
        e.append("sad")
    else:
        emotion.append('unknown')
         e.append("unknown")
          path.append(tess + i + '/' + f)
tess_df = pd.DataFrame(list(zip(e,emotion)), columns=['emotion','labels'])
tess_df["gender"]="female"
tess_df = pd.concat([tess_df, pd.DataFrame(path, columns=['path'])],axis=1)
tess_df
```

	emotion	labels	gender	path
0	fear	female_fear	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
1	fear	female_fear	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
2	fear	female_fear	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
3	fear	female_fear	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
4	fear	female_fear	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
...	...	...	...	...
2795	sad	female_sad	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
2796	sad	female_sad	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
2797	sad	female_sad	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
2798	sad	female_sad	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
2799	sad	female_sad	female	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...

2800 rows × 4 columns

8.2.3 COMBINING RAVDESS AND TESS DATASET

```
df = pd.concat([rav_df, tess_df], axis=0)
df.index=np.arange(len(df))
df
```

	gender	emotion	labels	path
0	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
1	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
2	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
3	male	neutral	male_neutral	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
4	male	calm	male_calm	C:/Users/HAFSA KURSHEED/Downloads/Ravdess/data...
...	...	...	...	...
4235	female	sad	female_sad	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
4236	female	sad	female_sad	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
4237	female	sad	female_sad	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
4238	female	sad	female_sad	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...
4239	female	sad	female_sad	C:/Users/HAFSA KURSHEED/Downloads/Tess/TESS To...

4240 rows × 4 columns

8.3 REMOVING UNNECESSARY VALUES

The audio samples whose emotion is calm and unknown are removed as we are detecting only the emotions neutral, happy, sad, angry, fear, disgust and surprise.

```
ind = df[(df['emotion']=="calm")].index
df.drop(ind, inplace=True)
sns.countplot(x=df["emotion"]);
```

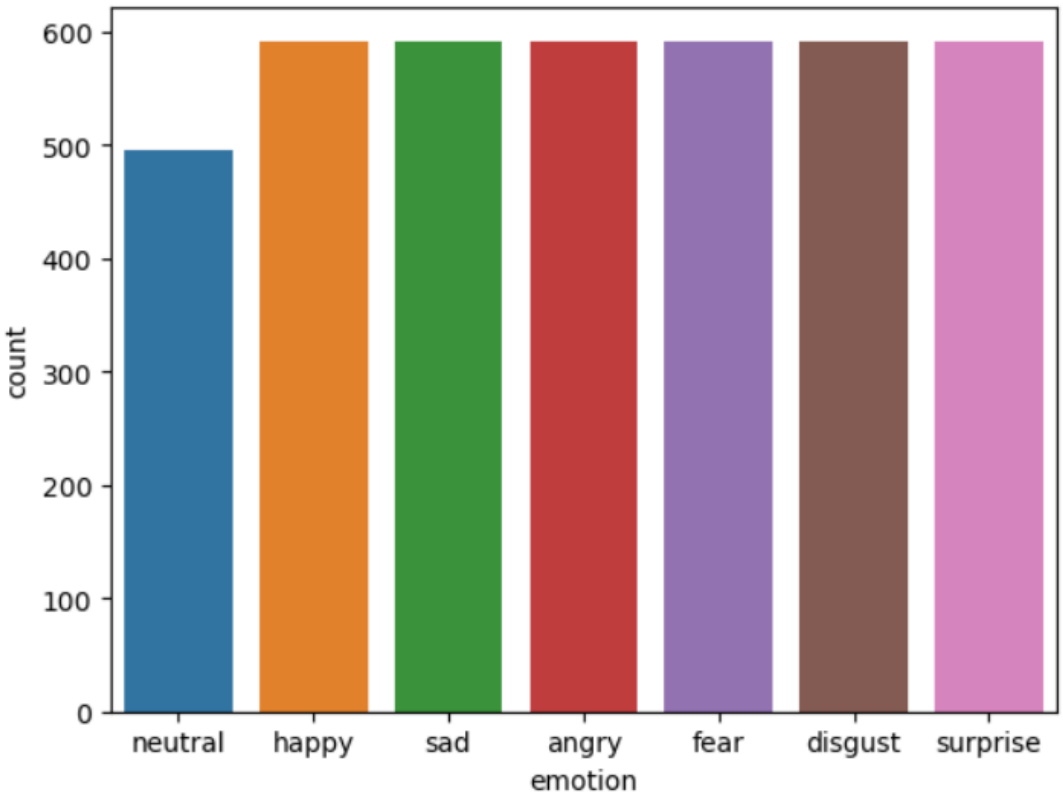


Fig 11 Countplot for different labelled audio samples

8.4 VISUALIZATION OF AUDIO

8.4.1 WAVEPLOT

Waveplot is used to plot waveform of amplitude vs time where the x-axis is time and y-axis is amplitude. Create_waveplot is a function to display waveplot of the given audio file. It uses librosa module of the python for displaying waveplot.

```
def create_waveplot(f,t):
    data,sr = librosa.load(f)
    plt.figure(figsize=(15,5))
    librosa.display.waveshow(data, sr=sr);
    plt.xlabel("Time")
    plt.ylabel("Amplitude")
```

```
plt.title(t)
plt.show()
```

Ravdess Dataset sample:

```
fname = rav + 'Actor_14/03-01-06-02-02-02-14.wav'
create_waveplot(fname,"waveplot")
```

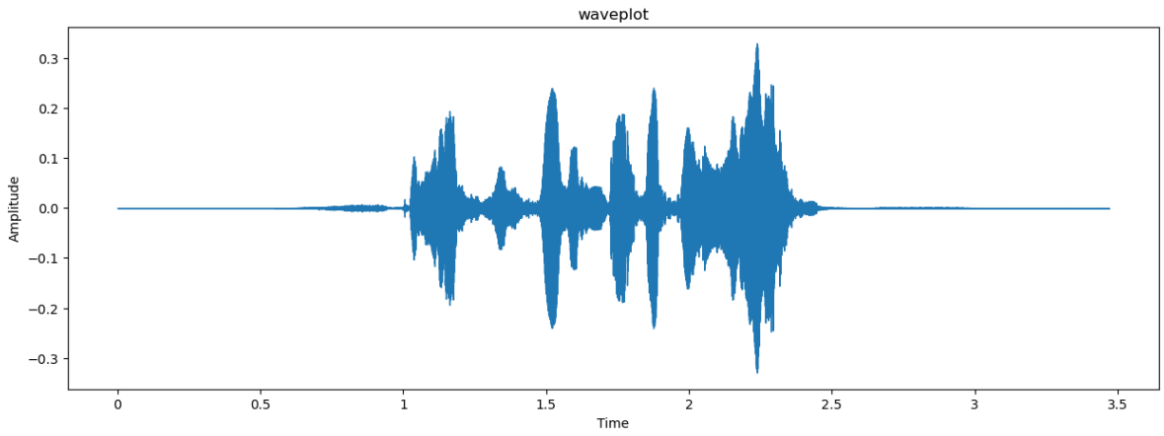


Fig 12 Waveplot for ravdess audio sample

Tess Dataset Sample:

```
fname = tess + 'YAF_fear/YAF_dog_fear.wav'
create_waveplot(fname,"waveplot")
```

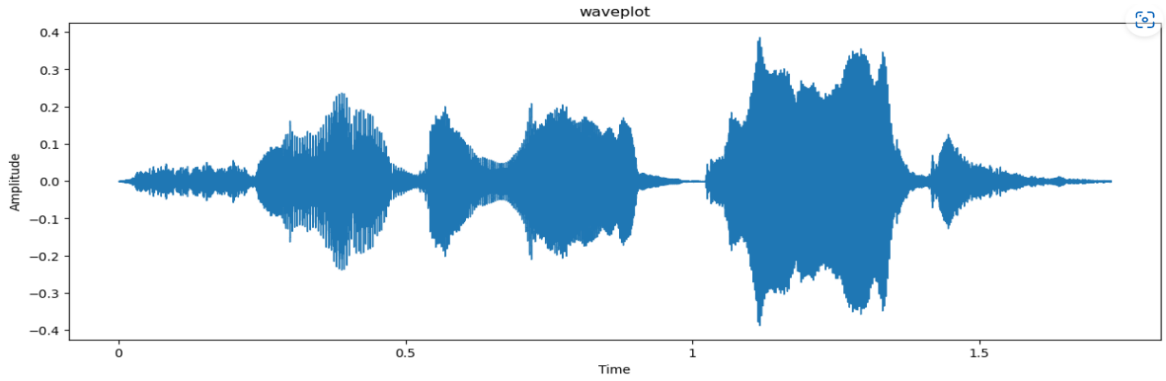


Fig 13 Waveplot for tess audio sample

8.4.2 SPECTOGRAM

A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given audio signals.

The spectrogram is calculated by the magnitude squared of the STFT. STFT stands for short term fourier transform. librosa.stft is a function in python used to calculate the stft on the given audio sample.

Short Term Fourier Transform:

The Short-Term Fourier Transform (STFT) is a signal processing technique that analyzes the frequency content of a signal over time by dividing it into short time intervals and performing a Fourier Transform on each interval. The STFT provides a time-frequency representation of the signal, allowing us to accurately analyze the frequency content of the signal at different time intervals.

```
data,sr = librosa.load(df["path"][0])
frame_size = 2048
hop_size =512
data_stft = librosa.stft(data, n_fft=frame_size, hop_length=hop_size)
```

librosa.stft returns two dimensional array where one dimension is frequency bins and the other is number of frames. The frequency bins and number of frames can be calculated using the following mathematical formulae.

$$\text{Frequency bins} = (\text{frame size}/2) + 1$$

$$\text{Frames} = ((\text{samples} - \text{frame size})/\text{hop size}) + 1$$

Calculating spectrogram:

```
y=np.abs(data_stft)**2
```

The dimensions are same as above but it changes complex data type to float.

Visualizing Spectogram:

```
def display_spectogram(data, sr, hl, ya):
    plt.figure(figsize=(15,5))
    librosa.display.specshow(data, sr=sr, hop_length = hl, x_axis="time", y_axis = ya)
    plt.title("Spectogram")
    plt.colorbar(format="%+2.f")
    plt.show()
y1=librosa.power_to_db(y)
display_spectogram(y1,sr,hop_size,"log")
```

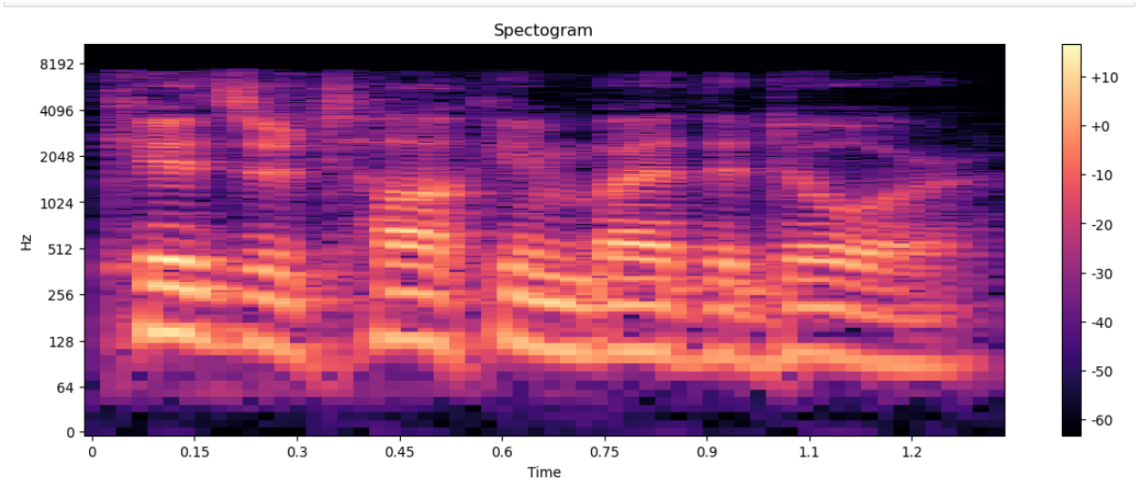


Fig 14 Spectogram representation of audio sample

8.5 REMOVING SILENCE

Removing silence is an important pre-processing step in speech emotion recognition (SER) tasks, as it can improve the accuracy and robustness of the emotion classification algorithms. Silence or background noise in speech signals can interfere with the accurate classification of emotions and can lead to misclassification. Therefore, it is important to remove the silence before the speech signal is analyzed.

Waveplot without silence:

```
data1, _ = librosa.effects.trim(data, top_db = 30)
plt.figure(figsize=(15,5))
librosa.display.waveshow(data1);
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Audio without silence")
plt.show()
```

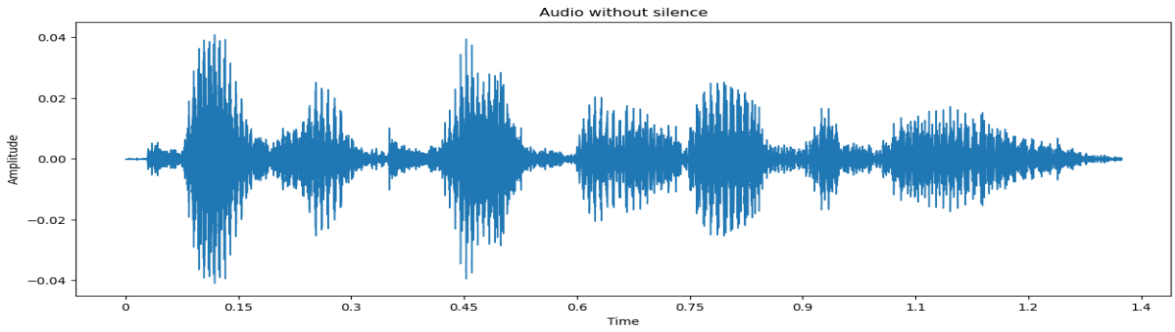


Fig 15 Waveplot of audio containing no silence

Audio Sample duration before and after removing silence:

The duration of the original audio is 3sec but after removing the silence from the audio sample its duration reduces to 1sec.

```
print(librosa.get_duration(data), librosa.get_duration(data1));
```

```
3.3033560090702947 1.3235374149659864
```

8.6 DATA AUGMENTATION

Data augmentation is a popular technique in deep learning to increase the size of the training data by generating new data from existing data. In speech emotion recognition, data augmentation can help to improve the performance of deep learning models by increasing the amount and variety of training data.

In this system, we have applied various signal processing techniques like pitch shifting, time stretching, adding noise, changing dynamic range to the original speech signals for the process of Data Augmentation.

```
def noise(data):
    noise_amp = 0.005*np.random.uniform()*np.amax(data)
    data = data.astype("float64")+noise_amp*np.random.normal(size=data.shape[0])
    return data


def shift(data):
    s_range = int(np.random.uniform(low=-5,high=5)*500)
    return np.roll(data,s_range)



def stretch(data,rate=0.8):
    data = librosa.effects.time_stretch(data,rate)
    return data


def dyn_change(data):
    dyn_change = np.random.uniform(low=1.5,high = 3)
    return (data*dyn_change)
```

```
def speedNpitch(data):
    length_change=np.random.uniform(low=0.8,high=1)
    speed_fac = 1.0/length_change
    tmp=np.interp(np.arange(0,len(data),speed_fac),np.arange(0,len(data)),data)
    minlen = min(data.shape[0],tmp.shape[0])
    data*=0
    data[0:minlen]=tmp[0:minlen]
    return data
```

8.7 MFCC FEATURE EXTRACTION

MFCC stands for Mel-Frequency Cepstral Coefficients which have 39 features. The feature count is small enough to force us to learn the information of the audio. 13 parameters are related to the amplitude of frequencies. It provides us enough frequency channels to analyze the audio. MFCC is an audio feature extraction technique which extracts speaker specific parameters from the speech. Mel-Frequency Cepstral Coefficients (MFCC) is the most popular and dominant method to extract spectral features for speech by the use of perceptually based Mel spaced filter bank processing of the Fourier Transformed signal.

Visualize MFCC:

```
def plot_mfcc(m):
    plt.figure(figsize=(15,5))
    librosa.display.specshow(m, x_axis="time")
    plt.ylabel('MFCC COEFFS')
    plt.title('MFCC');
    plt.colorbar()
    plt.show()
plot_mfcc(mfccs)
```

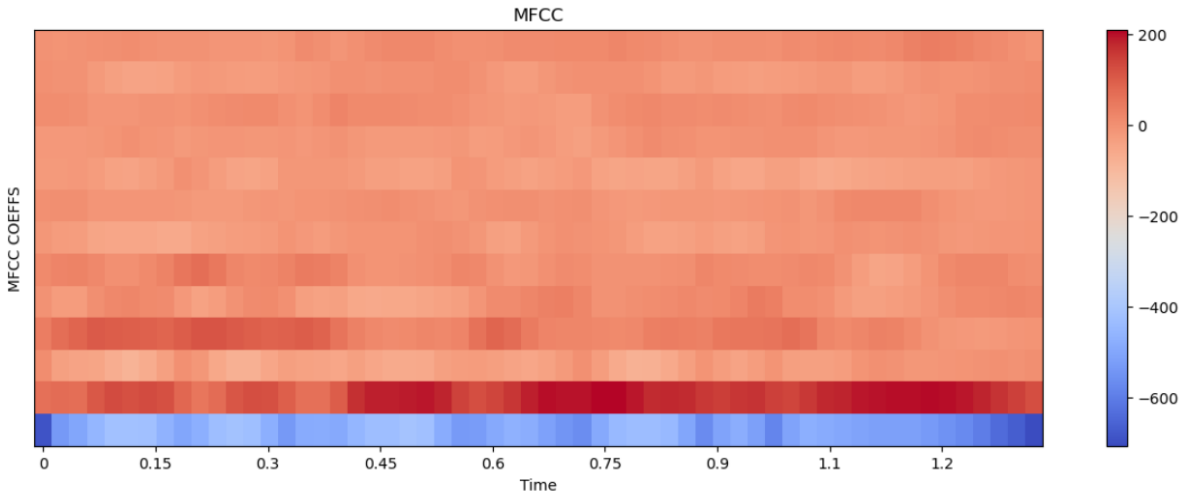


Fig 16 Visualization of MFCC Coefficients

Function to find MFCC:

```
def extract(i):
    data,sr = librosa.load(df["path"][i])
    data, _ = librosa.effects.trim(data, top_db = 30)
    mfccs = np.mean(librosa.feature.mfcc(data, n_mfcc=13, sr=sr),axis=0)
    delta_mfccs = librosa.feature.delta(mfccs)
    delta2_mfccs = librosa.feature.delta(mfccs, order=2)
    mfcc = np.concatenate((mfccs, delta_mfccs, delta2_mfccs),axis=0)
    return mfcc.
```

8.8 SPLITTING THE DATASET

Splitting the dataset refers to dividing the available data into separate subsets for training and testing.

Splitting the dataset is important to evaluate the performance of a deep learning model on data that it has not been trained on, and to avoid overfitting or underfitting**.**

Dataset is splitted using model_selection module from sklearn package.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(d,d_emotion,test_size=0.25,random_state=122)
x_train
```

```
x_train
```

	0	1	2	3	4	5	6	7	8	9	...	677	678	679	680	681	682
16663	-31.037203	-20.895317	-19.664120	-23.783195	-25.207500	-27.565134	-28.369850	-27.923107	-28.762554	-30.024736	...	0.0	0.0	0.0	0.0	0.0	0.0
1954	-23.336184	-25.878662	-31.698734	-32.114845	-32.014256	-32.385677	-29.775095	-27.441162	-28.350769	-30.374224	...	0.0	0.0	0.0	0.0	0.0	0.0
9719	-23.465466	-27.914783	-33.212029	-30.853342	-32.593166	-33.321934	-33.082088	-30.628536	-28.502831	-29.568007	...	0.0	0.0	0.0	0.0	0.0	0.0
17812	-23.385105	-24.008118	-24.272106	-22.036165	-23.709698	-24.017056	-23.161217	-24.286440	-24.349281	-20.041727	...	0.0	0.0	0.0	0.0	0.0	0.0
17515	-29.321123	-29.321430	-29.397356	-29.584557	-29.550476	-26.898716	-22.629751	-20.021568	-21.618086	-25.109865	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6974	-35.339895	-32.683412	-32.434429	-32.915506	-32.899821	-32.245120	-33.319554	-33.456592	-31.681519	-31.609137	...	0.0	0.0	0.0	0.0	0.0	0.0
15692	-33.441051	-27.625681	-26.879005	-28.139992	-31.321980	-33.511940	-34.312943	-34.024673	-33.893467	-35.383041	...	0.0	0.0	0.0	0.0	0.0	0.0
12214	-46.201820	-41.987068	-45.224060	-49.515591	-48.975567	-48.603458	-49.570065	-49.699104	-50.247917	-49.767532	...	0.0	0.0	0.0	0.0	0.0	0.0
2442	-35.832439	-35.852863	-36.875137	-34.816795	-34.212376	-34.977566	-32.807278	-32.686443	-34.681812	-34.800163	...	0.0	0.0	0.0	0.0	0.0	0.0
7663	-32.611144	-24.079675	-20.930168	-22.378788	-27.161753	-30.904723	-27.710096	-25.569648	-21.590772	-19.647538	...	0.0	0.0	0.0	0.0	0.0	0.0

18216 rows × 687 columns

8.9 LABEL ENCODING

Label Encoding is a process of converting categorical data into numerical values. Each unique category value is assigned a unique integer value, typically used for ordinal data with inherent order or ranking between the categories. It can be performed using programming libraries such as scikit-learn in Python.

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
from keras.utils import to_categorical
lb = LabelEncoder()
ytrain = np_utils.to_categorical(lb.fit_transform(y_train))
ytest = np_utils.to_categorical(lb.fit_transform(y_test))
lb.classes_
```

```
 array(['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'],
      dtype=object)
lb.transform(['happy','angry','sad','fear','surprise','disgust','neutral'])
 array([3, 0, 5, 2, 6, 1, 4])
lb.inverse_transform([3, 0, 5, 2, 6, 1, 4])
 array(['happy', 'angry', 'sad', 'fear', 'surprise', 'disgust', 'neutral'],
      dtype=object)
```

8.10 EXPANDING DIMENSIONS

Expanding dimensions refers to increasing the number of dimensions in a tensor or array, typically by adding a new axis. This is a common operation in machine learning and data analysis, as it can allow for more complex computations and analysis.

```
xtrain = x_train.iloc[:,:].values.reshape(len(x_train),229,3,1)
xtrain.shape
(18216, 229, 3, 1)

xtest = x_test.iloc[:,:].values.reshape(len(x_test),229,3,1)
xtest.shape
(6072, 229, 3, 1)
```

8.11 CNN MODEL

CNN stands for Convolutional Neural Network, which is a type of neural network architecture commonly used in image, speech and video recognition tasks. CNNs are designed to automatically learn and extract relevant features from raw data by using a series of convolutional layers, pooling layers, and activation functions Tensorflow is used for building the CNN model.

```
import keras
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
model = Sequential()
model.add(Conv2D(filters = 64, kernel_size = 3, padding = "same", activation = "relu",
input_shape = xtrain[1].shape))
model.add(Conv2D(filters = 64, kernel_size = 3, padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size=(2,2),padding = "same",strides=2))
model.add(Flatten())
model.add(Dropout(0.6))
model.add(Dense(128, activation = "relu"))
model.add(Dropout(0.6))
model.add(Dense(7, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.0006, decay=0.0)
model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
hist = model.fit(xtrain,ytrain,validation_data=(xtest,ytest),epochs=300,batch_size=20)
```

9 EXPERIMENTAL ANALYSIS AND RESULTS

9.1 ARCHITECTURE OF OUR MODEL

The CNN architecture of our model is as follows.

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 229, 3, 64)        640

 conv2d_3 (Conv2D)           (None, 229, 3, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 115, 2, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 14720)             0

 dropout_2 (Dropout)         (None, 14720)             0

 dense_2 (Dense)             (None, 128)               1884288

 dropout_3 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 7)                 903

=================================================================
Total params: 1,922,759
Trainable params: 1,922,759
Non-trainable params: 0
_________________________________________________________________
```

Fig 17 Architecture of our proposed CNN Model

9.2 MODEL ACCURACY GRAPHS

plt.plot(hist1.history['accuracy'])

plt.plot(hist1.history['val_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

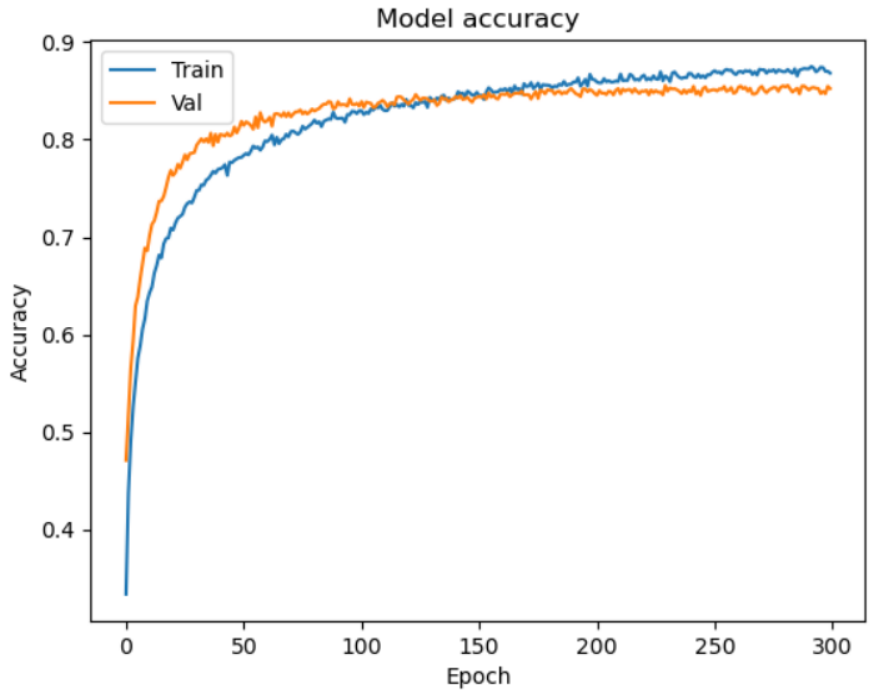


Fig 18 Model Accuracy graphs

9.3 MODEL LOSS GRAPHS

```
plt.plot(hist1.history['loss'])
plt.plot(hist1.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

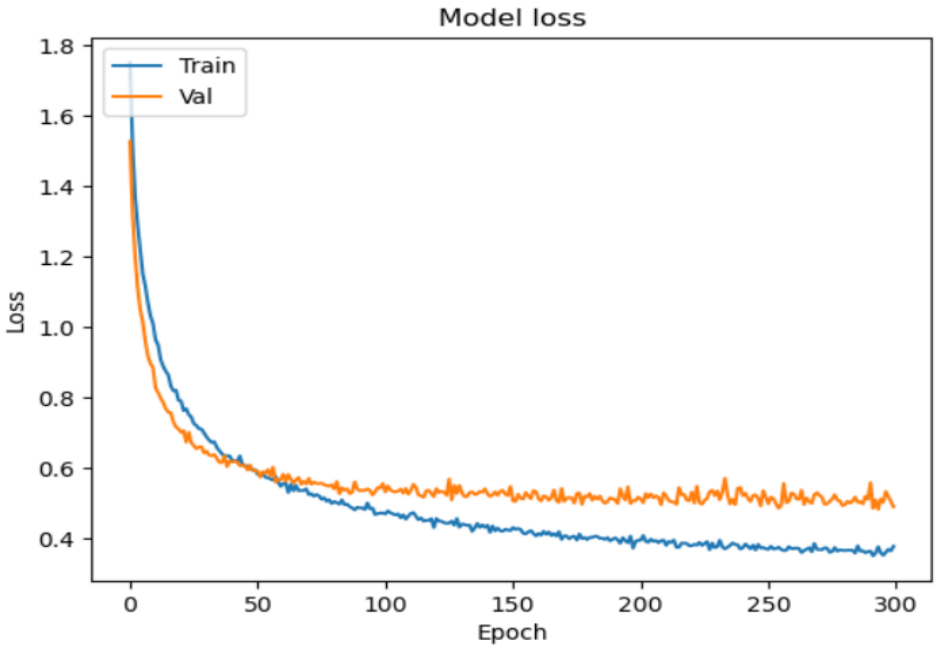


Fig 19 Model Loss graphs

9.4 CONFUSION MATRIX

```
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(ytrue, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
display_labels = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'])
cm_display.plot()
plt.show()
```

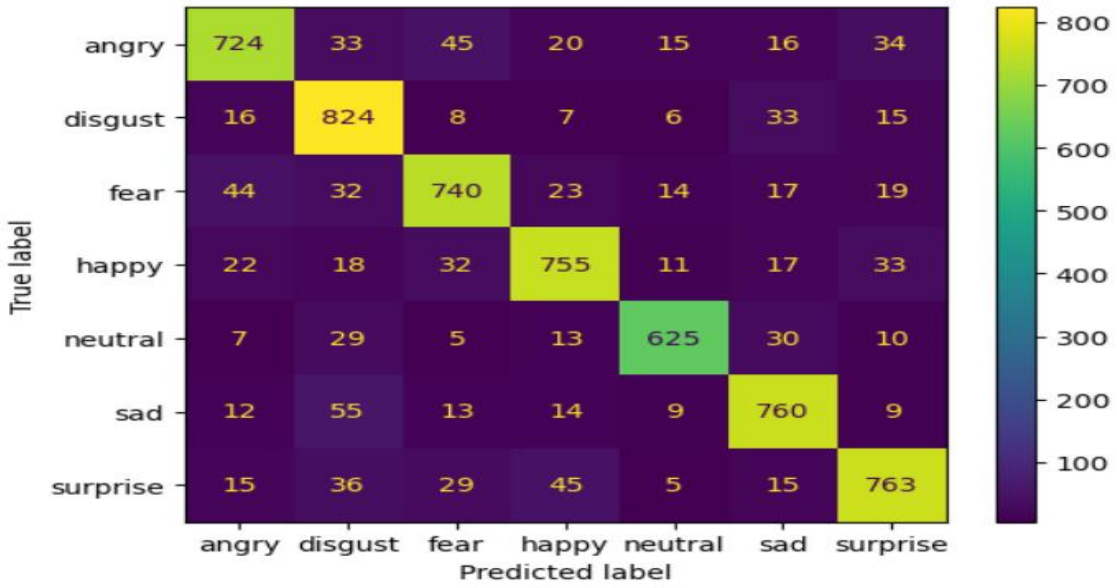


Fig 20 Confusion Matrix

9.5 PREDICTING EMOTION

```
def find_emotion(path):
    data,sr = remove_silence(path)
    mfcc = extract(data,sr)
    m=mfcc.reshape(-1,len(mfcc))
    l=pd.DataFrame(m)
    l[np.arange(l.shape[1],687)]=np.nan
    l=l.fillna(0)
    lt = l.values.reshape(len(l),229,3,1)
    y_check = new_model.predict(lt)
    y_pred = (y_check>0.5)
    res = y_pred.flatten()
    rr =res.argmax(axis=-1)
    return lb.inverse_transform([rr])
```

```
print(*find_emotion(path))
```

```
1/1 [==============================] - 0s 25ms/step
sad
```

10. CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION

In this project, a SER system was developed utilizing a CNN to classify a emotional state of an individual based on their audio data. The CNN model was specifically designed to automatically extract features from original audio data and learn to predict emotional categories through supervised learning. Our experimental results indicated that the proposed SER system was highly effective, achieving an accuracy rate of 85% on the testing dataset. The CNN outperformed traditional machine learning methods in the field, demonstrating its efficacy in SER tasks. However, the model's performance may have been further improved with more data.

10.2 FUTURE SCOPE

Speech emotion recognition systems can include the integration of other forms of data, such as facial expressions and physiological signals, so that it can enhance the performance of the system. Other areas of potential improvement include incorporating contextual information, such as the speaker's background and environment, and using more advanced deep learning techniques, such as recurrent neural networks and attention mechanisms.

11.REFERENCES

1. Speech Emotion Recognition Systems: A Comprehensive Review
   A Comprehensive Review of Speech Emotion Recognition Systems | IEEE Journals & Magazine | IEEE Xplore

2. The paper titled "Speech Emotion Recognition Using CNN" is authored by Apoorv Singh, Kshitij Kumar Srivastava, and Harini Murugan.
   DOI:10.37200/IJPR/V24I8/PR280260,
   https://www.researchgate.net/publication/342231090_Speech_Emotion_Recognition_Using_CNN

3. In September 2020, Darshan K.A from U.B.D.T. College of Engineering, Davanagere, Karnataka, India, and Dr. B.N. Veerappa from the Department of Studies in Computer Science and Engineering, U.B.D.T. College of Engineering, Davanagere, Karnataka, India, published an article on speech emotion recognition.
   https://www.irjet.net/archives/V7/i9/IRJET-V7I9154.pdf

4. In 2021, Babak Joze Abbaschian, Daniel Sierra-Sosa, and Adel Elmaghraby published a paper titled "Deep Learning Techniques for Speech Emotion Recognition," which explores the use of deep learning models for recognizing emotions in speech, transitioning from databases to models.
   Sensors | Free Full-Text | Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models (mdpi.com)

5. Ravdess data
   https://www.kaggle.com/datasets/deepakk6/ravdess-data

6. Toronto emotional speech set (TESS) data
   https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess

7. Jianfeng Zhaoa,b, Xia Maoa, and Lijiang Chen's work focuses on using deep 1D and 2D CNN LSTM networks for speech emotion recognition.
   Sci-Hub | Speech emotion recognition using deep 1D & 2D CNN LSTM networks. Biomedical Signal Processing and Control, 47, 312–323 | 10.1016/j.bspc.2018.08.035