TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

There are many ways of communication but the speech signal is one of the fastest and most natural methods of communications between humans. Therefore, the speech can be the fast and efficient method of interaction between human and the machine as well. Humans have the natural ability to use all their available senses for maximum awareness of the received message. Through all the available senses people actually sense the emotional state of their communication partner. The emotional detection is natural for humans but it is very difficult task for machine. Therefore, the purpose of emotion recognition system is to use emotion related knowledge in such a way that human machine communication will be improved.

In this system, the quality of feature extraction directly affected the accuracy of speech emotion recognition. In the process of feature extraction, it usually took the whole emotion sentence as units for feature extracting, and extraction contents were four aspects of emotion speech, which were several acoustic characteristics of time construction, amplitude construction, fundamental frequency construction, and formant construction. Then contrast emotion speech with no emotion sentence from these four aspects, acquiring the law of emotional signal distribution, then classify emotion speech according to the law.

Deep neural network (DNN) has unprecedented success in the field of speech recognition and image recognition; however, so far no research on deep neural network has been applied to speech emotion processing. We found that the DNN in speech emotion processing has a huge advantage. Therefore, this paper proposed a method to realize the emotional features automatically extracted from the audio using the librosa package in python. We used DNN to train a 5-layer-deep network to extract speech emotion features. It incorporates the speech emotion features of more consecutive frames, to build a high latitude characteristic, and uses softmax classifier layer to classify the emotional speech. The speech emotion recognition test accuracy reached 73.38% which is a high value compared to the other models of this size.

Traditional machine learning methods are k-nearest neighbors (KNN), Hidden Markov Model (HMM) and Support Vector Machine (SVM), Artificial Neural Network (ANN), Gaussian Mixtures Model (GMM) etc to classify the emotions.

The important issues in speech emotion recognition system are the signal processing unit in which appropriate features are extracted from available speech signal and another is a classifier which recognizes emotions from the speech signal. The average accuracy of the most of the classifiers for speaker independent system is less than that for the speaker dependent.

Automatic emotion recognitions from the human speech are increasing now a day because it results in the better interactions between human and machine.

1.1 Problem Definition, significance, and objective

Problem Definition:

The problem statement of emotional recognition using speech processing is to develop algorithms and techniques that can accurately identify and classify the emotions expressed in human speech. This involves analyzing various acoustic features of speech, such as pitch, volume, and tempo, and using machine learning techniques to identify patterns that correspond to different emotional states, such as happiness, sadness, anger, or fear. The ultimate goal is to develop robust and reliable models that can automatically recognize emotions in real-time applications, such as customer service interactions, virtual assistants, or mental health assessments. This field has important implications for improving human-computer interaction, mental health diagnosis, and communication research.

Significance:

This project report aims to provide comprehensive information about the technical aspects and real-time of the project – Emotion Recognition Using Speech Processing using Deep learning technique.

Objective:

Human speech is the most natural way to express ourselves. We use it everywhere from calls, emails, meetings, discussions etc. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication. This project can be defined as a collection of methodologies that process and classify speech signals to detect emotions in them. The objective is to detect the emotions of a person or speaker and to implement a Deep Neural Network (DNN) model to create the application.

1.2 Methodologies

We propose a modified speech emotion recognition method which uses deep neural networks for training. The method uses Mel-frequency cepstral coefficients (MFCC), Chromogram, Mel scaled spectrogram in conjunction with Spectral contrast and Tonal Centroid features to extract details about an audio file. The features are used to train DNN model in a 5 layer deep neural network. The dataset used here is the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). We have only chosen the speech part which consists of 24 actors (gender balanced) with 1440 audio files. The model classifies the speech audio in 8 different emotions namely neutral, calm, happy, sad, angry, fearful, disgust, surprised.

1.3 Outline of the project

At the end of the project, our code will find these things.

Detect Human Emotion

People Attack to this site

New techniques learn From Speech

Easy to find emotions

1.4 Scope of the Project

Automatic emotion recognition using speech can help organizations to understand their customers better when in a call.

Call centers can make separate strategies on dealing with people with different people.

For E-Learning, schools can monitor the emotions of their students to better prepare their education system for the betterment of the students.

Robotics has wide use of Emotion detection as a robot designed to interact with human should understand the human’s emotion.

Emotion detection is the key to Human Computer Interaction (HCI).

1.5 Organization Of the Report

This report will clearly explain the features, algorithm used, proposed implementations, results of the implementations, and conclusions drawn from the results. After this introduction section, section 2, the literature survey, will introduce the problem in detail. It tells accuracy, techniques, and discuss how they will be used. We will then go over the existing solutions that have already been presented along withany other related works.

Section 3, We will investigate the details of how exactly we implement and how this technique is run and implemented differently from the existing solutions. implementation architectures, module descriptions.

Next, section 4 will show the implementation of the proposed system. This section will depict the implementation of the execution through architecture. The algorithms will also be presented in this section. data sets would also be presented and described in this section.

section 5, We will look at how the code run at each step and displaying all the outputs and results obtained from the training and execution of our proposed model.

Finally in section 6, we will conclude, based on our implementation and results, whether the execution successful. Finally, after observing all the results and drawing conclusions, we present our recommendations on how to use this code, who should use this algorithm, what scenarios is it most effective in, how we can improve it, and so on. The future works and scopes of this project will also be discussed in this section.

2.LITERATURE SURVEY

[1] Szegedy, Christian & Toshev, Alexander & Erhan, Dumitru. (2013). Deep Neural Networks for Object Detection. 1-9.

Deep Neural Networks (DNNs) have recently shown outstanding performance on image classification tasks [14]. In this paper we go one step further and address the problem of object detection using DNNs, that is not only classifying but also precisely localizing objects of various classes. We present a simple and yet powerful formulation of object detection as a regression problem to object bounding box masks. We define a multi-scale inference procedure which is able to produce high-resolution object detections at a low cost by a few network applications. State-of-the-art performance of the approach is shown on Pascal VOC.

Summary: This journal discusses about the Deep Neural Networks theory and object detection using DNN.

[2] Benk, Sal & Elmir, Youssef & Dennai, Abdeslem. (2019). A Study on Automatic Speech Recognition. 10. 77-85. 10.6025/jitr/2019/10/3/77-85.

Speech is an easy and usable technique of communication between humans, but nowadays humans are not limited to connecting to each other but even to the different machines in our lives. The most important is the computer. So, this communication technique can be used between computers and humans. This interaction is done through interfaces, this area called Human Computer Interaction (HCI). This paper gives an overview of the main definitions of Automatic Speech Recognition (ASR) which is an important domain of artificial intelligence and which should be taken into account during any related research (Type of speech, vocabulary size... etc.). It also gives a summary of important research relevant to speech processing in the few last years, with a general idea of our proposal that could be considered as a contribution in this area of research and by giving a conclusion referring to certain enhancements that could be in the future works.

Summary: This article helps us in understanding and using the speech recognition by machines which improves Human Computer Interactions and is also useful in our project.

[3] Ashish B. Ingale & D. S. Chaudhari (2012). Speech Emotion Recognition. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2 Issue-1, March 2012

In human machine interface application, emotion recognition from the speech signal has been research topic since many years. To identify the emotions from the speech signal, many systems have been developed. In this paper speech emotion recognition based on the previous technologies which uses different classifiers for the emotion recognition is reviewed. The classifiers are used to differentiate emotions such as anger, happiness, sadness, surprise, neutral state, etc. The database for the speech emotion recognition system is the emotional speech samples and the features extracted from these speech samples are the energy, pitch, linear prediction cepstrum coefficient (LPCC), Mel frequency cepstrum coefficient (MFCC). The classification performance is based on extracted features. Inference about the performance and limitation of speech emotion recognition system based on the different classifiers are also discussed.

Summary: In this paper, we learn the importance and the need of a different features in any audio or speech including mfcc, mel and other features which are used in our application for the purpose of predicting the emotions based on audio.

[4] Chenchen Huang, Wei Gong, Wenlong Fu, Dongyu Feng, "A Research of Speech Emotion Recognition Based on Deep Belief Network and SVM", ***Mathematical Problems in Engineering*****, vol. 2014, Article**
ID 749604, 7 pages, 2014. https://doi.org/10.1155/2014/749604

Feature extraction is a very important part in speech emotion recognition, and in allusion to feature extraction in speech emotion recognition problems, this paper proposed a new method of feature extraction, using DBNs in DNN to extract emotional features in speech signal automatically. By training a 5 layers depth DBNs, to extract speech emotion feature and incorporate multiple consecutive frames to form a high dimensional feature. The features after training in DBNs were the input of nonlinear SVM classifier, and finally speech emotion recognition multiple classifier system was achieved. The speech emotion recognition rate of the system reached 86.5%, which was 7% higher than the original method.

Summary: In this paper, we learned the importance of DNN model and its implementation which we are going to use in our application as well.

2.1Introduction to the problem

There are many ways of communication but the speech signal is one of the fastest and most natural methods of communications between humans. Therefore, the speech can be the fast and efficient method of interaction between human and the machine as well. Humans have the natural ability to use all their available senses for maximum awareness of the received message. Through all the available senses people actually sense the emotional state of their communication partner. The emotional detection is natural for humans but it is very difficult task for machine. Therefore, the purpose of emotion recognition system is to use emotion related knowledge in such a way that human machine communication will be improved.

In this system, the quality of feature extraction directly affected the accuracy of speech emotion recognition. In the process of feature extraction, it usually took the whole emotion sentence as units for feature extracting, and extraction contents were four aspects of emotion speech, which were several acoustic characteristics of time construction, amplitude construction, fundamental frequency construction, and formant construction. Then contrast emotion speech with no emotion sentence from these four aspects, acquiring the law of emotional signal distribution, then classify emotion speech according to the law.

Deep neural network (DNN) has unprecedented success in the field of speech recognition and image recognition; however, so far no research on deep neural network has been applied to speech emotion processing. We found that the DNN in speech emotion processing has a huge advantage. Therefore, this paper proposed a method to realize the emotional features automatically extracted from the audio using the librosa package in python. We used DNN to train a 5-layer-deep network to extract speech emotion features. It incorporates the speech emotion features of more consecutive frames, to build a high latitude characteristic, and uses softmax classifier layer to classify the emotional speech. The speech emotion recognition test accuracy reached 73.38% which is a high value compared to the other models of this size.

We propose a modified speech emotion recognition method which uses deep neural networks for training. The method uses Mel-frequency cepstral coefficients (MFCC), Chromogram, Mel scaled spectrogram in conjunction with Spectral contrast and Tonal Centroid features to extract details about an audio file. The features are used to train DNN model in a 5 layer deep neural network. The dataset used here is the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). We have only chosen the speech part which consists of 24 actors (gender balanced) with 1440 audio files. The model classifies the speech audio in 8 different emotions namely neutral, calm, happy, sad, angry, fearful, disgust, surprised.

2.2 EXISTING METHOD

In most of the currently available systems, the model used for emotion recognition uses traditional Machine learning algorithms like Support Vector Machines (SVM), K-Nearest neighbors (KNN) etc. The accuracies of these models are low. However, there are other Deep learning models as well but they are generally trained using large datasets which takes a lot of time and hence are very complex models.

DISADVANTAGES:

Lower accuracy.

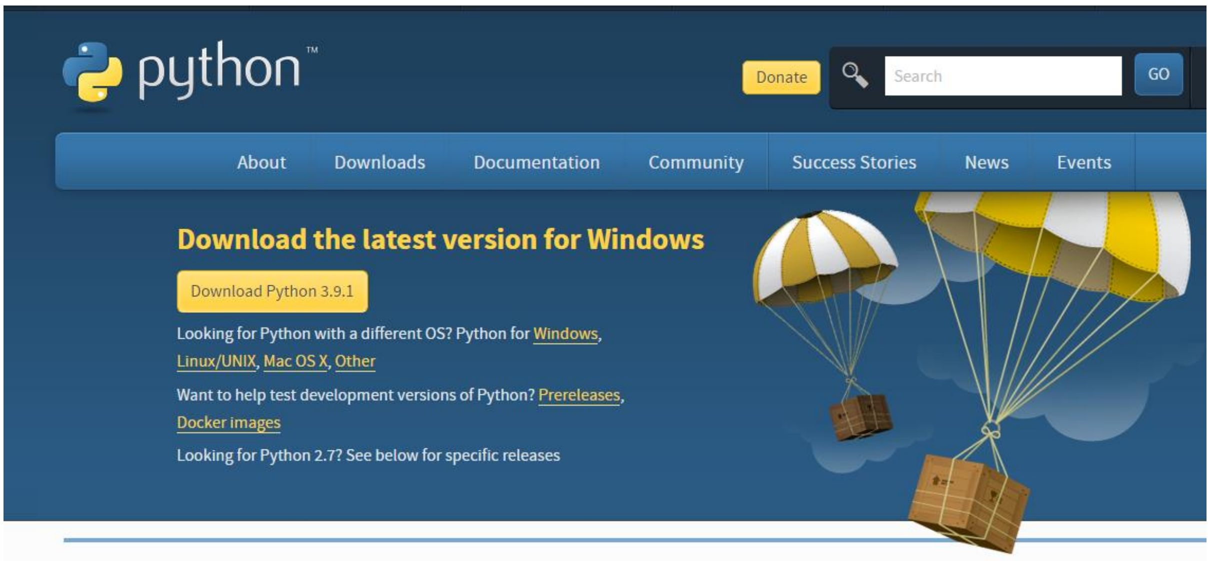
High computational complexity.

Requires high performance hardware to use the application.

2.3 Related works

2.3.1 Installing Python

1. To download and install Python visit the official website of Python https://www.python.org/downloads/ and choose your version.



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

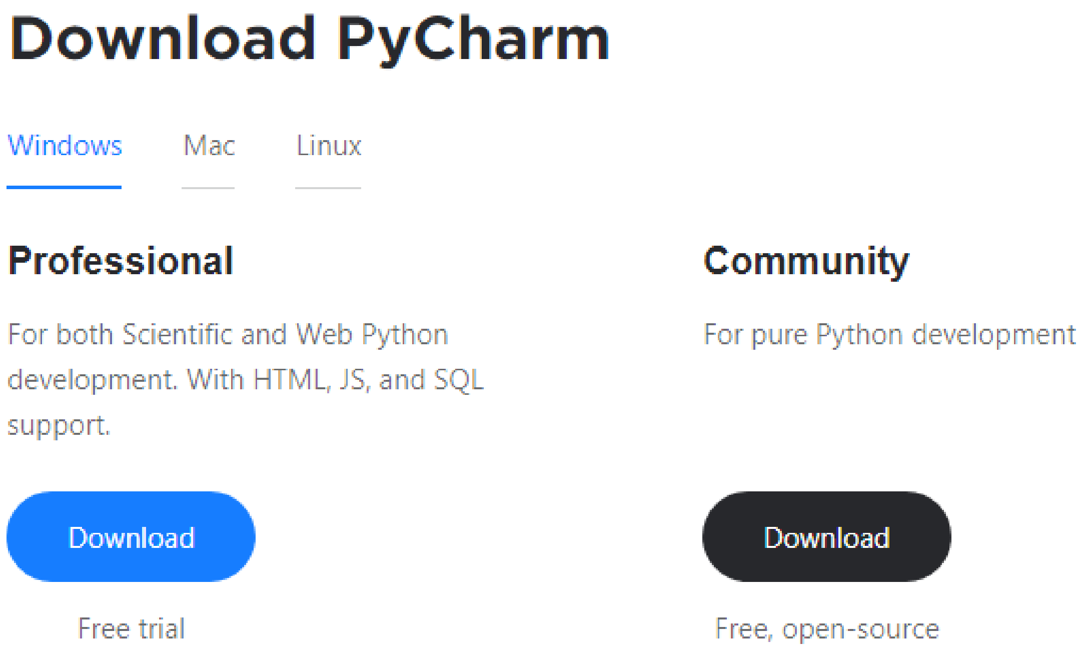
2. Once the download is complete, run the exe for install Python. Now click on Install Now.

3. You can see Python installing at this point.

4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

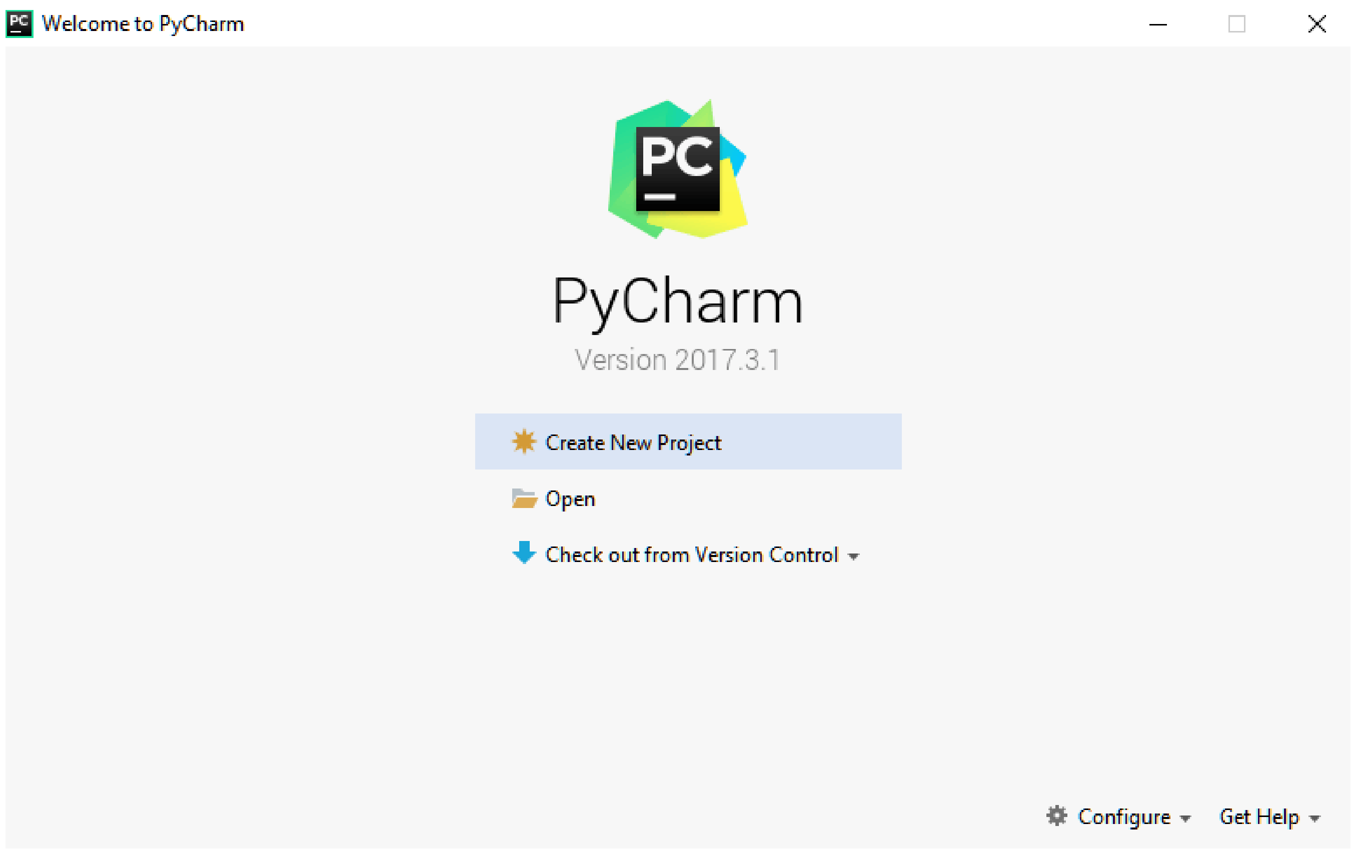
2.3.2 Installing PyCharm:

1. TodownloadPyCharmvisitthewebsite https://www.jetbrains.com/pycharm/download/ and Click the "DOWNLOAD" link under the Community Section.


Download PyCharm
Windows
Mac
Linux
Professional
For both Scientific and Web Python development. With HTML, JS, and SQL support.
Download
Free trial
Community
For pure Python development
Download
Free, open-source

2. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click "Next".
3. On the next screen, Change the installation path if required. Click "Next".
4. On the next screen, you can create a desktop shortcut if you want and click on "Next".
5. Choose the start menu folder. Keep selected JetBrains and click on "Install".
6. Wait for the installation to finish.
7. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the "Run PyCharm Community Edition" box first and click "Finish".

8. After you click on "Finish," the Following screen will appear.

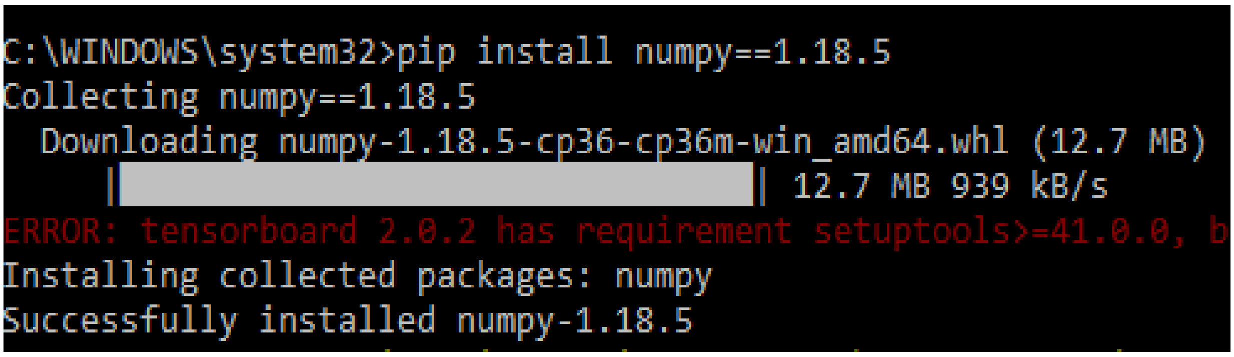


9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib.pyplot)

Ex: pip install numpy



```
C:\WINDOWS\system32>pip install numpy==1.18.5
Collecting numpy==1.18.5
  Downloading numpy-1.18.5-cp36-cp36m-win_amd64.whl (12.7 MB)
     |████████████████████████████████| 12.7 MB 939 kB/s
ERROR: tensorboard 2.0.2 has requirement setuptools>=41.0.0, b
Installing collected packages: numpy
Successfully installed numpy-1.18.5
```

2.4 MODULES

1) Upload:

Upload the dataset of audio (.wav files) to be read using librosa library.

2) View:

Uploaded dataset can be viewed.

3) Pre-processing:

Data Pre-processing is a technique that is used to convert the raw data into a clean data set. Cleaning the data refers to removing the null values, filling the null values with meaningful value, removing duplicate values, removing outliers, removing unwanted attributes. If dataset contains any categorical records means convert those categorical variables to numerical values.

4) Identifying Features:

The extracted features are Mel-frequency cepstral coefficients (MFCC), Chromogram, Mel scaled spectrogram in conjunction with Spectral contrast and Tonal Centroid features.

5) Train and Test Split:

We split our dataset of 1440 audio files in 2 parts, training data with 1008 audio files and testing data with 432 audio files. Here 70% of the data is taken for the training dataset.

6) Building the model:

To understand the audio and predict emotions, we are proposing a Deep learning-based method Deep learning can provide increased accuracy and decrease in computational power.We will use Deep Neural Networks (DNN) to create the model. Deep Neural Network (DNN) is widely used in deep learning to train models for tasks which traditional machine learning algorithms cannot do or is hard to doThe model is created using 5 layers of neural networks.We have used dropouts to minimize the problem of overfitting. In order to classify the audio to the emotions, we are using softmax in the outermost layer of our DNN model. Softmax takes in a vector of numbers and converts them to probabilities which are then used for image generating results.

Softmax converts logits into probabilities by taking the exponents from every output and then normalize each of these numbers by the sum of such exponents, such that the entire output vector adds up to one.

8) Prediction:

An audio is uploaded by the user (which includes speech of a person), and the model is used to predict the emotion of the speaker in the audio.

9) User Interface:

A Flask architecture based web application is developed to use the model. It has 2 parts, the system and the user. There is a user registration and login management system in the UI.

10) Registration:

A new user first needs to register their details which includes name, email and the password. This user information is stored in MySQL database.

11) User Login:

An already registered user, whose data is stored in the system's MySQL database can login to the web app using their valid credentials. Once they successfully log in, only then they are provided access to the application to predict the emotions.

2.5 Technologies used

Deep Learning – python library. Used for implementation algorithm. Flask , sql and xmapp control server and pycharm are used.

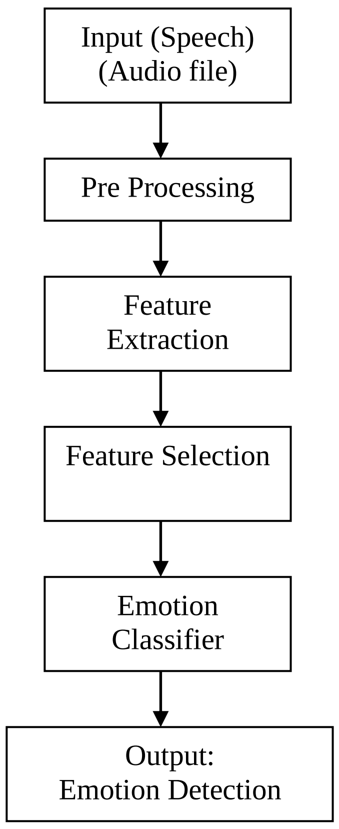
3 PROBLEM STATEMENT

3.1 Objectives

Human speech is the most natural way to express ourselves. We use it everywhere from calls, emails, meetings, discussions etc. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication. This project can be defined as a collection of methodologies that process and classify speech signals to detect emotions in them. The objective is to detect the emotions of a person or speaker and to implement a Deep Neural Network (DNN) model to create the application.

4 Block Diagram

The flow for the project is given below:



4.1 Theoretical Foundation/Algorithm

1) Artificial Neural Network:

An artificial neural network is a system of hardware or software that is patterned after the working of neurons in the human brain and nervous system. Artificial neural networks are a variety of deep learning technology which comes under the broad domain of Artificial Intelligence.

Deep learning is a branch of Machine Learning which uses different types of neural networks. These algorithms are inspired by the way our brain functions and therefore many experts believe they are our best shot to moving towards real AI (Artificial Intelligence).

Some types of Neural Networks:

2) Feed forward Neural Network – Artificial Neuron

This is one of the simplest types of artificial neural networks. In a feed forward neural network, the data passes through the different input nodes until it reaches the output node.

In other words, data moves in only one direction from the first tier onwards until it reaches the output node. This is also known as a front propagated wave which is usually achieved by using a classifying activation function.

Unlike in more complex types of neural networks, there is no back propagation and data moves in one direction only. A feed forward neural network may have a single layer or it may have hidden layers.

In a feed forward neural network, the sum of the products of the inputs and their weights are calculated. This is then fed to the output. Here is an example of a single layer feedforward neural network.

3) Radial Basis Function Neural Network

A radial basis function considers the distance of any point relative to the centre. Such neural networks have two layers. In the inner layer, the features are combined with the radial basis function.

Then the output of these features is taken into account when calculating the same output in the next time-step.

The radial basis function neural network is applied extensively in power restoration systems. In recent decades, power systems have become bigger and more complex. This increases the risk of a blackout. This neural network is used in the power restoration systems in order to restore power in the shortest possible time.

4) Multilayer Perceptron (MLP)

A multilayer perceptron has three or more layers. It is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer.

A multilayer perceptron uses a nonlinear activation function (mainly hyperbolic tangent or logistic function).

This type of neural network is applied extensively in speech recognition and machine translation technologies.

5) Convolutional Neural Network

A convolutional neural network (CNN) uses a variation of the multilayer perceptrons. A CNN contains one or more than one convolutional layer. These layers can either be completely interconnected or pooled.

Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters.

Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems.

Convolutional neural networks also show great results in semantic parsing and paraphrase detection. They are also applied in signal processing and image classification.

6) Recurrent Neural Network (RNN) – Long Short-Term Memory

A Recurrent Neural Network is a type of artificial neural network in which the output of a particular layer is saved and fed back to the input. This helps predict the outcome of the layer.

The first layer is formed in the same way as it is in the feed forward network. That is, with the product of the sum of the weights and features. However, in subsequent layers, the recurrent neural network process begins.

From each time-step to the next, each node will remember some information that it had in the previous time-step. In other words, each node acts as a memory cell while computing and carrying out operations. The neural network begins with the front propagation as usual but remembers the information it may need to use later.

If the prediction is wrong, the system self-learns and works towards making the right prediction during the back propagation. This type of neural network is very effective in text-to-speech conversion technology.

7) Modular Neural Network

A modular neural network has a number of different networks that function independently and perform sub-tasks. The different networks do not really interact with or signal each other during the computation process. They work independently towards achieving the output.

As a result, a large and complex computational process can be done significantly faster by breaking it down into independent components. The computation speed increases because the networks are not interacting with or even connected to each other.

8) Sequence-To-Sequence Models

A sequence-to-sequence model consists of two recurrent neural networks. There’s an encoder that processes the input and a decoder that processes the output. The encoder and decoder can either use the same or different parameters. This model is particularly applicable in those cases where the length of the input data is not the same as the length of the output data.

Sequence-to-sequence models are applied mainly in chatbots, machine translation, and question answering systems.

Let us consider the basic part of any Neural Network:

1. Input Layer: It is the layer where we provide the input for the model. The number of features our input has is equal to the number of neuron in the input layer.

2. Hidden Layer: The input features are transferred to the hidden layer(s) where different processes/activities takes place. There can be multiple hidden layers. The layers undergoes mathematical operations like matrix multiplication, convolutions, pooling etc. along with an activation function.

3. Output Layer: They layer which is used to generate probability scores using sigmoid or softmax functions which is then converted to the output of our model.

9) Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1, 1), then stride to (1, 3), then to (1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one. With padding, the output from an input of width w, height h and depth 3 would be the ceiling of width w/2, height h/2 and depth 1, as the kernel outputs a single summed output from each stride.

For example, with an input of 3x64x64 (say a 64x64 RGB three channel image), one kernel taking strides of two with padding the edge pixels, would produce a channel/feature map of 32x32.

The first step of creating and training a new convolutional neural network (Convnet) is to define the network architecture. This topic explains the details of Convent layers, and the order they appear in a ConvNet. For a complete list of deep learning layers and how to create them, see List of Deep Learning Layers. To learn about LSTM networks for sequence classification and regression, see Long Short-Term Memory Networks. To learn how to create your own custom layers, see Define Custom Deep Learning Layers. The network architecture can vary depending on the types and numbers of layers included.

4.2 Six Layers

1) Image Input Layer:

Create an image input layer using image input layer. An image input layer inputs images to a network and applies data normalization. Specify the image size using the input Size argument. The size of an image corresponds to the height, width, and the number of color channels of that image. For example, for a grayscale image, the number of channels is 1, and for a color image it is 3.

2) Convolution Layer:

Convolutional layers are the major building blocks used in convolutional neural networks.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input result in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modelling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

3) Pooling Layer:

It is common to periodically insert a Pooling layer in-between successive Convolution layer in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

4) Input Sequence Layer:

A sequence input layer inputs sequence data to a network. In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models" with no further context. Here's how it works:

In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models". Here's how it works:

- A RNN layer (or stack thereof) acts as "encoder": it processes the input sequence and returns its own internal state. Note that we discard the outputs of the encoder RNN, only recovering the state.
- Another RNN layer (or stack thereof) acts as "decoder": it is trained to predict the next characters of the target sequence, given previous characters of the target sequence.

5) LSTM Layer:

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning.

During back propagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural networks weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time.

6) Fully Connected Layer:

Fully connected layers connect every neuron in one layer to every neuron in another layer. The flattened matrix goes through a fully connected layer to classify the images.

Convolutional layers act to detect features that help classification, by picking up edges and curves and then from there detecting shapes and from there picking out ears for example, but as they can only filter images a dense layer is required to look at the output of the final convolutional neurons/filters and output a number (or numbers if one hot encoding is being used) as the classification. The outputs of all the neurons/filters in the last convolutional layer are joined together and then are “flattened” into 1D data ie it all becomes one row of data instead of the rows and columns of the data. The 1D data then acts as the input to the neuron(s) of the fully connected layer which performs a dot product of this input data and the neuron’s weights to produce a single number as output (a single number per neuron).

7) Output Layers:

1) Softmax and Classification Layers:

Softmax converts logits into probabilities by taking the exponents from every output and then norms each of these numbers by the sum of such exponents, such that the entire output vector adds up to one – every probability should be one. Generally, cross-entropy loss is the loss of such a problem in several classes. In the last layer of an image classification network such as CNN (e.g. VGG16) used in ImageNet competitions, softmax is also applied.

A softmax layer applies a softmax function to the input. A classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes. Create a classification layer using classification Layer. For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer. The softmax function is also known as the normalized exponential and can be considered the multi-class generalization of the logistic sigmoid function. For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, train Network takes the values from the Softmax function.

STEPS FOR EXECUTING THE PROJECTS

1. Import all the Libraries/packages.
2. Load the RAVDESS speech dataset.
3. Extract the features from the audio files.
4. Store the feature vector in local machine.
5. The labels to the features indicating the emotions must be separated and one hot encoded.
6. Split the dataset in train and test dataset with test size of 30%.
7. A Sequential() model with 5 layers is created.
8. Train dataset is used for training the dataset using 700 epochs.
9. The best one with respect to test accuracy is the model which we will use for prediction.
10. The model is loaded.
11. User uploads audio file location using the UI.
12. The model predicts the emotion of the file uploaded by the user.
13. The predicted emotion is displayed back to the user in the UI.

4.3 SYSTEM DESIGN

4.3.1 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

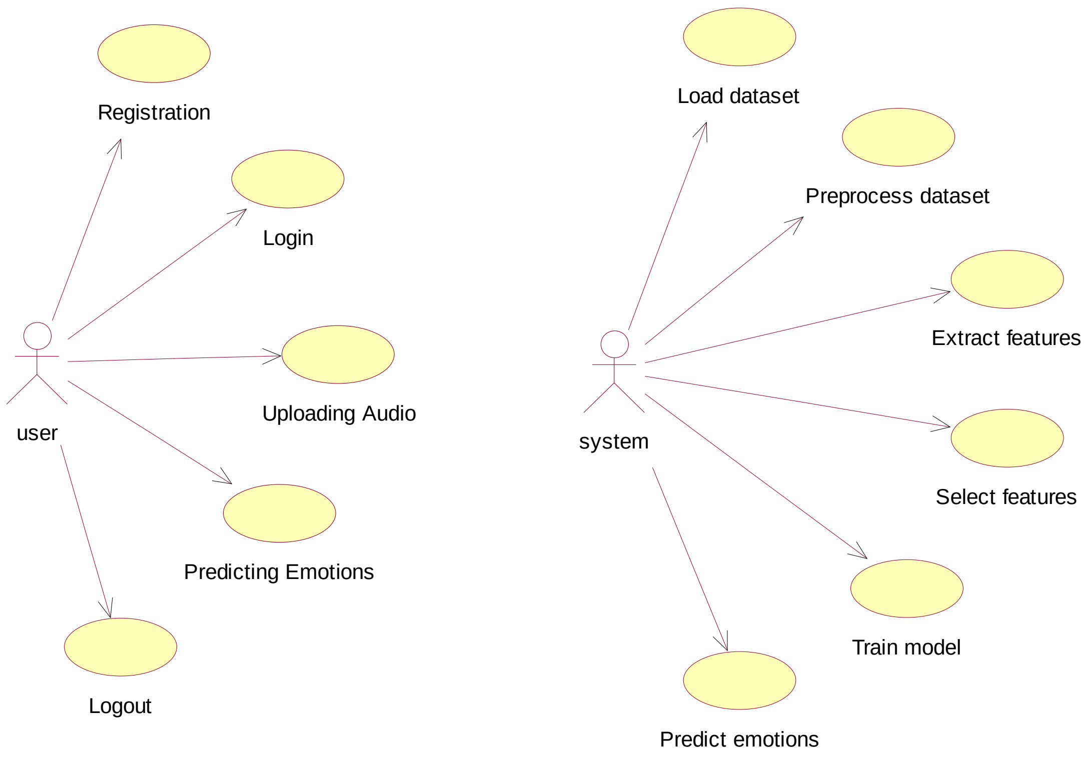
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

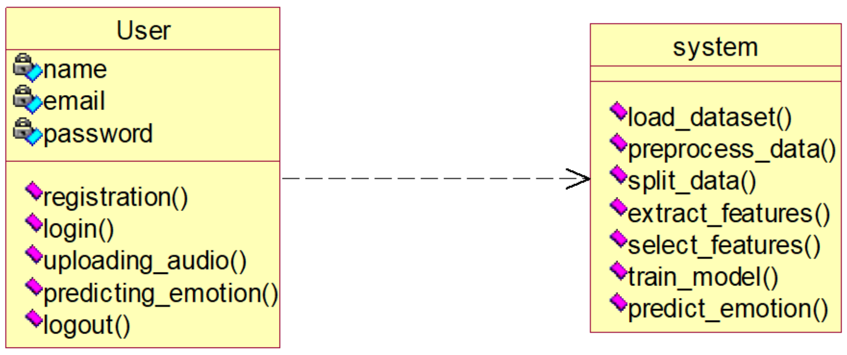
4.3.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



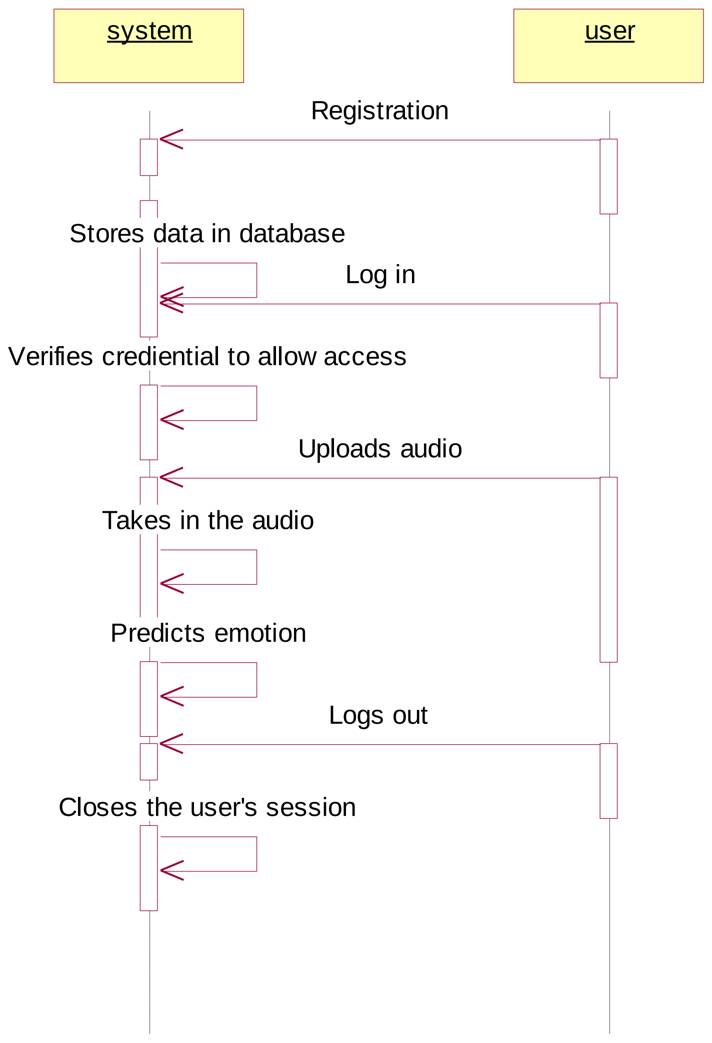
4.3.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



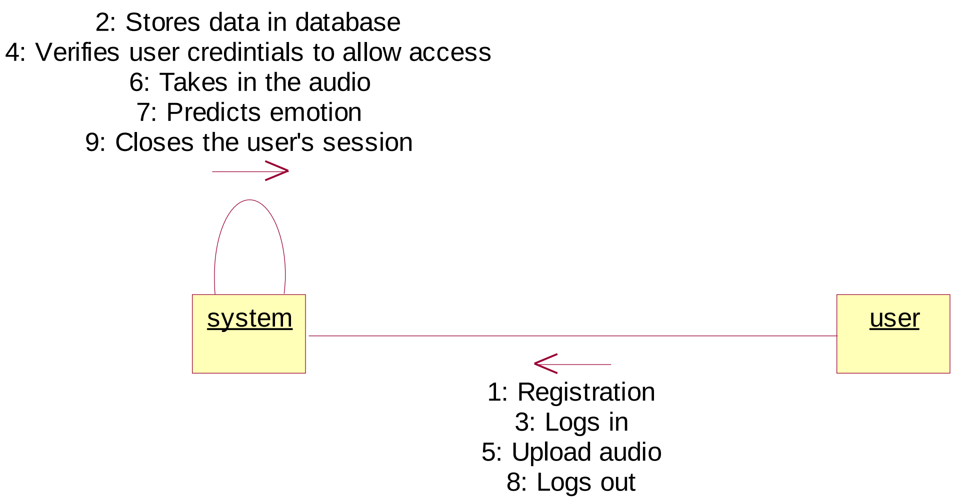
4.3.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



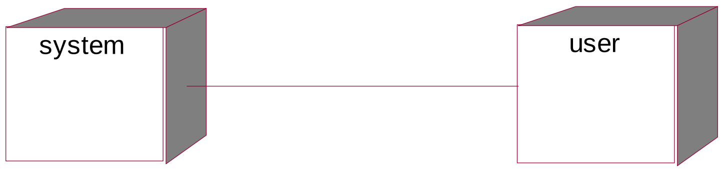
4.3.5 COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



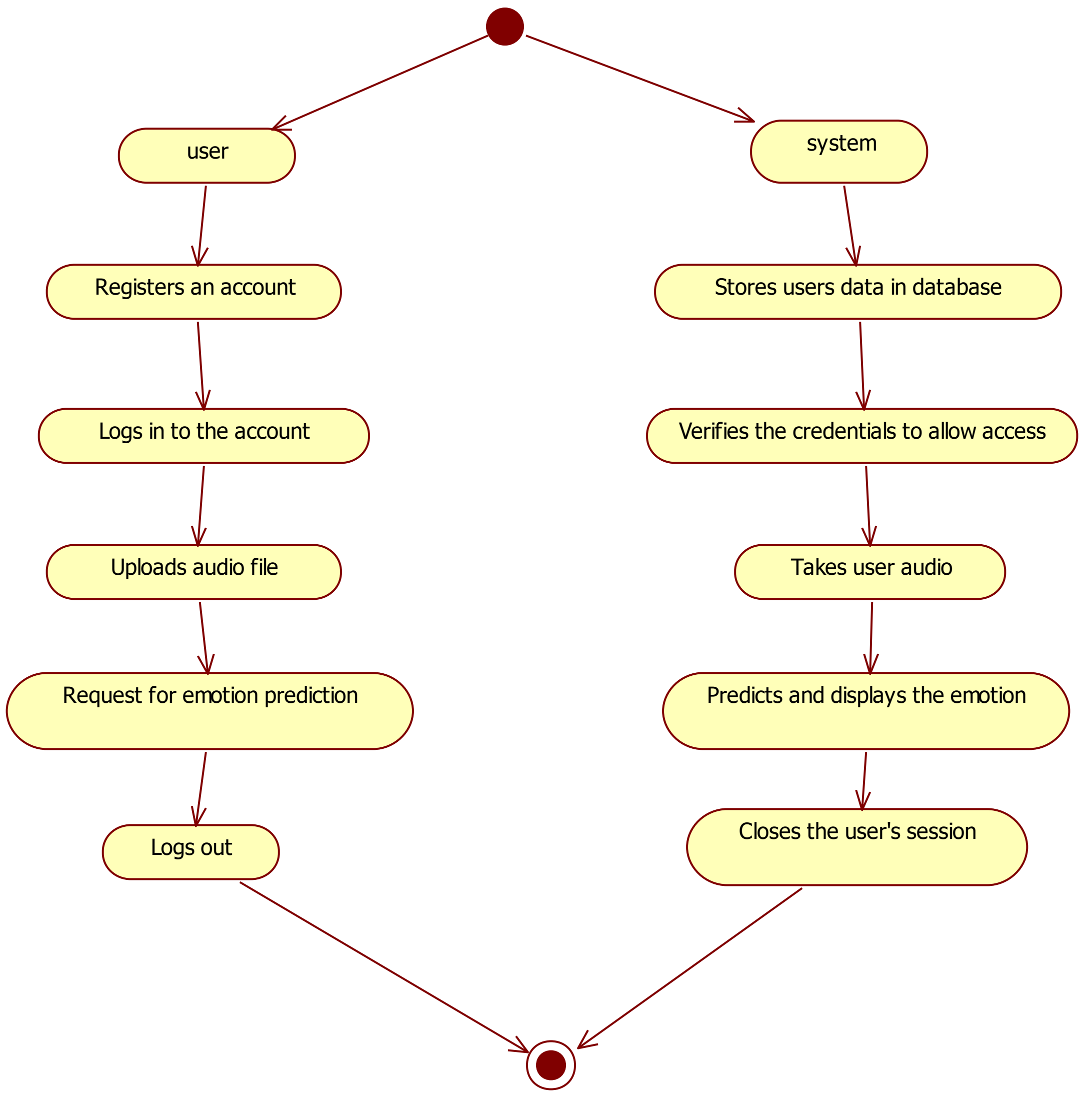
4.3.6 DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.



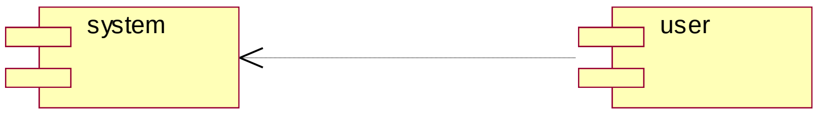
4.3.7 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.


user
system
Registers an account
Stores users data in database
Logs in to the account
Verifies the credentials to allow access
Uploads audio file
Takes user audio
Request for emotion prediction
Predicts and displays the emotion
Logs out
Closes the user's session

4.3.8 COMPONENT DIAGRAM

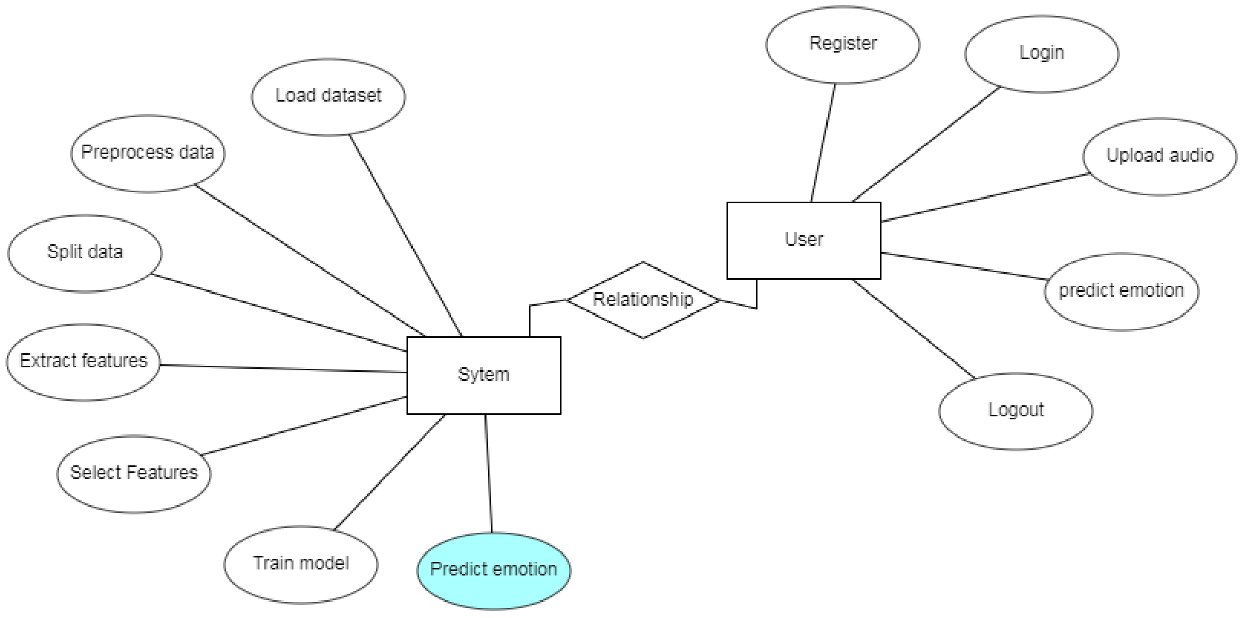
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



4.3.9 ER DIAGRAM:

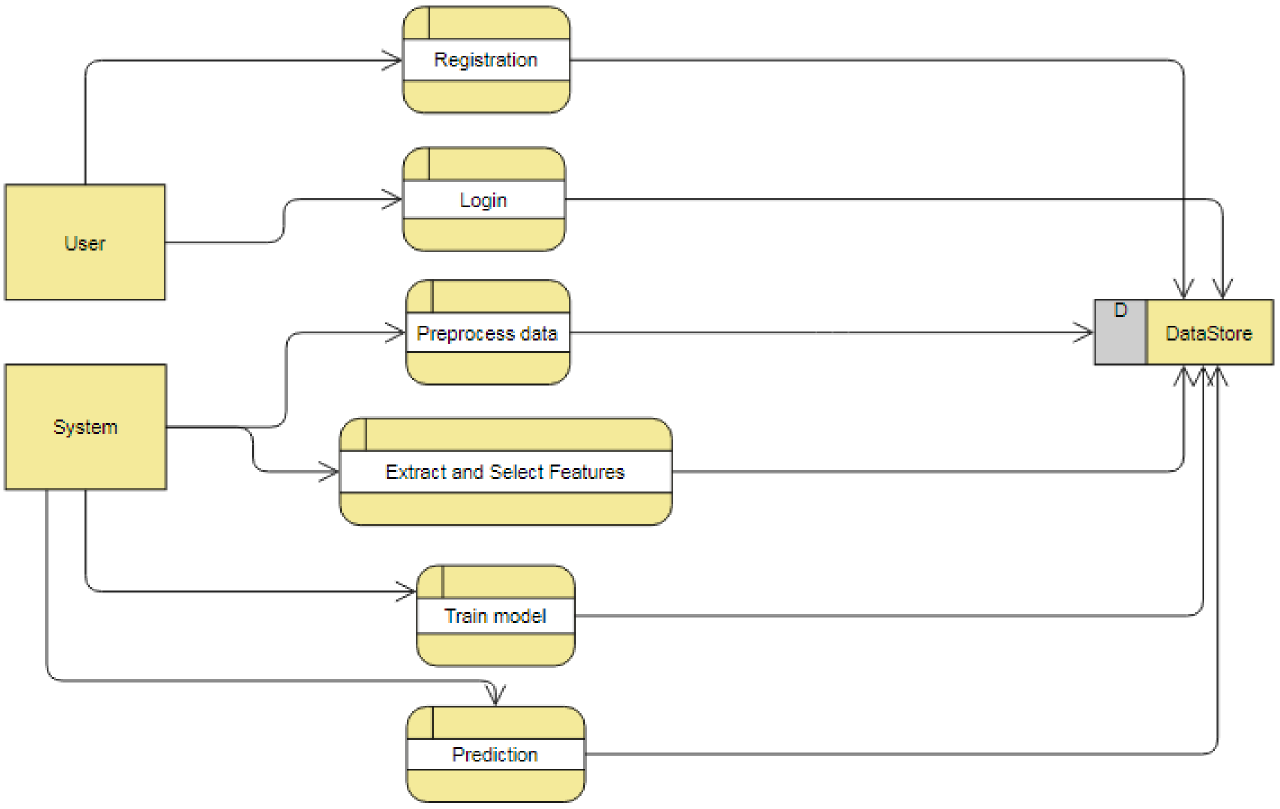
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

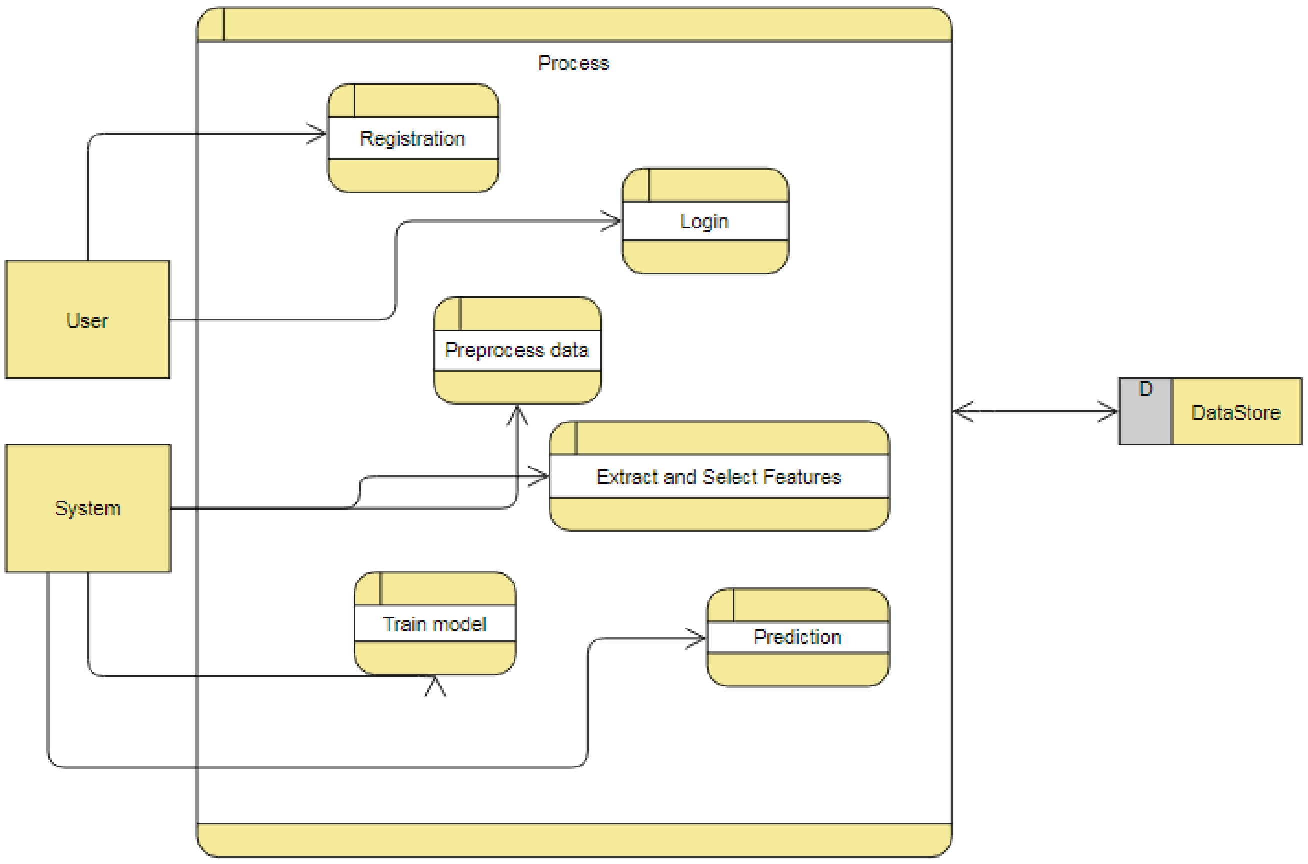
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



4.3.10 DFD DIAGRAM:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.


Registration
Login
User
Preprocess data
D
DataStore
System
Extract and Select Features
Train model
Prediction


Process
Registration
Login
User
Preprocess data
D
DataStore
System
Extract and Select Features
Train model
Prediction

5 SOFTWARE AND HARDWARE SPECIATIONS

5.1 Introduction

We must use operating systems in this using as windows or mac, Laptops or pc like electronic gadgets we must use to execute the purpose, Without this thing, we cannot move forward with data storage like this gadget's

5.2 Specific Requirements

Windows

5.3Hardware and Software Requirement

5.3.1 Hardware Requirement

Processor	- I3/Intel Processor
RAM	- 4GB (min)
Hard Disk	- 128 GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- Any

5.3.2 Software Requirement

Operating System	: Windows 7+
Server side Script	: Python 3.6+
IDE	: PyCharm
Libraries Used	: Pandas, Numpy, Keras, Tensorflow, Librosa, OpenCV, Flask, Pickle.
Dataset	: RAVDESS speech dataset.

6 IMPLEMENTATION

6.1 Code

6.1.1 MAIN CODE

```
from flask import Flask, render_template, request, url_for, redirect, flash,
send_from_directory, session
from forms import RegistrationForm, LoginForm
import pymysql
import pymysql.cursors
import pandas as pd
import os
from audio_wave import *

APP_ROOT=os.path.dirname(os.path.abspath(__file__))
app=Flask(__name__)
app.config['UPLOAD_FOLDER']=os.path.join(APP_ROOT, 'static/image/')
app.config['SECRET_KEY']='b0b4fbefdc48be27a6123605f02b6b86'

@app.before_first_request
def initialize():
    session['loggedin']=False

@app.route('/')
@app.route('/home')
def home():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/application')
def application():
    return render_template('application.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/library')
def library():
    return render_template('library.html')

@app.route('/login', methods=['GET','POST'])
def login():
    form = LoginForm()

    #Database connection
    db=pymysql.connect(host="localhost", port=3306, user="root", password="",
db="emotion_recognition", cursorclass=pymysql.cursors.DictCursor)
    #Table as a data frame
    register=pd.read_sql_query('select * from {}'.format('register'), db)

    if form.validate_on_submit():
        all_emails=register['email']
        if form.email.data in list(all_emails):
            row=all_emails[all_emails==form.email.data]
            index=row.index[0] ### Id is (1 + index)
            if form.password.data == register['password'][index]:
                session['loggedin']=True
                flash(f"Welcome {register['name'][index]} ! You have been logged 
in.",'success')
                return redirect(url_for('application'))
            else:
                flash("You password was incorrect. Please try again with correct 
password.",'warning')
                return redirect(url_for('login'))
        else:
            flash(f"No account with the email id {form.email.data} exists. Please 
register now.",'info')
            return redirect(url_for('register'))
    return render_template('login.html', form=form)

@app.route("/register", methods=['GET','POST'])
def register():
    form = RegistrationForm()
    #Database connection
    db=pymysql.connect(host="localhost", port=3306, user="root", password="", 
db="emotion_recognition", cursorclass=pymysql.cursors.DictCursor)
    #Table as a data frame
    register=pd.read_sql_query('select * from {}'.format('register'), db)

    if form.validate_on_submit():
        email=form.email.data
        all_emails=register['email']
        if email in list(all_emails):
            flash('Account already exists with this Email Id! Please Log 
In.','warning')
            db.close()
            return redirect(url_for('login'))
        else:
            #Insert new record
            try:
                with db.cursor() as cur:
                    sql = "INSERT INTO `register` (`name`,`email`,`password`) 
VALUES (%s, %s, %s)"
                    cur.execute(sql, ({form.username.data}, {form.email.data}, 
{form.password.data}))
                db.commit()
            finally:
                db.close()
            flash(f'Account Created for {form.username.data} 
Sucessfully!','success')

        return redirect(url_for('login'))
    return render_template('register.html', form=form)

@app.route("/upload", methods=["POST"])
def upload():
    target=os.path.join(APP_ROOT, 'static/audio/')

    if not os.path.isdir(target):
        os.mkdir(target)
    file=request.files["myaudio"]
    filename=file.filename
    if filename=="":
        flash('No File Selected','danger')
        return redirect(url_for('application'))

    destination="/".join([target, filename])
    #Extension check
    ext = os.path.splitext(destination)[1]
    if (ext==".wav"):
        pass
    else:
        flash("Invalid Extenstions! Please select a .wav audio file only.",
category="danger")
        return redirect(url_for('application'))

    if not os.path.isfile(destination):
        file.save(destination)

    result=record_audio(record=False, file_loc=destination)
    new_dest=('static/audio/'+str(filename))
    return render_template("upload.html", img_name=filename, emo=result,
destination=new_dest)


@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    #flash('Logged Out Sucessfully!','success')
    return redirect(url_for('home'))

if __name__ == '__main__':
    app.run(debug=True)
```

6.1.2 SPEECH EXTRACT

```
#Import libraries
import glob
import os
import librosa
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.layers import Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint


#Extract features
def extract_features(file_name):
    X, sample_rate = librosa.load(file_name)
    #Short time fourier transformation
    stft = np.abs(librosa.stft(X))
    #Mel Frequency Cepstra coeff (40 vectors)
    mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
    #Chromogram or power spectrum (12 vectors)
    chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
    #mel scaled spectogram (128 vectors)
    mel=np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T, axis=0)
```

```python
    # Spectral contrast (7 vectors)
    contrast=np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate).T,
axis=0)
    #tonal centroid features (6 vectors)

tonnetz=np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),sr=sample_ra
te).T, axis=0)
    return mfccs, chroma, mel, contrast, tonnetz

#parsing audio files
def parse_audio_files(parent_dir, sub_dirs, file_ext="*.wav"):
    features, labels = np.empty((0,193)), np.empty(0)
    for label, sub_dir in enumerate(sub_dirs):
        for fn in glob.glob(os.path.join(parent_dir, sub_dir, file_ext)):
            try:
                mfccs, chroma, mel, contrast, tonnetz= extract_features(fn)
            except Exception as e:
                print("Error encountered while parsing file: ", fn)
                continue
            ext_features = np.hstack([mfccs, chroma, mel, contrast, tonnetz])
            features = np.vstack([features, ext_features])
            labels = np.append(labels, fn.split("\\")[8].split("-")[2])
    return np.array(features), np.array(labels, dtype = np.int)

#fn=glob.glob(os.path.join(main_dir, sub_dir[0], "*.wav"))[0]

#One-Hot Encoding the multi class labels
def one_hot_encode(labels):
    n_labels = len(labels)
    n_unique_labels = len(np.unique(labels))
    one_hot_encode = np.zeros((n_labels, n_unique_labels + 1))
    one_hot_encode[np.arange(n_labels), labels] = 1
    one_hot_encode=np.delete(one_hot_encode, 0, axis=1)
    return one_hot_encode

#Extracting features in X
#Storing labels in y
main_dir = r'C:\Users\YMTS0297\PycharmProjects\Emotion Recognition using
speech\Datasets\Audio_Speech_Actors_01-24'
sub_dir = os.listdir(main_dir)
print("\nCollecting features and labels.")
print("\nThis will take some time.")
features, labels = parse_audio_files(main_dir, sub_dir)
print("\nCompleted")

#save features
np.save('X', features)
#one hot encode labels
labels = one_hot_encode(labels)
np.save('y', labels)

emotions=['neutral','calm','happy','sad','angry','fearful','disgused','surprised']
#labels2=np.array([emotions[labels[i]-1] for i in range(len(labels))])
#np.save('y2', labels2)

#Loading features and labels
#X=np.load('X.npy') #features
#y=np.load('y.npy') #labels
```

```
#Splitting the dataset
#train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=
0.3,random_state=42)

#Parameters
#n_dim = train_X.shape[1]
#n_classes = train_y.shape[1]

#n_hidden_units_1 = n_dim
#n_hidden_units_2 = 400
#n_hidden_units_3 = 200
#n_hidden_units_4 = 100

#Define model
#def create_model(activation_function='relu', init_type='normal',
optimizer='adam', dropout_rate=0.2):
#    model = Sequential()
#    #Layer 1
#    model.add(Dense(n_hidden_units_1, input_dim = n_dim,
kernel_initializer=init_type, activation=activation_function))
#    #Layer 2
#    model.add(Dense(n_hidden_units_2,kernel_initializer=init_type,
activation=activation_function))
#    model.add(Dropout(0.2))
#    #Layer 3
#    model.add(Dense(n_hidden_units_3, kernel_initializer=init_type,
activation=activation_function))
#    model.add(Dropout(0.2))
#    #Layer 4
#    model.add(Dense(n_hidden_units_4, kernel_initializer=init_type,
activation=activation_function))
#    model.add(Dropout(0.2))
#    #Output layer
#    model.add(Dense(n_classes, kernel_initializer=init_type,
activation='softmax'))

    #Model compilation
#    model.compile(loss="categorical_crossentropy", optimizer=optimizer,
metrics=['accuracy'])
 #   return model

#Create the model
#model=create_model()

# Model Training
#lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=20,
min_lr=0.000001)
# Please change the model name accordingly.
#mcp_save = ModelCheckpoint('model/hello.h5', save_best_only=True,
monitor='val_accuracy', mode='max')

#Train the model
#import time
#start=time.time()
#history = model.fit(train_X, train_y, epochs=200, batch_size=4,
validation_data=(test_X, test_y), callbacks=[mcp_save, lr_reduce])
#end=time.time()
#print(end-start)
```

```
#Prediction
#predict=model.predict(test_X, batch_size=4)

#convert PREDICTED probabilities to emotions
#emotions=['neutral','calm','happy','sad','angry','fearful','disgused','surprised'
]
#y_pred=np.argmax(predict, axis=1)
#predicted emotions of test dataset
#predicted_emo=[]
#for i in range(test_y.shape[0]):
#    emo=emotions[y_pred[i]]
#    predicted_emo.append(emo)

#actual emotions of test dataset
#actual_emo=[]
#y_true=np.argmax(test_y, axis=1)
#for i in range(test_y.shape[0]):
#    emo=emotions[y_true[i]]
#    actual_emo.append(emo)

#Creating a confusion matrix
#cm=confusion_matrix(actual_emo, predicted_emo)
#index = ['angry', 'calm', 'disgust', 'fearful', 'happy', 'neutral', 'sad',
'surprised']
#columns = ['angry', 'calm', 'disgust', 'fearful', 'happy', 'neutral', 'sad',
'surprised']
#cm_df = pd.DataFrame(cm,index,columns)
#plt.figure(figsize=(10,10))
#sns.heatmap(cm_df, annot=True)

#Training accuracy
accuracy=accuracy_score(actual_emo, predicted_emo)

#Plots
# Plotting the Train Valid Loss Graph

#plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
#plt.title('model accuracy'+str(max(history.history['val_accuracy'])))
#plt.ylabel('accuracy')
#plt.xlabel('epoch')
#plt.legend(['train', 'test'], loc='upper left')
#plt.show()
```

6.1.3 SPEECH TRAIN

```
#Import libraries
import glob
import os
import librosa
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.layers import Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint,
EarlyStopping
```

```
import keras
from keras import regularizers

#Loading features and labels
X=np.load('X.npy') #features
y=np.load('y.npy') #labels

#Splitting the dataset
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=
0.3,random_state=42)

def get_network():
    input_shape = (193,)
    num_classes = 8
    keras.backend.clear_session()

    model = keras.models.Sequential()
    model.add(keras.layers.Dense(1024, activation="relu",
input_shape=input_shape))
    model.add(Dropout(0.5))
    model.add(keras.layers.Dense(512, activation="relu", input_shape=input_shape))
    model.add(keras.layers.Dense(256, activation="relu", input_shape=input_shape))
    model.add(keras.layers.Dense(128, activation="relu", input_shape=input_shape))
    model.add(keras.layers.Dense(num_classes, activation = "softmax"))
    model.compile(optimizer='adam',
        loss='categorical_crossentropy',
        metrics=["accuracy"])
    return model

model = get_network()

# Model Training
lr_reduce = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=20,
min_lr=0.000001)
# Please change the model name accordingly.
mcp_save = ModelCheckpoint('model/hello.h5', save_best_only=True,
monitor='val_accuracy', mode='max')

#callbacks = [EarlyStopping(monitor='val_loss', mode='min', patience=20),
mcp_save, lr_reduce]

history=model.fit(train_X, train_y, epochs = 700, batch_size = 24,
validation_data=(test_X, test_y), callbacks=[mcp_save, lr_reduce])

#l, a = model.evaluate(x_test, y_test, verbose = 0)

#Plots
# Plotting the Train Valid Loss Graph

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy: '+str(max(history.history['val_accuracy'])))
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

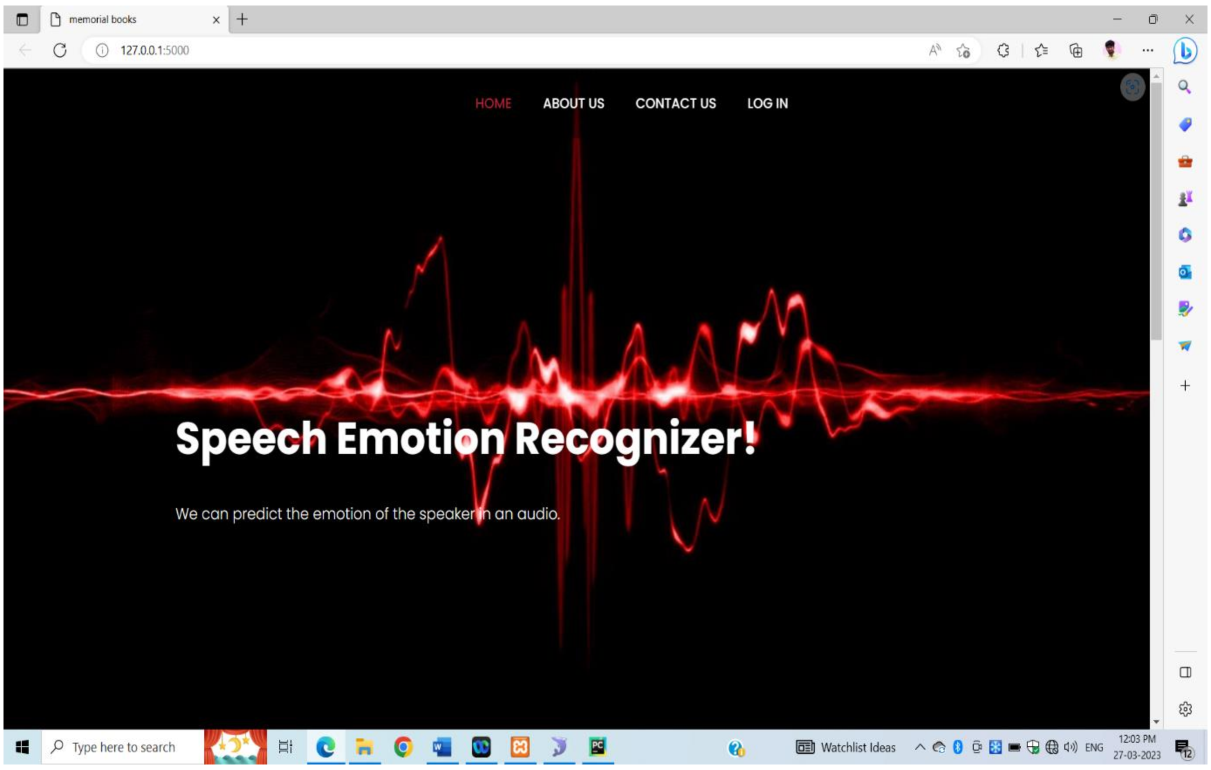
7 TESTING

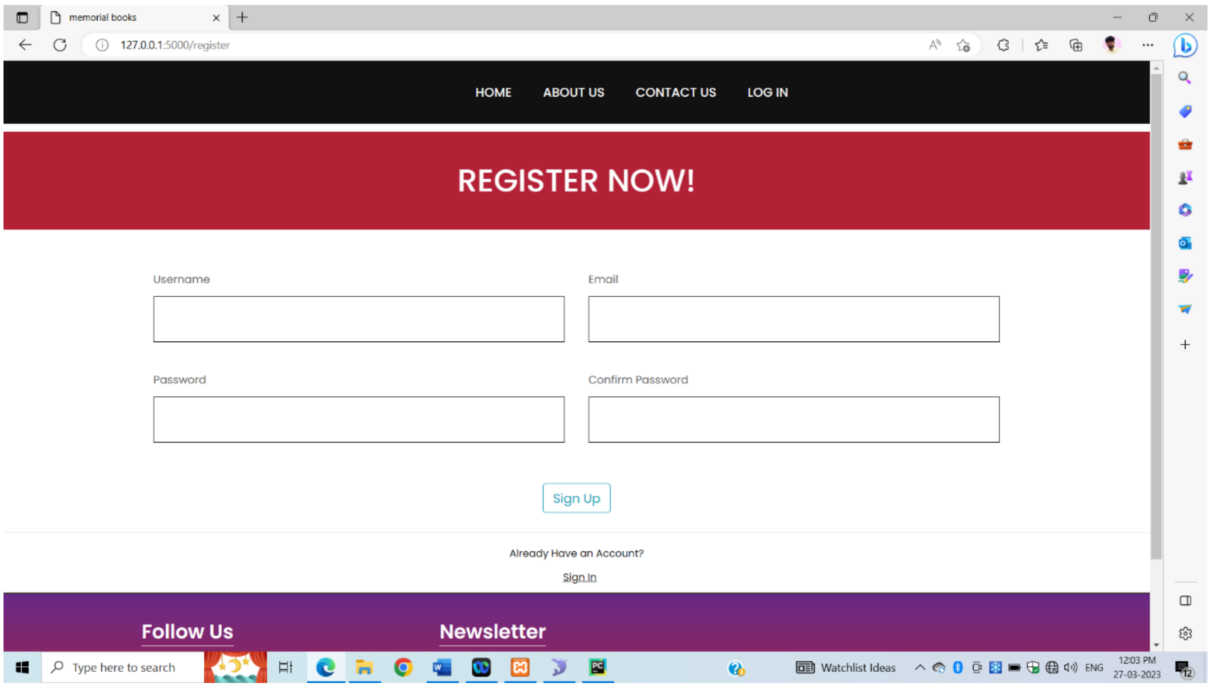
The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

8 EXPERIMENTAL RESULTS

8.1 Output




memorial books
127.0.0.1:5000/register

HOME ABOUT US CONTACT US LOG IN

REGISTER NOW!

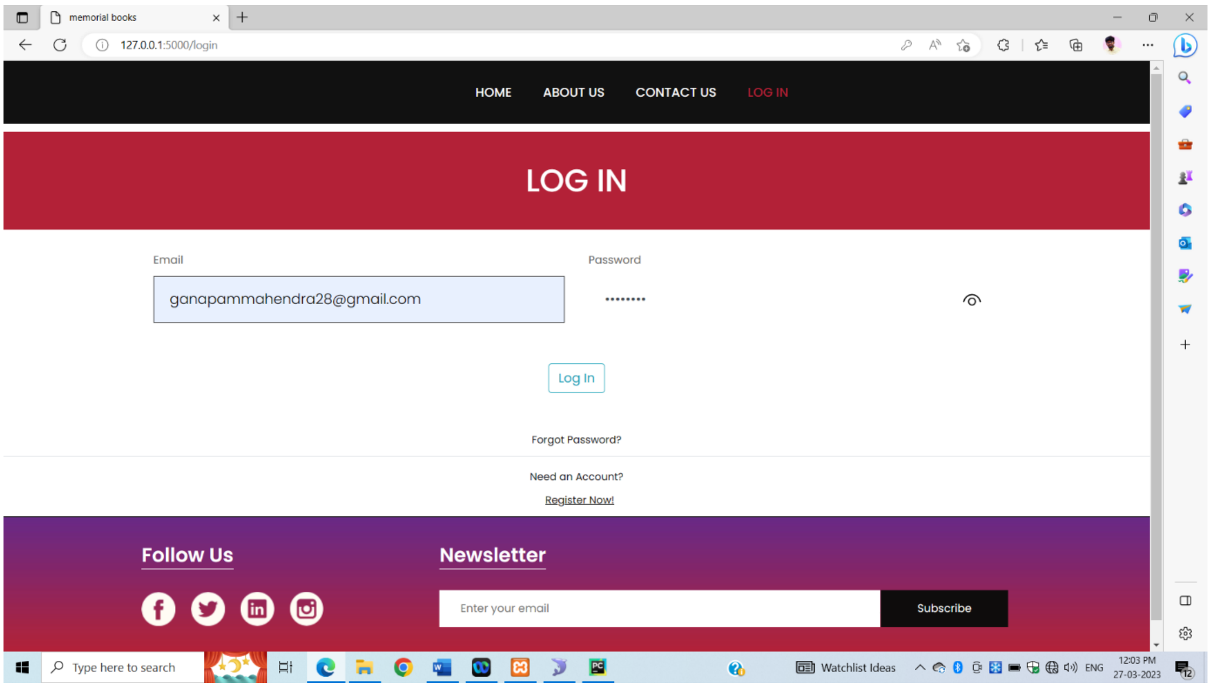
Username
Email
Password
Confirm Password

Sign Up

Already Have an Account?
Sign In

Follow Us
Newsletter

Type here to search
Watchlist Ideas
ENG
12:03 PM
27-03-2023


memorial books
127.0.0.1:5000/login

HOME ABOUT US CONTACT US LOG IN

LOG IN

Email: ganapammahendra28@gmail.com
Password: ••••••••

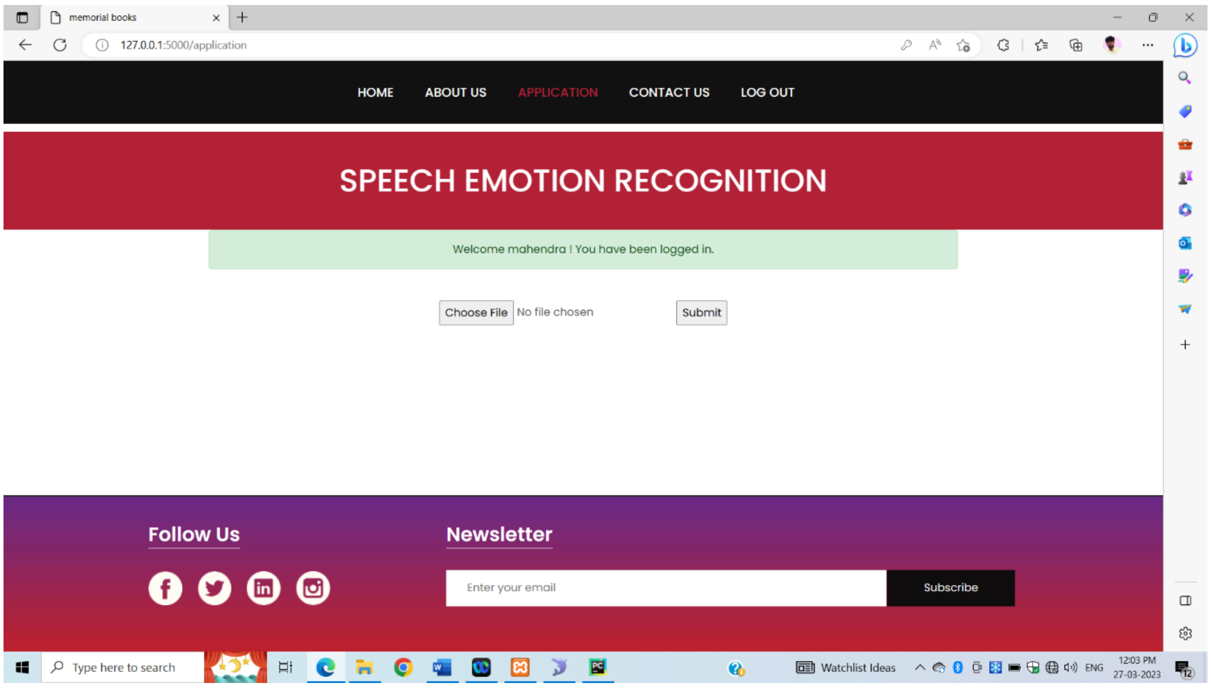
Log In

Forgot Password?

Need an Account?
Register Now!

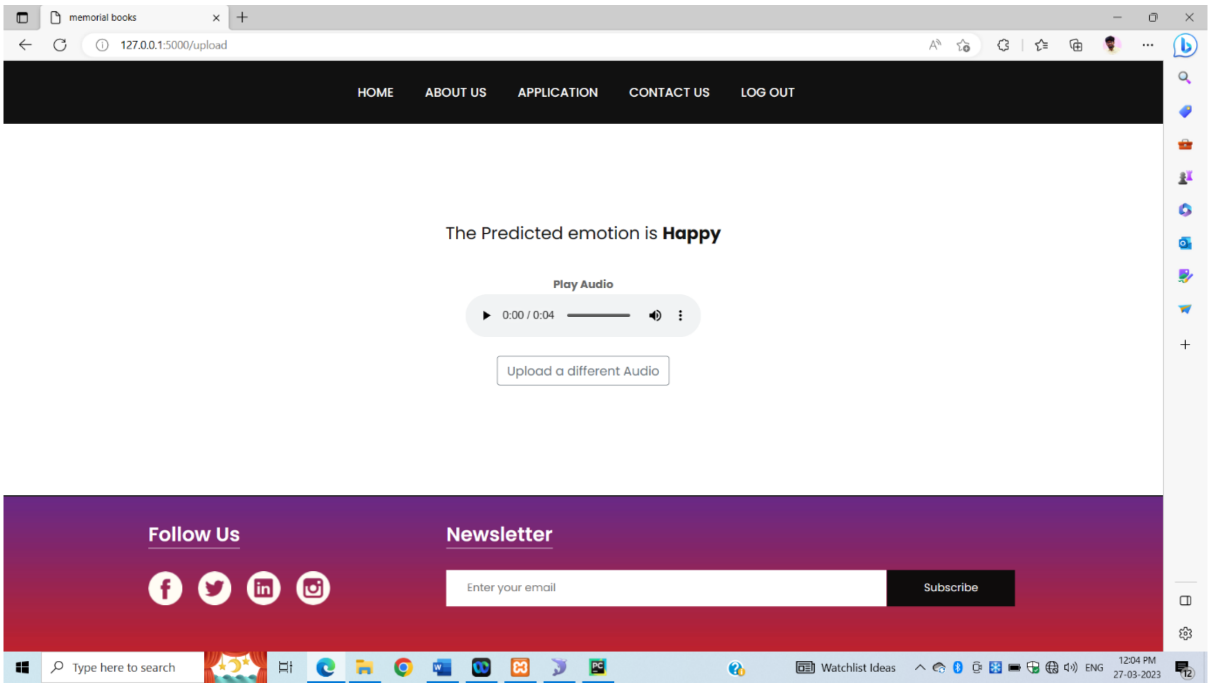
Follow Us
Newsletter
Enter your email
Subscribe

Type here to search
Watchlist Ideas
ENG
12:03 PM
27-03-2023



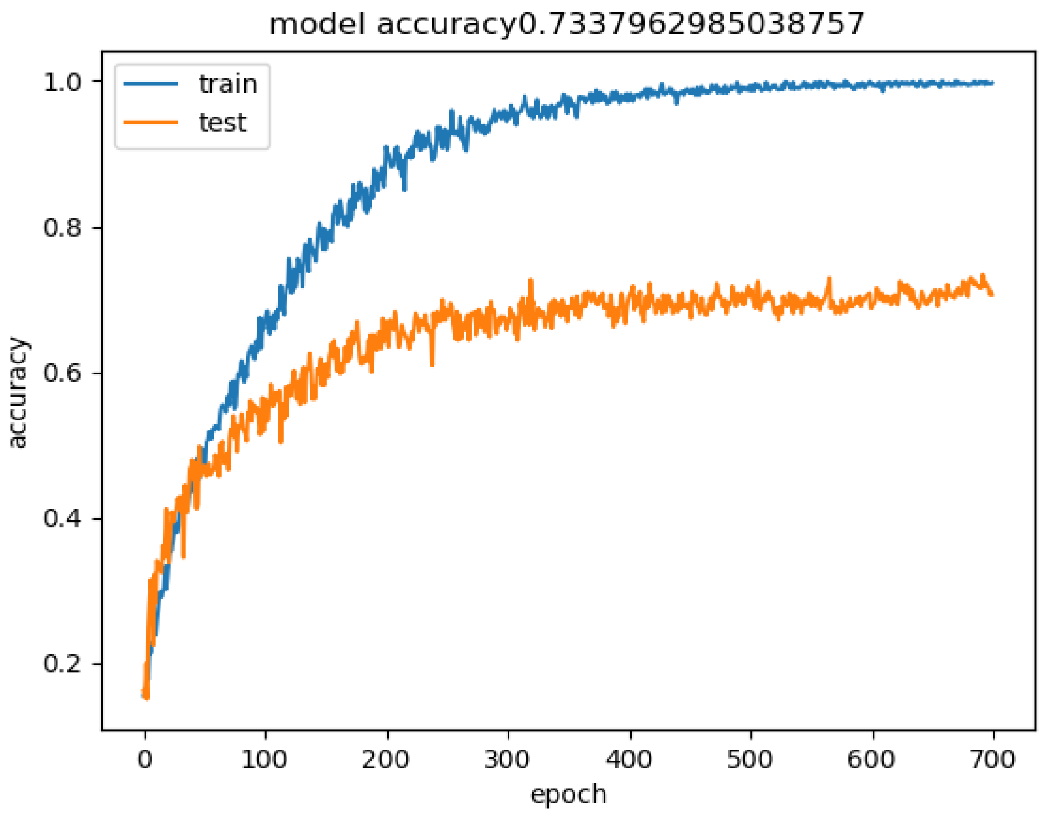
SPEECH EMOTION RECOGNITION

Welcome mahendra ! You have been logged in.



The Predicted emotion is **Happy**

8.2 Accuracy



9 CONCLUSIONS

The proposed scheme presented an approach to recognize the emotion from the human speech. This approach has been implemented by the using the neural networks. We have successfully developed a deep learning model using the deep neural network architecture to predict the emotions of the speaker in an audio. We have famed our project in a web based application using the Flask architecture. The UI also includes user registration system. We were able to get a test accuracy of 73.4% using the trained model.Please note that emotion prediction is subjective and the emotions rated by a person for the same audio can differ from person to person. This is also the reason why the algorithm which is trained on human rated emotions can generate erratic results sometimes. The model was trained of RAVDESS dataset, so the accent of the speaker can also lead to erratic results as the model is only trained on North American accent database.

10 FUTURE WORK

The ability to record a voice live and to predict the emotions in real time as the speaker is speaking. It also requires the knowledge of signal processing as the voice needs to be cleaned of all the unwanted noises in them before prediction.

11 REFERENCES

[1] Szegedy, Christian & Toshev, Alexander & Erhan, Dumitru. (2013). Deep Neural Networks for Object Detection. 1-9.

[2] Benk, Sal & Elmir, Youssef & Dennai, Abdeslem. (2019). A Study on Automatic Speech Recognition. 10. 77-85. 10.6025/jitr/2019/10/3/77-85.

[3] Ashish B. Ingale & D. S. Chaudhari (2012). Speech Emotion Recognition. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2 Issue-1, March 2012

[4] Chenchen Huang, Wei Gong, Wenlong Fu, Dongyu Feng, "A Research of Speech Emotion Recognition Based on Deep Belief Network and SVM", *Mathematical Problems in Engineering*, vol. 2014, Article
ID 749604, 7 pages, 2014. https://doi.org/10.1155/2014/749604

[5] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391.

[6] M. E. Ayadi, M. S. Kamel, F. Karray, “Survey on Speech Emotion Recognition: Features, Classification Schemes, and Databases”, Pattern Recognition 44, PP.572-587, 2011.

[7] I. Chiriacescu, “Automatic Emotion Analysis Based On Speech”, M.Sc. THESIS Delft University of Technology, 2009.

[8] T. Vogt, E. Andre and J. Wagner, “Automatic Recognition of Emotions from Speech: A review of the literature and recommendations for practical realization”, LNCS 4868, PP.75-91, 2008.

[9] S. Emerich, E. Lupu, A. Apatean, “Emotions Recognitions by Speech and Facial Expressions Analysis”, 17th European Signal Processing Conference, 2009.

[10] P.Shen, Z. Changjun, X. Chen, “Automatic Speech Emotion Recognition Using Support Vector Machine”, International Conference On Electronic And Mechanical Engineering And Information Technology, 2011.