**Emotion Detection using speech**

*A project report submitted to*

**Osmania University**

In Partial fulfilment of the requirements

for the award of the degree of

**Bachelor of Engineering**

**In**

**Computer Science and Engineering**

Submitted by

B.E VIII semester

**Prabhala Datta Parjanya Saketh (1608-17-733-063)**

**V. Shashi Chandra Reddy (1608-17-733-067)**

**D.Dinesh (1608-17-733-069)**

Under the Esteemed Guidance of

**Mrs. M. Swapna Reddy**

****

**Department of Computer Science and Engineering**

**(Accredited by NBA)**

# **Matrusri Engineering College**

**(Affiliated to Osmania University, Approved by AICTE)**

**Saidabad, Hyderabad-500059, Telangana, India)**

**(2020-21)**

Department of Computer Science and Engineering



**CERTIFICATE**

This is to certify that the project entitled **“Emotion Detection using Speech”** submitted by **P.Datta Parjanya Saketh, V. Shashi Chandra Reddy and D.Dinesh** bearing **HT .No: 1608-17-733-063, 1608-17-733-067, 1608-17-733-069** in the partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering** is a bonafide work carried by them.

The results of the investigations enclosed in this report have been verified and found satisfactory.

**M. Swapna Reddy Dr. P. Vijaya Pal Reddy**

**Assistant Professor Professor & Head**

**Dept. of CSE Dept. of CSE**

**Project Guide H.O.D.**

**External Examiner(s)**

**ACKNOWLEDGEMENT**

This project consumed huge amount of work, research and dedication. Still implementation would not have been possible if we did not have support from our Project Guide, Project Coordinator’s, Head of the Department and Principal. Therefore, we like to extend our sincere gratitude to all of them.

We are grateful to our project guide Mrs. M.Swapna Reddy, Assistant Professor, Dept. of CSE, Matrusri Engineering College for provision of expertise, technical support and guidance in the implementation.

We wish to express our gratitude to project coordinators **Mrs. B.J.Praveena & Mrs. K.Bhagyalaxmi,** for their indefatigable inspiration, constructive criticism and encouragement throughout this dissertation work.

We would like to express our sincere thanks to the Professor and Head of the Department, **Dr. P. Vijaya Pal Reddy**, for permitting us to do this project.

We would like to express our gratitude to **Dr. D. Hanumantha Rao**, principal of Matrusri Engineering College who permitted to carry out this project as per the academics.

We would like to thank **CSE** Department for providing us this opportunity to share and contribute our part of work to accomplish the project in time and all the teaching and support staff for their steadfast support and encouragement.

Nevertheless, we express our gratitude towards our families and colleagues for their kind cooperation and encouragement which helped us in completion of this project.

## **DECLARATION**

We hereby declare that the project entitled **“Emotion Detection using Speech”** is submitted to Osmania University in the partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering.**

This is a record of the bonafide work carried out by us under the guidance of Mrs. M.Swapna Reddy, Assistant Professor, Matrusri Engineering College, Saidabad, Hyderabad. The Results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Prabhala Datta Parjanya Saketh 1608-17-733-063

V. Shashi Chandra Reddy 1608-17-733-067

D. Dinesh 1608-17-733-069

Department of CSE

ABSTRACT

The aim of our project is about the finding of the emotions that will be resided in the speaker while talking. As an example, speech produced in a state of fear, anger, or joy will be loud and fast, with a higher and wider range in pitch, whereas the emotions such as sadness or tiredness generates slow and low-pitched speech.

Detection of human emotions through voice and speech-pattern analysis has many applications such as better assisting human and machine interactions. Particularly, we are presenting a classification models of emotions extracted by speeches based on Deep Neural Networks (CNNs), Support Vector Machine (SVM), Multilayer Perceptron (MLP) Classification based on acoustic features such as Mel Frequency Cepstral Coefficient (MFCC). The models have been trained to classify different emotions like neutral, calm, happy, sad, angry, fearful, disgust, surprise. Our evaluation shows that the proposed approach yields accuracies of ""86%, 84% and 82% using CNN, MLP and SVM respectively,"" for 7 emotions using Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset and Toronto Emotional Speech Set (TESS) Dataset.

CONTENTS

1. INTRODUCTION 9

1.1 Abstract

1.2 Purpose of the project 10

1.3 Problem in existing system 10

1.4 Solution for problem statement 10

2. LITERATURE SURVEY 11-12

3. SYSTEM ANALYSIS 13

3.1 Tensor Flow 13

3.1.1 History of Tensor Flow 14

3.1.2 Tensor Flow Architecture 14

3.1.3 Tensor 14

3.2 Keras 15

3.2.1 Backend 15

3.2.2 Advantages of Keras 16

3.2.3 Disadvantages of Keras 17

3.3 MatPlotlib 17

3.4 Scikit – Learn

3.5 Requirements

3.5.1 Software Requirements 21

3.5.2 Technologies 21

4. SYSTEM DESIGN 22

4.1 Activity Diagram 22

4.2 State chart Diagram 23

4.3 Sequence Diagram 24

5. IMPLEMENTATION 27

5.1 Architecture Diagram 27

5.2 Loading dataset 28

5.3 Data Pre-processing 28

5.4 Building Model 29

5.5 Compiling Model 30

5.6 Creating GUI 31

5.6 Training the Model 31

5.7 Pseudo code 32-36

6. SOFTWARE TESTING 37-38

7. RESULTS 39-43

8. CONCLUSION 44

8.1 Future Scope 44

9. REFERENCES 45

1.INTRODUCTION

1.1 Abstract

Speech Emotion Recognition, abbreviated as SER, is the act of attempting to recognize human emotion and affective states from speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch. This is also the phenomenon that animals like dogs and horses employ to be able to understand human emotion. SER is tough because emotions are subjective and annotating audio is challenging.Speech emotion recognition is a technology that extracts emotional features from speech signals by computer and contrasts and analyses the characteristic parameters and the emotional change acquired. Finally, the law of speech and emotion was concluded and speech emotional states were judged according to the law. At present, speech emotion recognition was an emerging crossing field of artificial intelligence and artificial psychology; besides, it was a hot research topic of signal processing and pattern recognition. The research was widely applied in human-computer interaction, interactive teaching, entertainment, security fields, and so on.

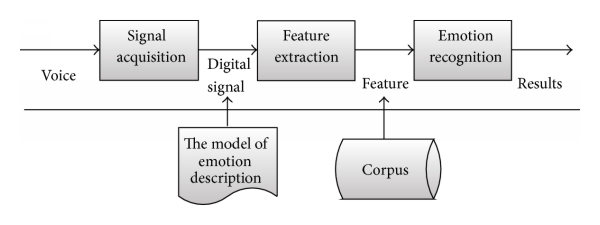
Speech emotion processing and recognition system was generally composed of three parts, which were speech signal acquisition, feature extraction, and emotion recognition. System framework is shown in Figure [1](https://www.hindawi.com/journals/mpe/2014/749604/fig1/).



FIGURE 1

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 extraction, 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.

1.2 Purpose of the project

Emotions are fundamental for humans, impacting perception and everyday activities such as communication, learning and decision-making. They are expressed through speech, facial expressions, gestures and other non-verbal clues. Speech emotion detection refers to analysing vocal behaviour as a marker of affect, with focus on the nonverbal aspects of speech. Its basic assumption is that there is a set of objectively measurable parameters in voice that reflect the affective state a person is currently expressing. This assumption is supported by the fact that most affective states involve physiological reactions which in turn modify the process by which voice is produced. For example, anger often produces changes in respiration and increases muscle tension, influencing the vibration of the vocal folds and vocal tract shape and affecting the acoustic characteristics of the speech. So far, vocal emotion expression has received less attention than the facial equivalent, mirroring the relative emphasis by pioneers such as Charles Darwin.

In the past, emotions were considered to be hard to measure and were consequently not studied by computer scientists. Although the field has recently received an increase in contributions, it remains a new area of study with a number of potential applications. These include emotional hearing aids for people with autism; detection of an angry caller at an automated call centre to transfer to a human; or presentation style adjustment of a computerised e-learning tutor if the student is bored.

A new application of emotion detection proposed in this dissertation is speech tutoring. Especially in persuasive communication, special attention is required to what non-verbal clues the speaker conveys. Untrained speakers often come across as bland, lifeless and colourless. Precisely measuring and analysing the voice is a difficult task and has in the past been entirely subjective. By using a similar approach as for detecting emotions, this report shows that such judgements can be made objective.

1.3 Problem in existing system

Firstly, discovering which features are indicative of emotion classes is a difficult task. The key challenge, in emotion detection and in pattern recognition in general, is to maximise the between-class variability whilst minimising the within class variability so that classes are well separated. However, features indicating different emotional states may be overlapping, and there may be multiple ways of expressing the same emotional state. One strategy is to compute as many features as possible. Optimisation algorithms can then be applied to select the features contributing most to the discrimination while ignoring others, creating a compact emotion code that can be used for classification. This avoids making difficult a priori assumptions about which features may be relevant.

Secondly, previous studies indicate that several emotions can occur simultaneously. For example, co-occurring emotions could include being happy at the same time as being tired, or feeling touched, surprised and excited when hearing good news. This requires a classifier that can infer multiple temporally co-occurring emotions.

Thirdly, real-time classification will require choosing and implementing efficient algorithms and data structures. Despite there existing some working systems, implementations are still seen as challenging and are generally expected to be imperfect and imprecise.

1.4 Solution for Problem Statement

2. LITERATURE SURVEY

It is argued that heart rate variation (HRV) has some dependency on the emotional state of a person. The results of the study revealed that the emotional stress can cause death due to heart disorders like acute myocardial infarction. It can also make predictions on risk of developing hypertension. However, it was proved that positive emotions can be beneficial in treatment of that hypertension by causing alterations in HRV. The idea proposed in used speech parameters for detection and analysis of the vocal fold pathology. There were developed algorithms for speech signal processing in order to create a model for characterizing healthy and pathology conditions of human vocal folds. The methodology was based on extraction of the separate speech signal components from both healthy and assumed pathology conditions. The iterative maximum likelihood (ML) estimation was applied for solving that problem. An algorithm was developed for the detection of hypernasal resonance. This approach is important because it provides noninvasive contactless interaction with patients, hence maximizing the accurate detection of speech due to the naturalness of speaking in comfort conditions without extra devices on the face and body. The work studies the dependency of physiology with human emotions in application to HCI. The approach uses physiological characteristics like temperature and electro dermal activity as an input for emotion recognition. The results indicate the possibility of developing an emotional relationship between humans and computers which enables the development of a human-friendly personal robot. They proposed a method of measuring the heart rate of a patient sitting on a chair. The method was based on using electro-mechanical film and traditional ear lobe photo-plethysmo graph (PPG) embedded into the chair. The experiment was conducted on twenty four participants in order to demonstrate whether human emotions can be directed to computers in the same way as to society. They conducted an experiment of speech therapy through telemedicine technology with a group of patients. The experiment showed that interaction among patients through speech accelerates their recovery and positively influences the quality of life. The same could be improved by monitoring heart rate detected from the speech of participants and evaluating the progress of recovering as well as preventing any risks related to heart failure. They introduced a remote detection of the Body Mass Index from the speech signal. The novel approach was designed for remote monitoring the patients’ weight in order to control the risks of diseases and death that are resulted from underweighting or in opposite from overweighting. The importance of the telemedicine was also discussed in this study, showing its benefits from different aspects such as comfort, low cost, time efficiency, etc. The work proposed an innovative approach of using mobile phones for measuring the heart rate continuously. The design consisted of three sub-systems; the first one records the signals and performs an offline analysis of the heart rate; the second system provides support for remote real time monitoring of the detected electrocardiogram (ECG) signal by sending the data to the medical centre or doctor through certain communication media; the third system performs a local real time classification of collected data. The main advantage of this design is the support for mobility of both patient and doctor. This design can be improved by applying speech signal recording for evaluating the heart rate instead of using an ECG sensor that sends the signal to the mobile phone via bluetooth connection.

3. SYSTEM ANALYSIS

3.1 Tensor Flow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

3.1.1 History of TensorFlow

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.TensorFlow computations are expressed as stateful dataflow graphs. Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kube flow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript. In Jan 2019, Google announced TensorFlow 2.0.It became officially available in Sep 2019. In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

3.1.2 TensorFlow Architecture

Tensorflow architecture works in three parts:

* Preprocessing the data
* Build the model
* Train and estimate the model

It is called Tensorflow because it takes input as a multi dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

3.1.3 Tensor

Tensorflow's name is directly derived from its core framework: Tensor. In Tensorflow, all the computations involve tensors. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data types with a known (or partially known) shape. The shape of the data is the dimensionality of the matrix or array. A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a graph. The graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other.The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

3.2 Keras

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. Keras is a high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras in Python doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

3.2.1 Backend

Backend is a term in Keras that performs all low-level computation such as tensor products, convolutions and many other things with the help of other libraries such as Tensorflow or Theano. So, the "backend engine" will perform the computation and development of the models. Tensorflow is the default "backend engine" but we can change it in the configuration.

3.2.2 Advantages of Keras

### **Fast Deployment and Easy to understand**

* + Keras is very quick to make a network model. If you want to make a simple network model with a few lines, Python Keras can help you with that.

### **Large Community Support**

* + There are lots of AI communities that use Keras for their Deep Learning framework. Many of them publish their codes as well as tutorials to the general public.

### **Have multiple Backends**

* + You can choose Tensorflow, CNTK, and Theano as your backend with Keras. You can choose a different backend for different projects depending on your needs. Each backend has its own unique advantage.

### **Cross-Platform and Easy Model Deployment**

* + With a variety of supported devices and platforms, you can deploy Keras on any device like iOS with CoreML, Android with Tensorflow Android,Web browser with .js support, Cloud engine, Raspberry Pi.

### **Multi GPUs Support**

You can train Keras on a single GPU or use multiple GPUs at once. Because Keras has a built-in support for data parallelism so it can process large volumes of data and speed up the time needed to train it.

3.2.3 Disadvantages of Keras

* **Cannot handle low-level API**

Keras only handles high-level API which runs on top other frameworks or backend engines such as Tensorflow, Theano, or CNTK. So it's not very useful if you want to make your own abstract layer for your research purposes because Keras already have pre-configured layers.

3.3 MatPlotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter. Since then it has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell. Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

3.4 Scikit – Learn

**Scikit-learn** is an open-source Python library for machine learning. It supports state-of-the-art algorithms such as KNN, XGBoost, random forest, and SVM. It is built on top of NumPy. Scikit-learn is widely used in Kaggle competition as well as prominent tech companies. It helps in preprocessing, dimensionality reduction(parameter selection), classification, regression, clustering, and model selection.

Scikit-learn has the best documentation of all open source libraries. It provides you an interactive chart at <https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>.

Scikit-learn is not very difficult to use and provides excellent results. However, scikit learn does not support parallel computations. It is possible to run a deep learning algorithm with it but is not an optimal solution, especially if you know how to use TensorFlow.

3.5 Requirements

3.5.1 Software Requirements

* Chrome v8 engine
* I3 processor
* Windows 8 and above
* CPU dual core
* RAM - 4 GB
* Microphone

3.5.2 Technologies

* Python

1. **Librosa**:

It is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

* Machine Learning Models

1. **SVM:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

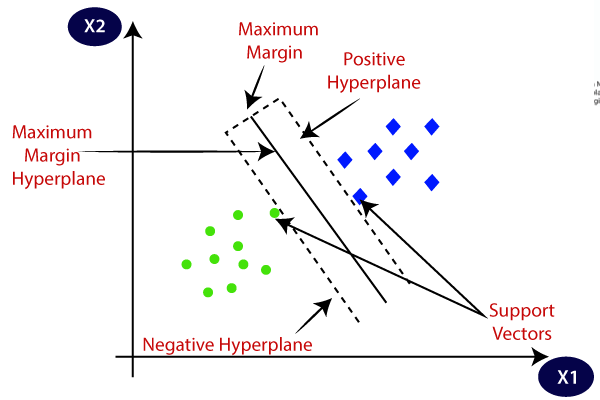


FIGURE 2

2.**MLP**:

A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation);. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

3.**CNN:**

Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural Network. In this blog, we are going to build basic building block for CNN.

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels incase of an image).
2. **Hidden Layer:** The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

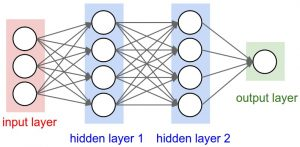
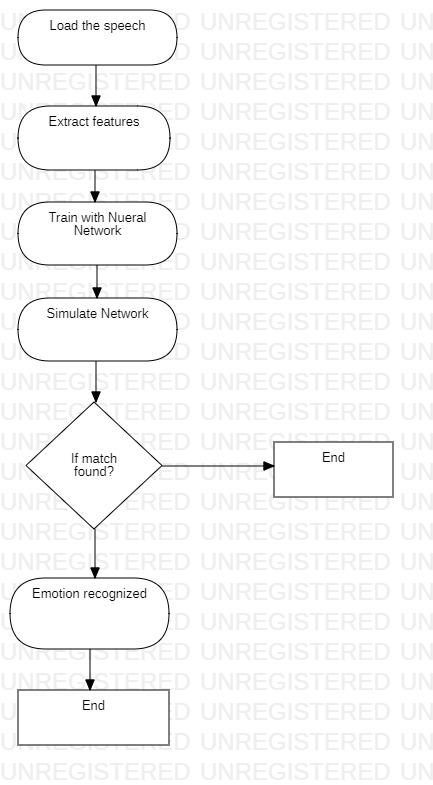
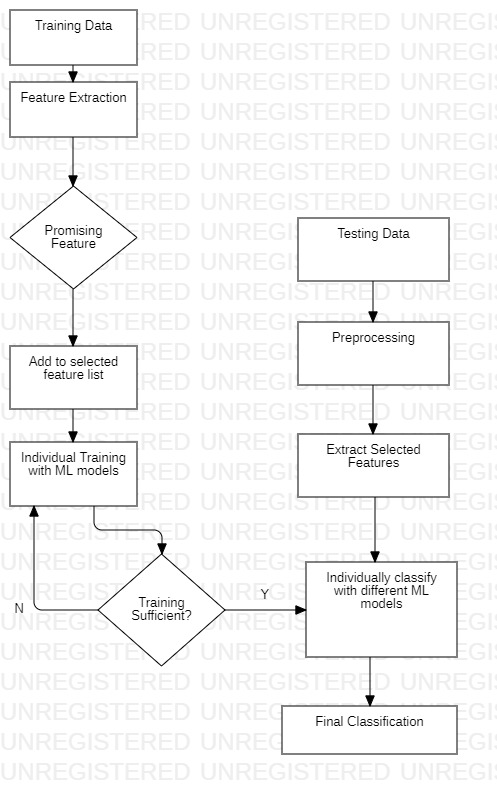


FIGURE 3

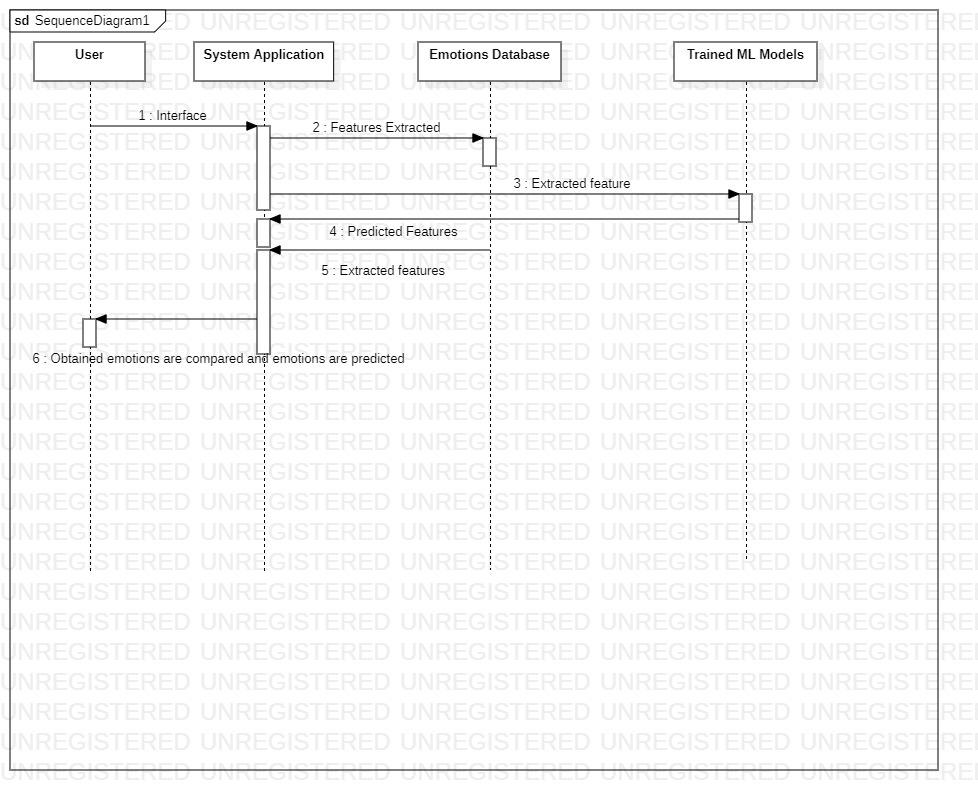
4. SYSTEM DESIGN

4.1 Flow Diagram

4.2 Activity Diagram



4.3 Sequence Diagram



Code

Mlp.ipynb

from google.colab import drive

drive.mount('/content/gdrive')

root\_path = 'gdrive/My Drive/majors/'

from google.colab import files

files.upload() #this will prompt you to upload the kaggle.json

!pip install -q kaggle

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

!ls ~/.kaggle

!chmod 600 /root/.kaggle/kaggle.json # set permission

!kaggle datasets download -d ejlok1/toronto-emotional-speech-set-tess

!kaggle datasets download -d uwrfkaggler/ravdess-emotional-speech-audio

!kaggle datasets download -d uwrfkaggler/ravdess-emotional-song-audio

!unzip ravdess-emotional-song-audio -d ravdesstotal

!unzip ravdess-emotional-speech-audio -d ravdesstotal

!mkdir TESS Toronto emotional speech set data

mkdir ravdesstotal/TESSTorontoemotionalspeechsetdata

!kaggle datasets download -d ejlok1/toronto-emotional-speech-set-tess --force

!unziptoronto-emotional-speech-set-tess -davdesstotal/TESSTorontoemotional

speechsetdata

!pip install librosa soundfile numpy sklearn pyaudio

!pip install soundfile

import librosa

import soundfile

import os, glob, pickle

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score

**def extract\_feature(file\_name, mfcc):**

**X, sample\_rate = librosa.load(os.path.join(file\_name), res\_type='kaiser\_fast')**

**result=np.array([])**

**if mfcc:**

**mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample\_rate, n\_mfcc=40).T, axis=0)**

**result=np.hstack((result, mfccs))**

**return result**

**emotions={**

**'01':'neutral',**

**'02':'calm',**

**'03':'happy',**

**'04':'sad',**

**'05':'angry',**

**'06':'fear',**

**'07':'disgust',**

**'08':'surprised'**

**}**

**#defined tess emotions to test on TESS dataset only**

**tess\_emotions=['angry','disgust','fear','ps','happy','sad']**

**##defined RAVDESS emotions to test on RAVDESS dataset only**

**ravdess\_emotions=['neutral','calm','angry', 'happy','disgust','sad','fear','surprised']**

**observed\_emotions = ['sad','angry','happy','disgust','surprised','neutral','calm','fear']**

**def dataset\_options():**

**# choose datasets**

**ravdess = True**

**tess = True**

**data = {'ravdess':ravdess, 'tess':tess}**

**print(data)**

**return data**

**# def load\_data(test\_size=0.2):**

**# x,y=[],[]**

**# for file in glob.glob('/content/ravdesstotal/Actor\_\*/\*.wav'):**

**# file\_name=os.path.basename(file)**

**# emotion=emotions[file\_name.split("-")[2]]**

**# if emotion not in observed\_emotions:**

**# continue**

**# feature=extract\_feature(file, mfcc=True, chroma=True, mel=True)**

**# x.append(feature)**

**# y.append(emotion)**

**# return train\_test\_split(np.array(x), y, test\_size=test\_size, train\_size= 0.75,random\_state=9)**

**def load\_data(test\_size=0.2):**

**x,y=[],[]**

**# feature to extract**

**mfcc = True**

**data = dataset\_options()**

**paths = []**

**if data['ravdess']:**

**paths.append('/content/ravdesstotal/Actor\_\*/\*.wav')**

**for path in paths:**

**for file in glob.glob(path):**

**file\_name=os.path.basename(file)**

**emotion=emotions[file\_name.split("-")[2]] #to get emotion according to filename. dictionary emotions is defined above.**

**if emotion not in observed\_emotions: #options observed\_emotions - RAVDESS and TESS, ravdess\_emotions for RAVDESS only**

**continue**

**feature=extract\_feature(file, mfcc)**

**x.append(feature)**

**y.append(emotion)**

**if data['tess']:**

**for file in glob.glob("/content/ravdesstotal/TESSTorontoemotionalspeechsetdata/TESS Toronto emotional speech set data/\*AF\_\*/\*"):**

**file\_name=os.path.basename(file)**

**emotion=file\_name.split("\_")[2][:-4] #split and remove .wav**

**if emotion == 'ps':**

**emotion = 'surprised'**

**if emotion not in observed\_emotions: #options observed\_emotions - RAVDESS and TESS, ravdess\_emotions for RAVDESS only**

**continue**

**feature=extract\_feature(file, mfcc)**

**x.append(feature)**

**y.append(emotion)**

**return train\_test\_split(np.array(x), y, test\_size=test\_size, train\_size= 0.75,random\_state=9)**

**count=0**

**For file in glob.glob("/content/ravdesstotal/TESSTorontoemotionalspeechsetdata/TESS Toronto emotional speech set data/\*AF\_\*/\*"):**

**count+=1**

**print(count)**

**# Split the dataset**

**import time**

**x\_train,x\_test,y\_train,y\_test=load\_data(test\_size=0.25)**

**#Get the shape of the training and testing datasets**

**print((x\_train.shape[0], x\_test.shape[0]))**

**# Get the number of features extracted**

**print(f'Features extracted: {x\_train.shape[1]}')**

**# Initialize the Multi Layer Perceptron Classifier**

**model=MLPClassifier(alpha=0.01, batch\_size=128, epsilon=1e-08, hidden\_layer\_sizes=(300,), learning\_rate='adaptive', max\_iter=500)**

**# Train the model**

**model.fit(x\_train,y\_train)**

**# Predict for the test set**

**y\_pred=model.predict(x\_test)**

**# Calculate the accuracy of our model**

**accuracy=accuracy\_score(y\_true=y\_test, y\_pred=y\_pred)**

**# Print the accuracy**

**print("Accuracy: {:.2f}%".format(accuracy\*100))**

**from sklearn.metrics import classification\_report**

**print(classification\_report(y\_test,y\_pred))**

**from sklearn.metrics import confusion\_matrix**

**matrix = confusion\_matrix(y\_test,y\_pred)**

**print (matrix)**

**SVM\_Feature\_Extraction.ipynb**

**from google.colab import drive**

**drive.mount('/content/gdrive')**

**root\_path = 'gdrive/My Drive/majors/'**

**from google.colab import files**

**files.upload() #this will prompt you to upload the kaggle.json**

**!pip install -q kaggle**

**!mkdir -p ~/.kaggle**

**!cp kaggle.json ~/.kaggle/**

**!ls ~/.kaggle**

**!chmod 600 /root/.kaggle/kaggle.json # set permission**

**!kaggle datasets download -d ejlok1/toronto-emotional-speech-set-tess**

**!kaggle datasets download -d uwrfkaggler/ravdess-emotional-speech-audio**

**!kaggle datasets download -d uwrfkaggler/ravdess-emotional-song-audio**

**!unzip ravdess-emotional-song-audio -d ravdesstotal**

**!unzip ravdess-emotional-speech-audio -d ravdesstotal**

**!mkdir TESS Toronto emotional speech set data**

**mkdir ravdesstotal/TESSTorontoemotionalspeechsetdata**

**!kaggle datasets download -d ejlok1/toronto-emotional-speech-set-tess --force**

**!unzip toronto-emotional-speech-set-tess -d ravdesstotal/TESSToronto**

**emotionalspeechsetdata**

**import glob**

**import os**

**import librosa**

**import time**

**import numpy as np**

**import pandas as pd**

**emotions={**

**'01':'neutral',**

**'02':'calm',**

**'03':'happy',**

**'04':'sad',**

**'05':'angry',**

**'06':'fear',**

**'07':'disgust',**

**'08':'surprised'**

**}**

**#defined tess emotions to test on TESS dataset only**

**tess\_emotions=['angry','disgust','fear','ps','happy','sad']**

**##defined RAVDESS emotions to test on RAVDESS dataset only**

**ravdess\_emotions=['neutral','calm','angry', 'happy','disgust','sad','fear','surprised']**

**observed\_emotions = ['sad','angry','happy','disgust','surprised','neutral','calm','fear']**

**def extract\_feature(file\_name, mfcc):**

**X, sample\_rate = librosa.load(os.path.join(file\_name), res\_type='kaiser\_fast')**

**result=np.array([])**

**if mfcc:**

**mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample\_rate, n\_mfcc=40).T, axis=0)**

**result=np.hstack((result, mfccs))**

**return result**

**def dataset\_options():**

**# choose datasets**

**ravdess = True**

**tess = True**

**data = {'ravdess':ravdess, 'tess':tess}**

**print(data)**

**return data**

**def load\_data(test\_size=0.2):**

**x,y=[],[]**

**# feature to extract**

**mfcc = True**

**data = dataset\_options()**

**paths = []**

**if data['ravdess']:**

**paths.append('/content/ravdesstotal/Actor\_\*/\*.wav')**

**for path in paths:**

**for file in glob.glob(path):**

**file\_name=os.path.basename(file)**

**emotion=emotions[file\_name.split("-")[2]] #to get emotion according to filename. dictionary emotions is defined above.**

**if emotion not in observed\_emotions: #options observed\_emotions - RAVDESS and TESS, ravdess\_emotions for RAVDESS only**

**continue**

**feature=extract\_feature(file, mfcc)**

**x.append(feature)**

**y.append(emotion)**

**if data['tess']:**

**for file in glob.glob("/content/ravdesstotal/TESSTorontoemotionalspeechsetdata/TESS Toronto emotional speech set data/\*AF\_\*/\*"):**

**file\_name=os.path.basename(file)**

**emotion=file\_name.split("\_")[2][:-4] #split and remove .wav**

**if emotion == 'ps':**

**emotion = 'surprised'**

**if emotion not in observed\_emotions: #options observed\_emotions - RAVDESS and TESS, ravdess\_emotions for RAVDESS only**

**continue**

**feature=extract\_feature(file, mfcc)**

**x.append(feature)**

**y.append(emotion)**

**return {"X":x,"y":y}**

**start\_time = time.time()**

**Trial\_dict = load\_data()**

**print("--- Data loaded. Loading time: %s seconds ---" % (time.time() - start\_time))**

**X = pd.DataFrame(Trial\_dict["X"])**

**y = pd.DataFrame(Trial\_dict["y"])**

**X.shape, y.shape**

**#renaming the label column to emotion**

**y=y.rename(columns= {0: 'emotion'})**

**#concatinating the attributes and label into a single dataframe**

**data = pd.concat([X, y], axis =1)**

**data.head()**

**#reindexing to shuffle the data at random**

**data = data.reindex(np.random.permutation(data.index))**

**# Storing shuffled ravdess and tess data to avoid loading again**

**data.to\_csv("RAVTESS\_MFCC\_Observed.csv")**

**SER\_SVM.ipynb**

**from google.colab import drive**

**drive.mount('/content/drive')**

**import pandas as pd**

**import numpy as np**

**import time**

**starting\_time = time.time()**

**data = pd.read\_csv('/content/drive/MyDrive/RAVTESS\_MFCC\_Observed.csv')**

**print("data loaded in " + str(time.time()-starting\_time) + "ms")**

**print(data.head())**

**data.shape**

**data.columns**

**data = data.drop('Unnamed: 0',axis=1)**

**data.columns**

**X = data.drop('emotion', axis = 1).values**

**y = data['emotion'].values**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**X.shape, y.shape**

**np.unique(y)**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20)**

**from sklearn.svm import SVC**

**svclassifier = SVC(kernel = 'linear')**

**import time**

**starting\_time = time.time()**

**svclassifier.fit(X\_train, y\_train)**

**print("Trained model in %s ms " % str(time.time() - starting\_time))**

**y\_pred = svclassifier.predict(X\_test)**

**from sklearn.metrics import classification\_report, confusion\_matrix,accuracy\_score**

**import seaborn as sn**

**print(classification\_report(y\_test,y\_pred))**

**acc = float(accuracy\_score(y\_test,y\_pred))\*100**

**print("----accuracy score %s ----" % acc)**

**cm = confusion\_matrix(y\_test,y\_pred)**

**df\_cm = pd.DataFrame(cm)**

**sn.heatmap(df\_cm, annot=True, fmt='')**

**plt.show()**

**train\_acc = float(svclassifier.score(X\_train, y\_train)\*100)**

**print("----train accuracy score %s ----" % train\_acc)**

**test\_acc = float(svclassifier.score(X\_test, y\_test)\*100)**

**print("----test accuracy score %s ----" % test\_acc)**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.pipeline import Pipeline**

**from sklearn.svm import SVC**

**#splitting dataset into train/ test sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20)**

**# Setup the pipeline steps: steps**

**steps = [('scaler', StandardScaler()),**

**('SVM', SVC())]**

**# Create the pipeline: pipeline**

**pipeline = Pipeline(steps)**

**# Fit the pipeline to the training set: svc\_scaled**

**svc\_scaled = pipeline.fit(X\_train, y\_train)**

**# Instantiate and fit a classifier to the unscaled data**

**svc\_unscaled = SVC(kernel = 'linear').fit(X\_train, y\_train)**

**# Compute and print metrics**

**print('Accuracy with Scaling: {}'.format(svc\_scaled.score(X\_test, y\_test)))**

**print('Accuracy without Scaling: {}'.format(svc\_unscaled.score(X\_test, y\_test)))**

**train\_acc = float(svc\_scaled.score(X\_train, y\_train)\*100)**

**print("----train accuracy score %s ----" % train\_acc)**

**test\_acc = float(svc\_scaled.score(X\_test, y\_test)\*100)**

**print("----test accuracy score %s ----" % test\_acc)**

**scaled\_predictions = svc\_scaled.predict(X\_test)**

**from sklearn.metrics**

**import classification\_report, confusion\_matrix,accuracy\_score**

**import seaborn as sn**

**print(classification\_report(y\_test,scaled\_predictions))**

**acc = float(accuracy\_score(y\_test,scaled\_predictions))\*100**

**print("----accuracy score %s ----" % acc)**

**cm = confusion\_matrix(y\_test,scaled\_predictions)**

**df\_cm = pd.DataFrame(cm)**

**sn.heatmap(df\_cm, annot=True, fmt='')**

**plt.show()**

**from sklearn.model\_selection**

**import cross\_val\_score**

**# no. of folds cv = 5**

**cv\_results = cross\_val\_score(svc\_scaled, X, y, cv = 5)**

**print(cv\_results)**