INTERNSHIP PROJECT REPORT
On
"SPEECH SIGNAL PROCESSING"

Submitted
By

S SAI SURYATEJA

Under the
Guidance of

Prof. RAM BILAS PACHORI

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JULY 2019

ACKNOWLEDGEMENTS

I wish to thank **Prof. Ram Bilas Pachori** for his kind support and valuable guidance. It is his help and support, due to which I became able to complete the “Internship training” report. Without the help of M. Tech and Ph. D students of the department, this report would not have been possible.

S Sai Suryateja
Roll No. 17BEC0420
B. Tech. II nd Year
Department of Electronics and Communication Engineering
Vellore Institute of Technology, Vellore

TABLE OF CONTENTS

CHAPTER 1:

SPECTROGRAM ANALYSIS OF SPEECH SIGNALS

INTRODUCTION:

- Speech is the most natural form of human-human communications.
- Speech is related to language; linguistics is a branch of social science.
- Speech is related to human physiological capability; physiology is a branch of medical science.
- Speech is also related to sound and acoustics, a branch of physical science.
- Therefore, speech is one of the most intriguing signals that humans work with every day.
- Purpose of speech processing: –
    - To understand speech as a means of communication;
    - To represent speech for transmission and reproduction;
    - To analyze speech for automatic recognition and extraction of information
    - To discover some physiological characteristics of the talker.

MY WORK:

To analyze the speech signal, we are using a basic method of the spectrogram representation of the signal.

The signal which is taken is having different vocals and instrumentals with different frequencies. So, first we have to identify the frequency of the vocal and then predict the speech signal.

So, we can go with two proposed methods i.e. either spectrogram function or short time Fourier Transform function.

Below given are the MATLAB codes and the outputs of the speech signal analysis of the audio files present with us.

MATLAB CODES USED:

- USING SHORT TIME FOURIER TRANSFORM
  - SHORT TIME FOURIER TRANSFORM (STFT) FUNCTION

```
function [stft, f, t] = stft(x, wlen, hop, nfft, fs)
x = x(:);
xlen = length(x);
win = hamming(wlen, 'periodic');
rown = ceil((1+nfft)/2);
coln = 1+fix((xlen-wlen)/hop);
stft = zeros(rown, coln);
indx = 0;

for col = 1:coln
xw = x(indx+1:indx+wlen).*win;
X = fft(xw, nfft);
stft(:, col) = X(1:rown);
indx = indx + hop;
end

t = (wlen/2:hop:wlen/2+(coln-1)*hop)/fs;
f = (0:rown-1)*fs/nfft;

end
```

  - MAIN FILE

```
clc;clear all;close all;
[x, fs] = audioread('1.mp3');
wlen = 1024;
hop = wlen/4;
nfft = 4096;
x = x(:, 1);
xlen = length(x);
a=fft(x);
[S, f, t] = stft(x, wlen, hop, nfft, fs);

s = pwelch(x);
plot(10*log10(s))
drawnow;
sound(x, Fs);

K = sum(hamming(wlen,'periodic'));
S = abs(S)/K;
if rem(nfft, 2)
S(2:end, :) = S(2:end, :).*2;
else
S(2:end-1, :) = S(2:end-1, :).*2;
end

S = 20*log10(S + 1e-6);

figure(1)
surf(t, f, S)
shading interp
xlabel('Time, s')
ylabel('Frequency, Hz')
zlabel('Magnitude, dB')
title('Amplitude spectrogram of the signal')
hold on;
```

- USING SPECTROGRAM FUNCTION (INBUILT)

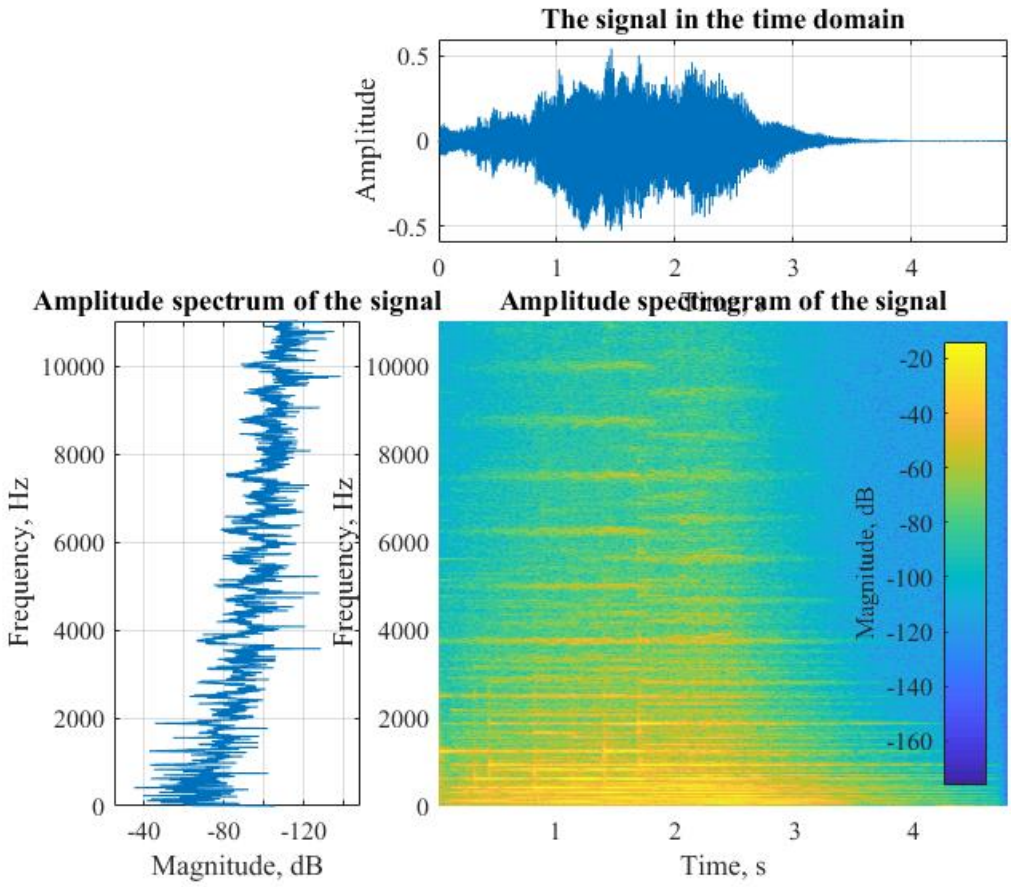
```
clc;clear all;close all;
[x, fs] = audioread('1.mp3');
wlen = 1024;
hop = wlen/4;
nfft = 4096;
x = x(:, 1);
xlen = length(x);
a=fft(x);
[S, f, t] = spectrogram(x, wlen, hop, nfft, fs);
s = pwelch(x);
plot(10*log10(s))
drawnow;
sound(x, Fs);

K = sum(hamming(wlen,'periodic'));
S = abs(S)/K;

if rem(nfft, 2)
S(2:end, :) = S(2:end, :).*2;
else
S(2:end-1, :) = S(2:end-1, :).*2;
end

S = 20*log10(S + 1e-6);

figure(1)
surf(t, f, S)
shading interp
xlabel('Time, s')
ylabel('Frequency, Hz')
zlabel('Magnitude, dB')
title('Amplitude spectrogram of the signal')
hold on;
```

THEORY:

This chapter is having a small database of thirty audio inputs divided into three groups namely: English, Hindi and Telugu.
Each dataset is having 10 audio inputs which can be seen in the link given in reference 1.

The output of the signal is a spectrogram and by using that spectrogram, we can easily depict few of the parameters such as peak frequencies, pitch range of the signal and many more.

An example of such signal is given below with its description:



The given output of the signal is having time domain analysis, frequency domain analysis and spectrogram analysis. The whole analysis database and outputs are in the reference 2.
The signal for this example is 5 seconds long and has a frequency range of 12000 Hz. The signal is having low amplitude at the lower frequencies i.e. in the vocal section, so it can be deduced that the speaker of the signal is either not having an audio or the speaker is having very low voice. Due to low amplitude, pitch can't be calculated.

CHAPTER 2:

TQWT BASED EEG SIGNAL PROCESSING

INTRODUCTION:

- The 'Tunable Q-Factor Wavelet Transform' (TQWT) is a flexible fully-discrete wavelet transforms.
- The TQWT toolbox is a set of MATLAB programs implementing and illustrating the TQWT.
- The programs validate the properties of the transform, clarify how the transform can be implemented, and show how it can be used.
- The TQWT is similar to the rational-dilation wavelet transform (RADWT), but the TQWT does not require that the dilation-factor be rational.

MY WORK:

In this chapter we are discussing about the disease named sleep apnea, and we are discussing about how to differentiate between seizure, non-seizure and seizure-free EEG signals.

So, we are using the TQWT (Tunable Q wavelet transform) to decompose the signal and we are analyzing and differentiating the signals. We are using box plots to differentiate the signals.

The function used to plot the box plot for all the mean frequencies of a particular set of EEG signals, so that we can differentiate he signals is **kruskalwallis**.

DATASET:

The recordings of EEG signals for both healthy and epileptic subjects are available in this dataset. The EEG signals are divided into five subsets namely, Z, O, N, F, and S. Each subset contains 100 single-channel EEG signals of duration 23.6 s. The EEG signals of subsets Z and O are acquired from five healthy subjects using standard electrode placement scheme [17]. These are the surface EEG recordings with eyes open and closed, respectively. The subsets F and N consist the EEG signals of seizure-free intervals from five patients. These signals are recorded in the epileptogenic zone for F subset and from the hippocampal formation of the opposite hemisphere of the brain for N subset. The fifth subset S includes the EEG signals of seizure activity and taken from all the recording sites which showed the ictal activity. The EEG signals of subsets N, F, and S are recorded using depth electrodes intracranially. The sampling rate of each recorded EEG signal is 173.61 Hz. EEG signals of normal, seizure-free and seizure classes.

MATLAB CODES USED:

- TWT FUNCTION

```
function[w]=TWT(x)
Q = 1; r = 3; J = 8;            % TQWT parameters
N = length(x);                  % Length of test signal
w = tqwt_radix2(x,Q,r,J);       % TQWT
for j=1:9
    w{1,j}=meanfreq(w{1,j});
end
```

- MAIN FILE

```
for k = 1:9
       % Create a text file name, and read the file.
    textFileName = ['F00' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
      Z(k,:)=TWT(textData);
        fclose(fid);
        else
        fprintf('File %s does not exist.\n', textFileName);
    end
end
    for k = 10:99
       % Create a text file name, and read the file.
    textFileName = ['F0' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
      Z(k,:)=TWT(textData);
        fclose(fid);
    else
        fprintf('File %s does not exist.\n', textFileName);
    end
    end
    for k = 100
       % Create a text file name, and read the file.
    textFileName = ['F' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
        Z(k,:)=TWT(textData);
        fclose(fid);
    else
        fprintf('File %s does not exist.\n', textFileName);
    end
end
```

- KRUSHALWALLIS

```
X=[us,uf];
p=kruskalwallis(X);
```

THEORY:

In the present work, an automatic diagnostic system for the identification of seizure EEG signals is suggested. This method is based on TQWT technique of signal decomposition.

The outputs of the different inputs are:

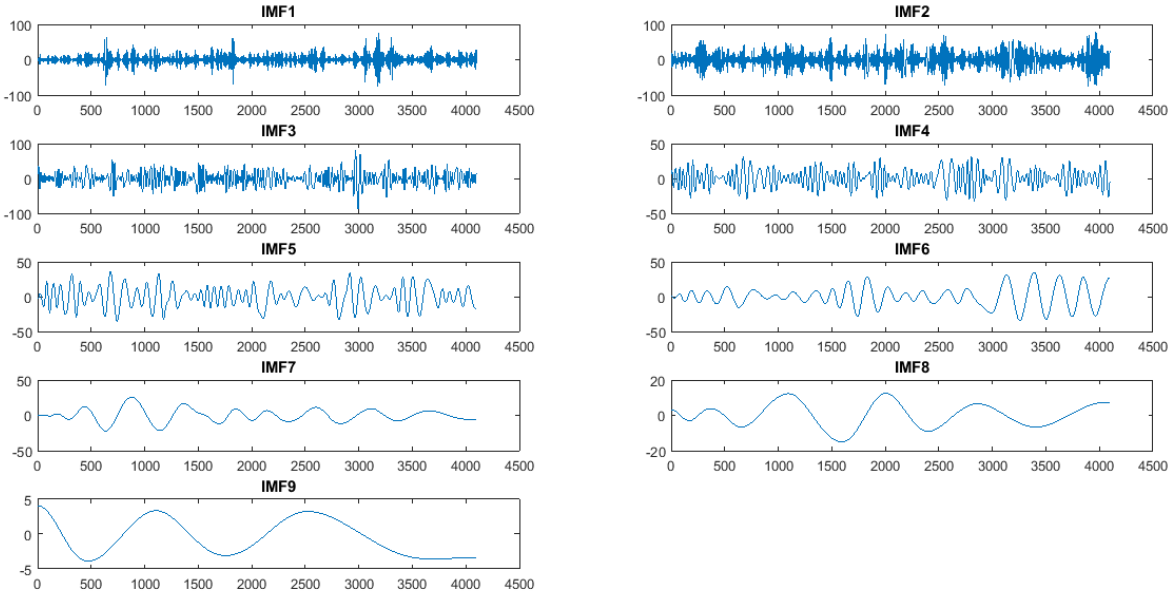


Fig1: IMF's of normal EEG signal

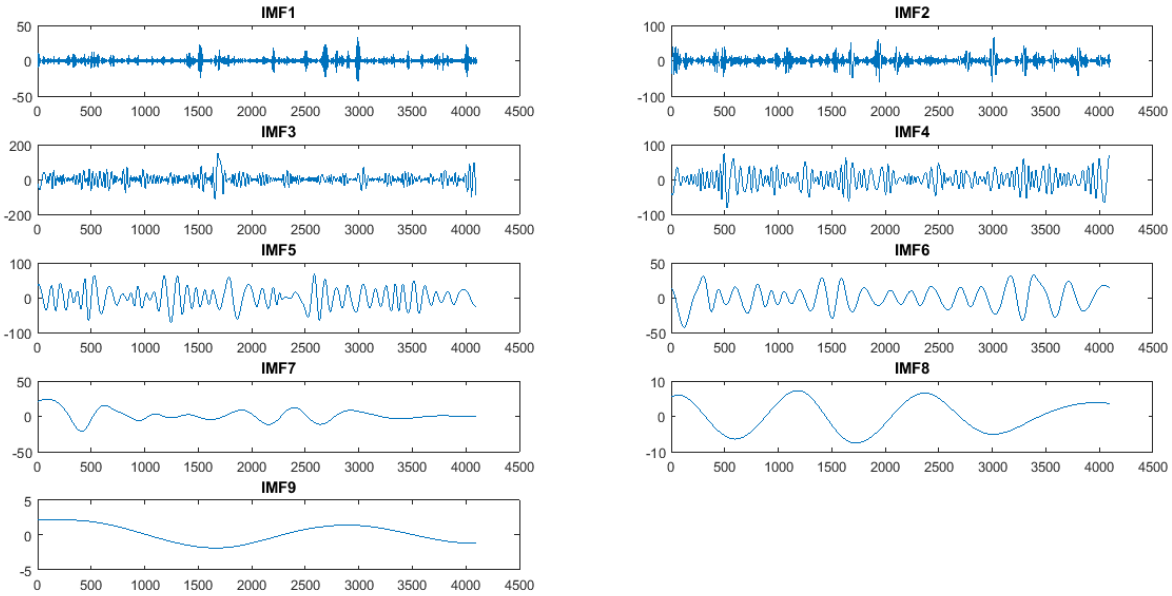


Fig2: IMF's of seizure-free EEG signals

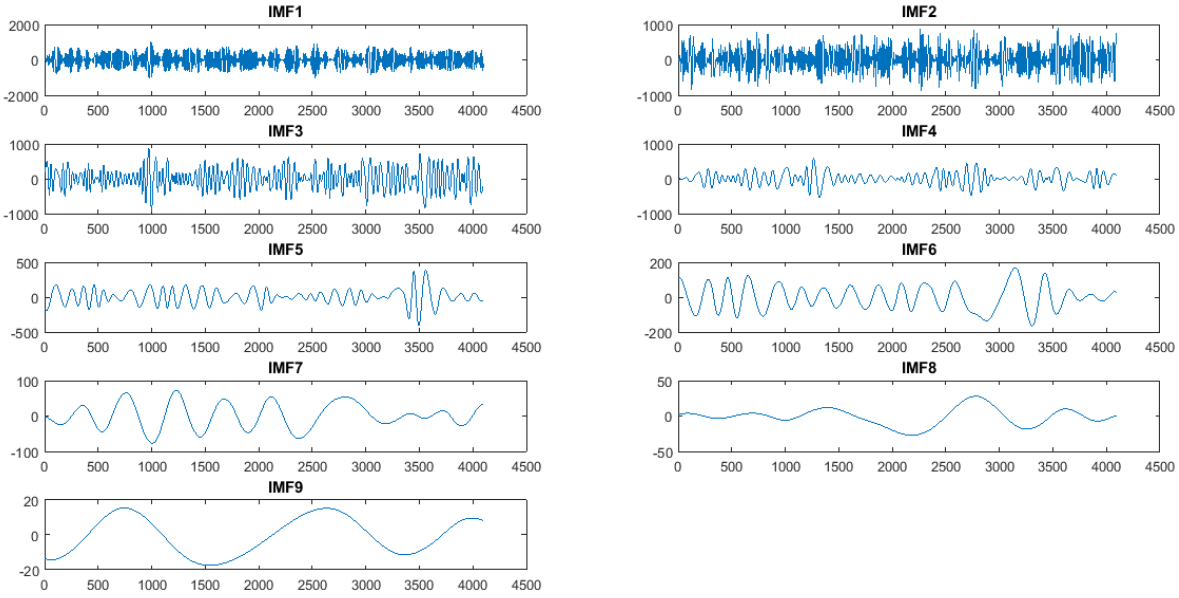
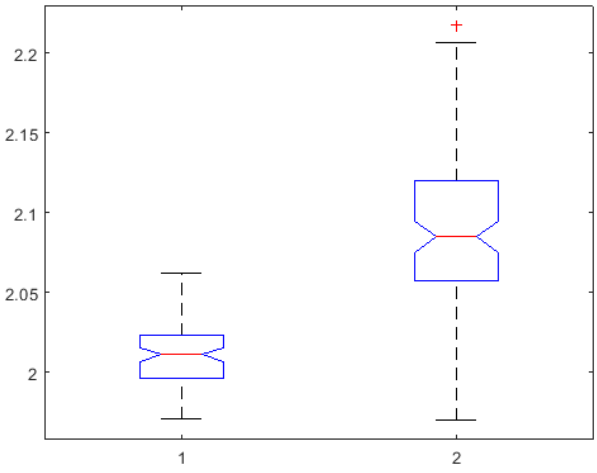


Fig3: IMF's of seizure EEG signal

Using these IMF's, we can predict whether the EEG signal is seizure, seizure-free or normal EEG signal. We predict this by using box plot of two subsets, i.e. by using mean frequencies of each EEG signal in a dataset combined as one using kruskalwallis function.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	344118.1	1	344118.1	102.72	3.85674e-24
Error	322531.9	198	1628.9		
Total	666650	199			



Using this method we can compare the mean frequencies of the datasets and we can compare the normal and seizure signals.

So, the error in this process is very small, so we can easily distinguish between normal, seizure and seizure-free signals.

CHAPTER 3:

EMD BASED EEG SIGNAL PROCESSING

INTRODUCTION:

The empirical mode decomposition method is a data adaptive and data-dependent method. This method does not require any assumption about the stationarity and linearity of the signals. Therefore, this method is a better choice for the analysis of non-stationary and nonlinear signals such as EEG. The main objective of the EMD technique is to decompose the nonlinear and non-stationary signal x(t) into a sum of intrinsic mode functions (IMFs). Each IMF satisfies two basic conditions namely, the number of extrema and the number of zero crossings must be the same or differ at most by one and the mean value of the envelope defined by the local maxima and the envelope defined by the local minima is zero at any point.

MY WORK:

In this chapter we are discussing about the disease named sleep apnea, and we are discussing about how to differentiate between seizure, non-seizure and seizure-free EEG signals.

So, we are using the EMD (Empirical Mode Decomposition) to decompose the signal and we are analyzing and differentiating the signals. We are using box plots to differentiate the signals.

The function used to plot the box plot for all the mean frequencies of a particular set of EEG signals, so that we can differentiate he signals is **kruskalwallis**.

DATASET:

The recordings of EEG signals for both healthy and epileptic subjects are available in this dataset. The EEG signals are divided into five subsets namely, Z, O, N, F, and S. Each subset contains 100 single-channel EEG signals of duration 23.6 s. The EEG signals of subsets Z and O are acquired from five healthy subjects using standard electrode placement scheme [17]. These are the surface EEG recordings with eyes open and closed, respectively. The subsets F and N consist the EEG signals of seizure-free intervals from five patients. These signals are recorded in the epileptogenic zone for F subset and from the hippocampal formation of the opposite hemisphere of the brain for N subset. The fifth subset S includes the EEG signals of seizure activity and taken from all the recording sites which showed the ictal activity. The EEG signals of subsets N, F, and S are recorded using depth electrodes intracranially. The sampling rate of each recorded EEG signal is 173.61 Hz. EEG signals of normal, seizure-free and seizure classes.

MATLAB CODES USED:

- EMDPROG FUNCTION

```
function[H]=emdprog(g)
g=emd(g);
figure;
subplot(521)
plot(g(1,:))
title('IMF1')
subplot(522)
plot(g(2,:))
title('IMF2')
subplot(523)
plot(g(3,:))
title('IMF3')
subplot(524)
plot(g(4,:))
title('IMF4')
subplot(525)
plot(g(5,:))
title('IMF5')
subplot(526)
plot(g(6,:))
title('IMF6')
subplot(527)
plot(g(7,:))
title('IMF7')
subplot(528)
plot(g(8,:))
title('IMF8')
subplot(529)
plot(g(9,:))
title('IMF9')
f=g(1,:)';
N=length(f);
nb=(1:N);
f=f';
MM=N;
if exist('alfa') == 0
    x=2;
    alfa=zeros(1,MM);
    for i=1:MM
        ex=1;
        while abs(ex)>.00001
            ex=-besselj(0,x)/besselj(1,x);
            x=x-ex;
        end
        alfa(i)=x;
        x=x+pi;
    end
end
a=N;
for m1=1:MM
    a3(m1)=(2/(a^2*(besselj(1,alfa(m1))).^2))*sum(nb.*f.*besselj(0,alfa(m1)/a*nb));
end
freq=(alfa)/(2*pi*length(f));
for m1=1:MM
E(m1)=((a3(m1)^2)*(MM^2)/2).*((besselj(1,alfa(m1)))^2);
end
fmean=[sum(freq.*E/sum(E))];
H=fmean;
```

- MAIN FILE

```
for k = 1:9
       % Create a text file name, and read the file.
    textFileName = ['F00' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
      Z(k,:)=TWT(textData);
        fclose(fid);
        else
        fprintf('File %s does not exist.\n', textFileName);
    end
end
    for k = 10:99
       % Create a text file name, and read the file.
    textFileName = ['F0' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
      Z(k,:)=TWT(textData);
        fclose(fid);
    else
        fprintf('File %s does not exist.\n', textFileName);
    end
    end
    for k = 100
       % Create a text file name, and read the file.
    textFileName = ['F' num2str(k) '.txt'];
    if exist(textFileName, 'file')
        fid = fopen(textFileName, 'rt');
        textData = fread(fid);
        Z(k,:)=TWT(textData);
        fclose(fid);
    else
        fprintf('File %s does not exist.\n', textFileName);
    end
end
```

- KRUSHALWALLIS

```
X=[us,uf];
p=kruskalwallis(X);
```

THEORY:

In the present work, an automatic diagnostic system for the identification of seizure EEG signals is suggested. This method is based on EMD technique of signal decomposition.

The outputs of the different inputs are:

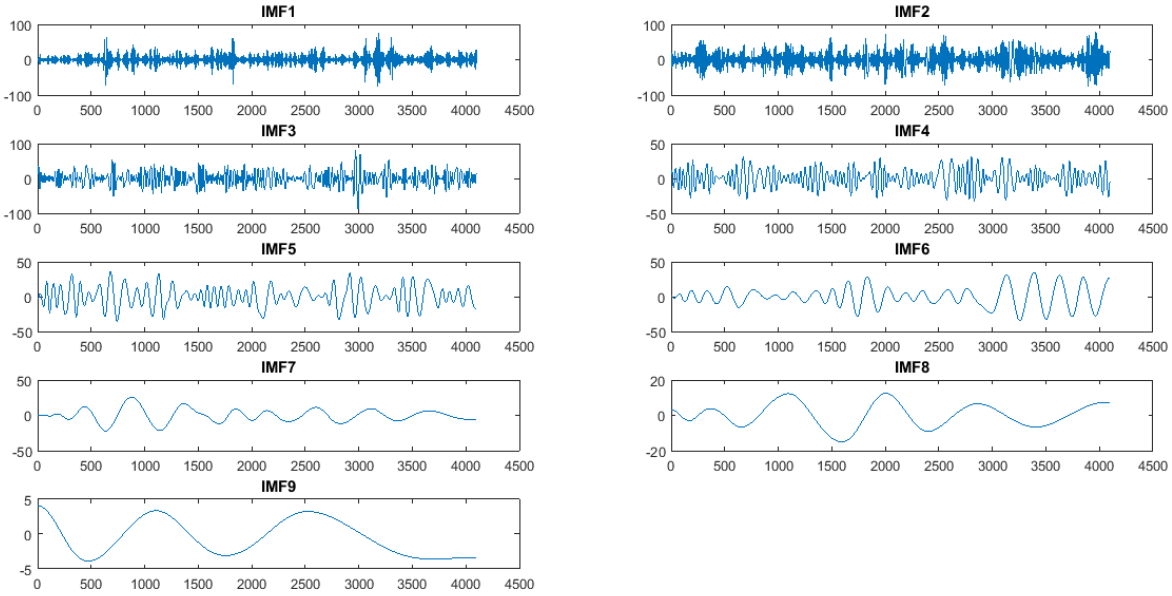


Fig1: IMF's of normal EEG signal

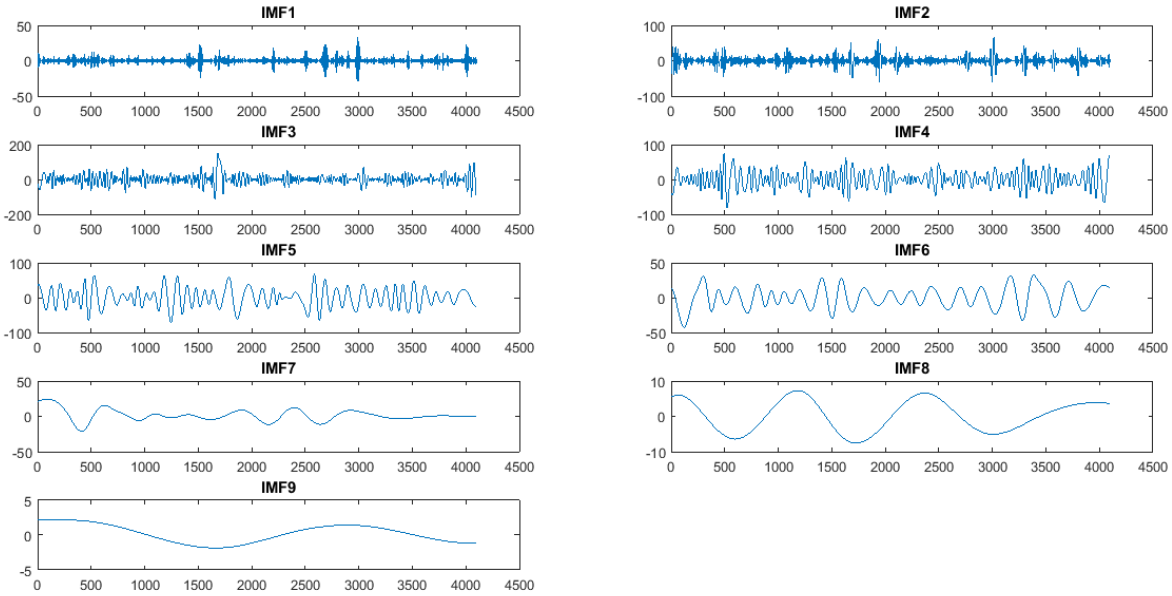


Fig2: IMF's of seizure-free EEG signals

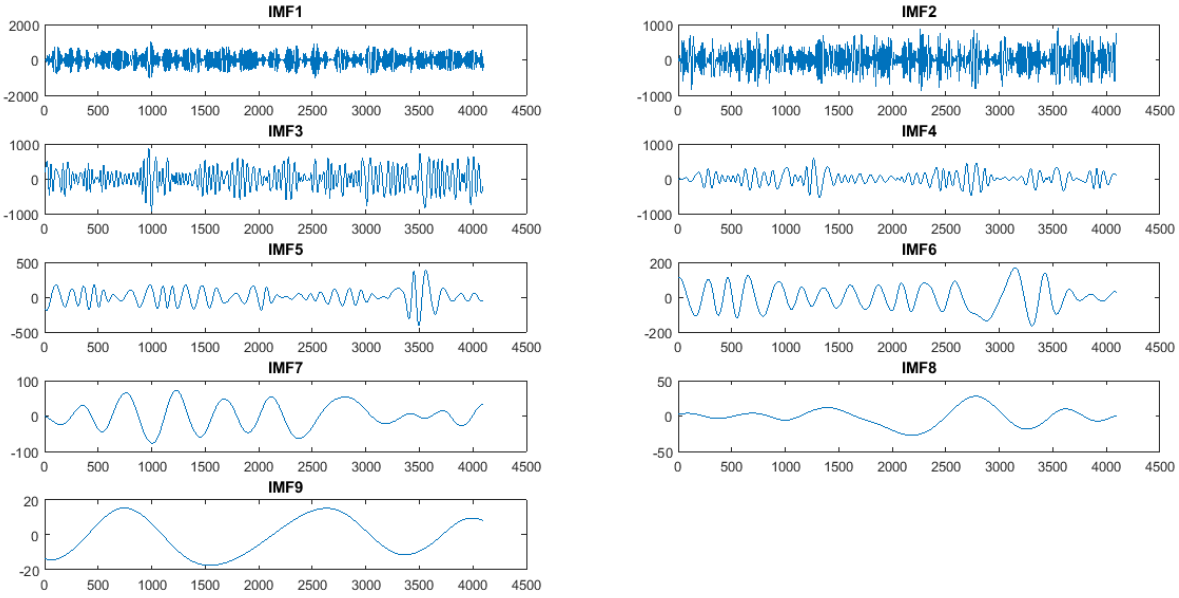
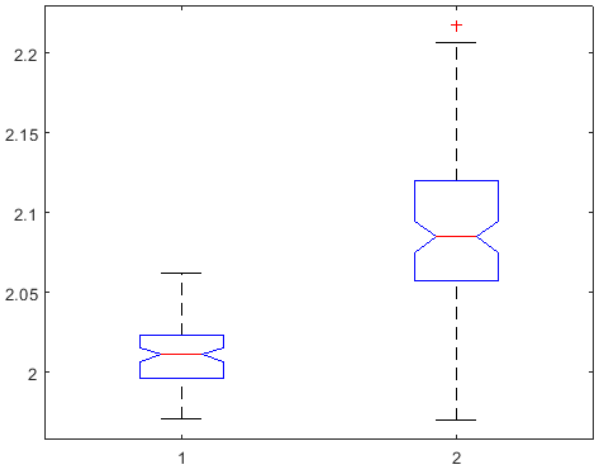


Fig3: IMF's of seizure EEG signal

Using these IMF's, we can predict whether the EEG signal is seizure, seizure-free or normal EEG signal. We predict this by using box plot of two subsets, i.e. by using mean frequencies of each EEG signal in a dataset combined as one using kruskalwallis function.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	344118.1	1	344118.1	102.72	3.85674e-24
Error	322531.9	198	1628.9		
Total	666650	199			



Using this method we can compare the mean frequencies of the datasets and we can compare the normal and seizure signals.

So, the error in this process is very small, so we can easily distinguish between normal, seizure and seizure-free signals.

CHAPTER 4:

SPEECH COMMAND RECOGNITION USING DEEP LEARNING

INTRODUCTION:

This example shows how to train a simple deep learning model that detects the presence of speech commands in audio. The example uses the Speech Commands Dataset [1] to train a convolutional neural network to recognize a given set of commands.

Download the data set
from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz and extract the downloaded file.

Specify the words that you want your model to recognize as commands. Label all words that are not commands as unknown. Labeling words that are not commands as unknowncreates a group of words that approximates the distribution of all words other than the commands. The network uses this group to learn the difference between commands and all other words.

To reduce the class imbalance between the known and unknown words and speed up processing, only include a fraction includeFraction of the unknown words in the training set. Do not include the longer files with background noise in the training set yet. Background noise will be added in a separate step later.

The data set folder contains text files, which list the audio files to be used as the validation and test sets. These predefined validation and test sets do not contain utterances of the same word by the same person, so it is better to use these predefined sets than to select a random subset of the whole data set. To prepare the data for efficient training of a convolutional neural network, convert the speech waveforms to log-mel spectrograms.

Define the parameters of the spectrogram calculation. Segment Duration is the duration of each speech clip (in seconds). Frame Duration is the duration of each frame for spectrogram calculation. Hop Duration is the time step between each column of the spectrogram. Num Bands is the number of log- mel filters and equals the height of each spectrogram.

MY WORK:

I have implemented the deep learning algorithm and got about 95% accuracy on testing and 98% accuracy on training.

The implemented codes are provided with the outputs in the next page. Please refer to the outputs and conclusions for better understanding of the program.

MATLAB CODES USED:

- SPLITDATA FUNCTION:

```
function [adsTrain,adsValidation,adsTest] = splitData(ads,datafolder)
c = fileread(fullfile(datafolder,'validation_list.txt'));
filesValidation = string(split(c));
filesValidation  = filesValidation(filesValidation ~= "");
c = fileread(fullfile(datafolder,'testing_list.txt'));
filesTest = string(split(c));
filesTest  = filesTest(filesTest ~= "");
files = ads.Files;
sf    = split(files,filesep);
isValidation = ismember(sf(:,end-1) + "/" + sf(:,end),filesValidation);
isTest       = ismember(sf(:,end-1) + "/" + sf(:,end),filesTest);

adsTest = subset(ads,isTest);
adsValidation = subset(ads,isValidation);
adsTrain = subset(ads,~isValidation & ~isTest);
end
```

- SPEECHSPECTROGRAMS FUNCTION:

```
function X =
speechSpectrograms(ads,segmentDuration,frameDuration,hopDuration,numBands)

disp("Computing speech spectrograms...");

numHops = ceil((segmentDuration - frameDuration)/hopDuration);
numFiles = length(ads.Files);
X = zeros([numBands,numHops,1,numFiles],'single');

for i = 1:numFiles

    [x,info] = read(ads);
    fs = info.SampleRate;
    frameLength = round(frameDuration*fs);
    hopLength = round(hopDuration*fs);

    spec = melSpectrogram(x,fs, ...
        'WindowLength',frameLength, ...
        'OverlapLength',frameLength - hopLength, ...
        'FFTLength',512, ...
        'NumBands',numBands, ...
        'FrequencyRange',[50,7000]);

    w = size(spec,2);
    left = floor((numHops-w)/2)+1;
    ind = left:left+w-1;
    X(:,ind,1,i) = spec;

    if mod(i,1000) == 0
        disp("Processed " + i + " files out of " + numFiles)
    end
end

disp("...done");

end
```

- BACKGROUNDSPECTROGRAMS FUNCTION:

```
function Xbkg =
backgroundSpectrograms(ads,numBkgClips,volumeRange,segmentDuration,frameDuration,ho
pDuration,numBands)

disp("Computing background spectrograms...");

logVolumeRange = log10(volumeRange);

numBkgFiles = numel(ads.Files);
numClipsPerFile = histcounts(1:numBkgClips,linspace(1,numBkgClips,numBkgFiles+1));

numHops = segmentDuration/hopDuration - 2;
Xbkg = zeros(numBands,numHops,1,numBkgClips,'single');

ind = 1;
for count = 1:numBkgFiles
    [wave,info] = read(ads);

    fs = info.SampleRate;
    frameLength = frameDuration*fs;
    hopLength = hopDuration*fs;

    for j = 1:numClipsPerFile(count)
        indStart =  randi(numel(wave)-fs);
        logVolume = logVolumeRange(1) + diff(logVolumeRange)*rand;
        volume = 10^logVolume;
        x = wave(indStart:indStart+fs-1)*volume;
        x = max(min(x,1),-1);

        Xbkg(:,:,:,ind) = melSpectrogram(x,fs, ...
            'WindowLength',frameLength, ...
            'OverlapLength',frameLength - hopLength, ...
            'FFTLength',512, ...
            'NumBands',numBands, ...
            'FrequencyRange',[50,7000]);

        if mod(ind,1000)==0
            disp("Processed " + string(ind) + " background clips out of " +
string(numBkgClips))
        end
        ind = ind + 1;
    end
end

disp("...done");

end
```

- MAIN FILE:

```
datafolder = fullfile('E:\SURYA-PRO\speech command\speech_commands_v0.01');
ads =
audioDatastore(datafolder,'IncludeSubfolders',true,'FileExtensions','.wav','LabelSo
urce','foldernames')
ads0 = copy(ads);

commands =
categorical(["yes","no","up","down","left","right","on","off","stop","go"]);

isCommand = ismember(ads.Labels,commands);
isUnknown = ~ismember(ads.Labels,[commands,"_background_noise_"]);

includeFraction = 0.2;
mask = rand(numel(ads.Labels),1) < includeFraction;
isUnknown = isUnknown & mask;
ads.Labels(isUnknown) = categorical("unknown");

ads = subset(ads,isCommand|isUnknown);
countEachLabel(ads)

[adsTrain,adsValidation,adsTest] = splitData(ads,datafolder);

segmentDuration = 1;
frameDuration = 0.025;
hopDuration = 0.010;
numBands = 40;

epsil = 1e-6;

XTrain =
speechSpectrograms(adsTrain,segmentDuration,frameDuration,hopDuration,numBands);
XTrain = log10(XTrain + epsil);

XValidation =
speechSpectrograms(adsValidation,segmentDuration,frameDuration,hopDuration,numBands
);
XValidation = log10(XValidation + epsil);

XTest =
speechSpectrograms(adsTest,segmentDuration,frameDuration,hopDuration,numBands);
XTest = log10(XTest + epsil);

YTrain = adsTrain.Labels;
YValidation = adsValidation.Labels;
YTest = adsTest.Labels;

specMin = min(XTrain(:));
specMax = max(XTrain(:));
idx = randperm(size(XTrain,4),3);
figure('Units','normalized','Position',[0.2 0.2 0.6 0.6]);
for i = 1:3
    [x,fs] = audioread(adsTrain.Files{idx(i)});
    subplot(2,3,i)
    plot(x)
    axis tight
    title(string(adsTrain.Labels(idx(i))))
```

```
    subplot(2,3,i+3)
    spect = XTrain(:,:,1,idx(i));
    pcolor(spect)
    caxis([specMin+2 specMax])
    shading flat

    sound(x,fs)
    pause(2)
end

figure
histogram(XTrain,'EdgeColor','none','Normalization','pdf')
axis tight
ax = gca;
ax.YScale = 'log';
xlabel("Input Pixel Value")
ylabel("Probability Density")

adsBkg = subset(ads0,ads0.Labels=="_background_noise_");
numBkgClips = 4000;
volumeRange = [1e-4,1];

XBkg =
backgroundSpectrograms(adsBkg,numBkgClips,volumeRange,segmentDuration,frameDuration
,hopDuration,numBands);
XBkg = log10(XBkg + epsil);

numTrainBkg = floor(0.8*numBkgClips);
numValidationBkg = floor(0.1*numBkgClips);
numTestBkg = floor(0.1*numBkgClips);

XTrain(:,:,:,end+1:end+numTrainBkg) = XBkg(:,:,:,1:numTrainBkg);
XBkg(:,:,:,1:numTrainBkg) = [];
YTrain(end+1:end+numTrainBkg) = "background";

XValidation(:,:,:,end+1:end+numValidationBkg) = XBkg(:,:,:,1:numValidationBkg);
XBkg(:,:,:,1:numValidationBkg) = [];
YValidation(end+1:end+numValidationBkg) = "background";

XTest(:,:,:,end+1:end+numTestBkg) = XBkg(:,:,:,1: numTestBkg);
clear XBkg;
YTest(end+1:end+numTestBkg) = "background";

YTrain = removecats(YTrain);
YValidation = removecats(YValidation);
YTest = removecats(YTest);

figure('Units','normalized','Position',[0.2 0.2 0.5 0.5]);
subplot(2,1,1)
histogram(YTrain)
title("Training Label Distribution")
subplot(2,1,2)
histogram(YValidation)
title("Validation Label Distribution")

sz = size(XTrain);
specSize = sz(1:2);
imageSize = [specSize 1];
augmenter = imageDataAugmenter( ...
    'RandXTranslation',[-10 10], ...
```

```
    'RandXScale',[0.8 1.2], ...
    'FillValue',log10(epsil));
augimdsTrain = augmentedImageDatastore(imageSize,XTrain,YTrain, ...
    'DataAugmentation',augmenter);

classWeights = 1./countcats(YTrain);
classWeights = classWeights'/mean(classWeights);
numClasses = numel(categories(YTrain));

timePoolSize = ceil(imageSize(2)/8);
dropoutProb = 0.2;
numF = 12;
layers = [
    imageInputLayer(imageSize)

    convolution2dLayer(3,numF,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,2*numF,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,4*numF,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(3,'Stride',2,'Padding','same')

    convolution2dLayer(3,4*numF,'Padding','same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(3,4*numF,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer([1 timePoolSize])

    dropoutLayer(dropoutProb)
    fullyConnectedLayer(numClasses)
    softmaxLayer
    weightedClassificationLayer(classWeights)];

miniBatchSize = 128;
validationFrequency = floor(numel(YTrain)/miniBatchSize);
options = trainingOptions('adam', ...
    'InitialLearnRate',3e-4, ...
    'MaxEpochs',25, ...
    'MiniBatchSize',miniBatchSize, ...
    'Shuffle','every-epoch', ...
    'Plots','training-progress', ...
    'Verbose',false, ...
    'ValidationData',{XValidation,YValidation}, ...
    'ValidationFrequency',validationFrequency, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20);
```

```
doTraining = true;
if doTraining
    trainedNet = trainNetwork(augimdsTrain,layers,options);
else
    load('commandNet.mat','trainedNet');
end

YValPred = classify(trainedNet,XValidation);
validationError = mean(YValPred ~= YValidation);
YTrainPred = classify(trainedNet,XTrain);
trainError = mean(YTrainPred ~= YTrain);
disp("Training error: " + trainError*100 + "%")
disp("Validation error: " + validationError*100 + "%")

figure('Units','normalized','Position',[0.2 0.2 0.5 0.5]);
cm = confusionchart(YValidation,YValPred);
cm.Title = 'Confusion Matrix for Validation Data';
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';
sortClasses(cm, [commands,"unknown","background"])

info = whos('trainedNet');
disp("Network size: " + info.bytes/1024 + " kB")

for i=1:100
    x = randn(imageSize);
    tic
    [YPredicted,probs] = classify(trainedNet,x,"ExecutionEnvironment",'cpu');
    time(i) = toc;
end
disp("Single-image prediction time on CPU: " + mean(time(11:end))*1000 + " ms")

fs = 16e3;
classificationRate = 20;
audioIn = audioDeviceReader('SampleRate',fs, ...
    'SamplesPerFrame',floor(fs/classificationRate));

frameLength = floor(frameDuration*fs);
hopLength = floor(hopDuration*fs);
waveBuffer = zeros([fs,1]);

labels = trainedNet.Layers(end).Classes;
YBuffer(1:classificationRate/2) = categorical("background");
probBuffer = zeros([numel(labels),classificationRate/2]);

h = figure('Units','normalized','Position',[0.2 0.1 0.6 0.8]);
```

OUTPUTS:

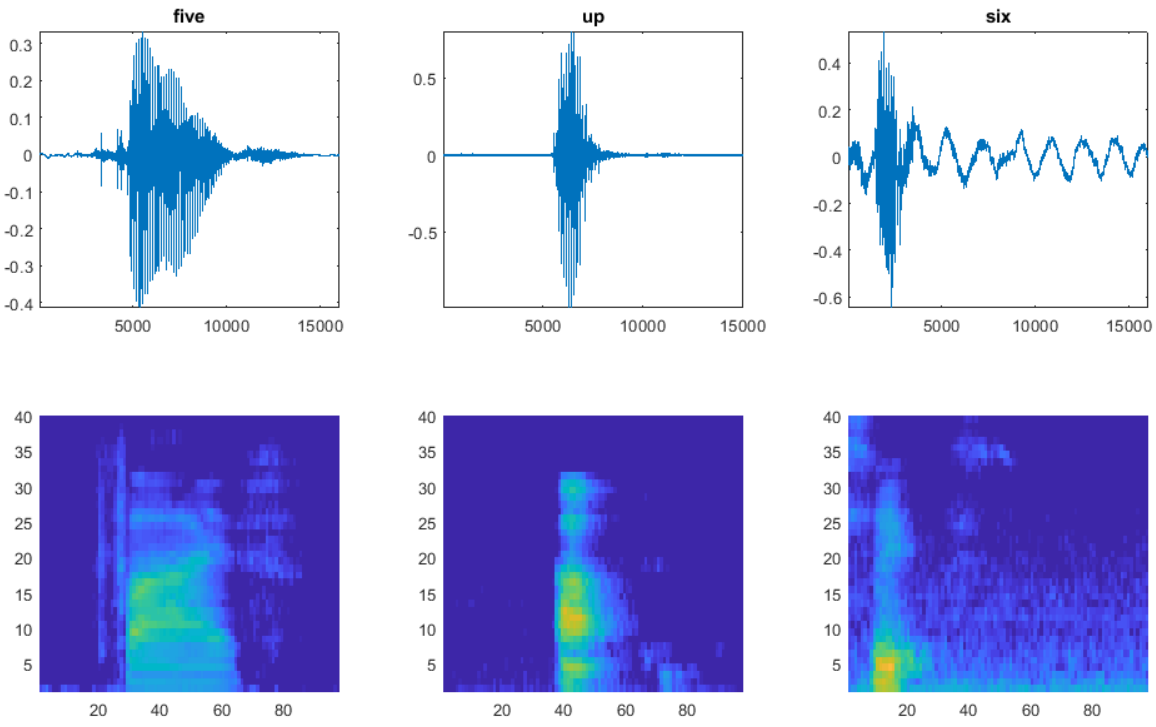


Fig1: Few Inputs with spectrograms

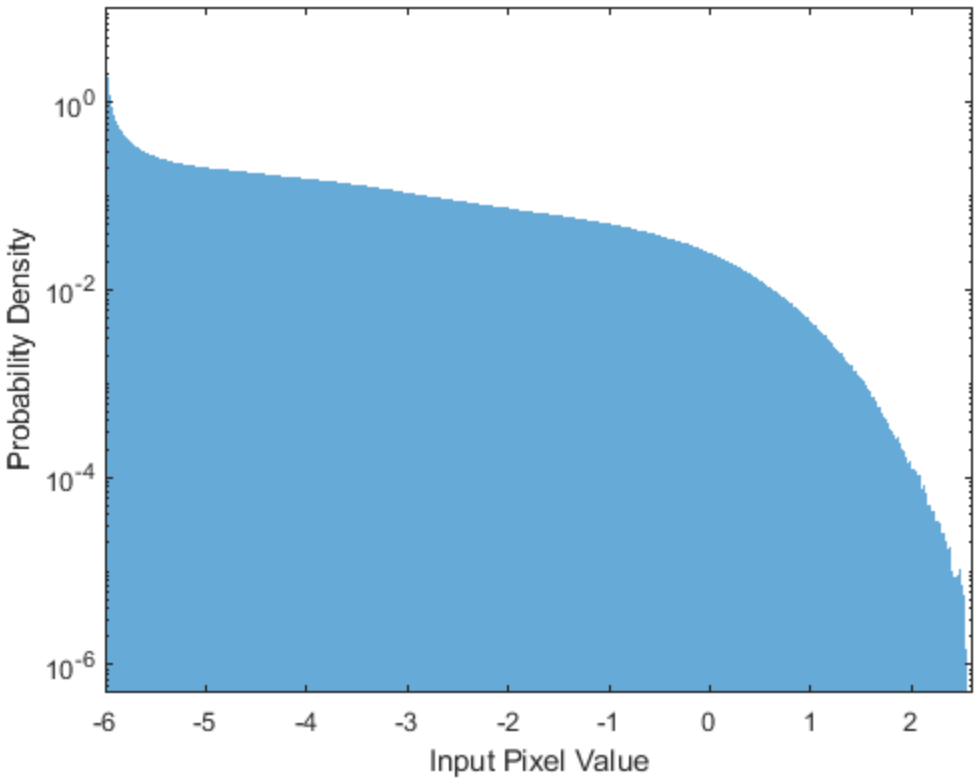


Fig2: Power Spectral Density of the Input commands

Fig3: Distributions of training and testing sets

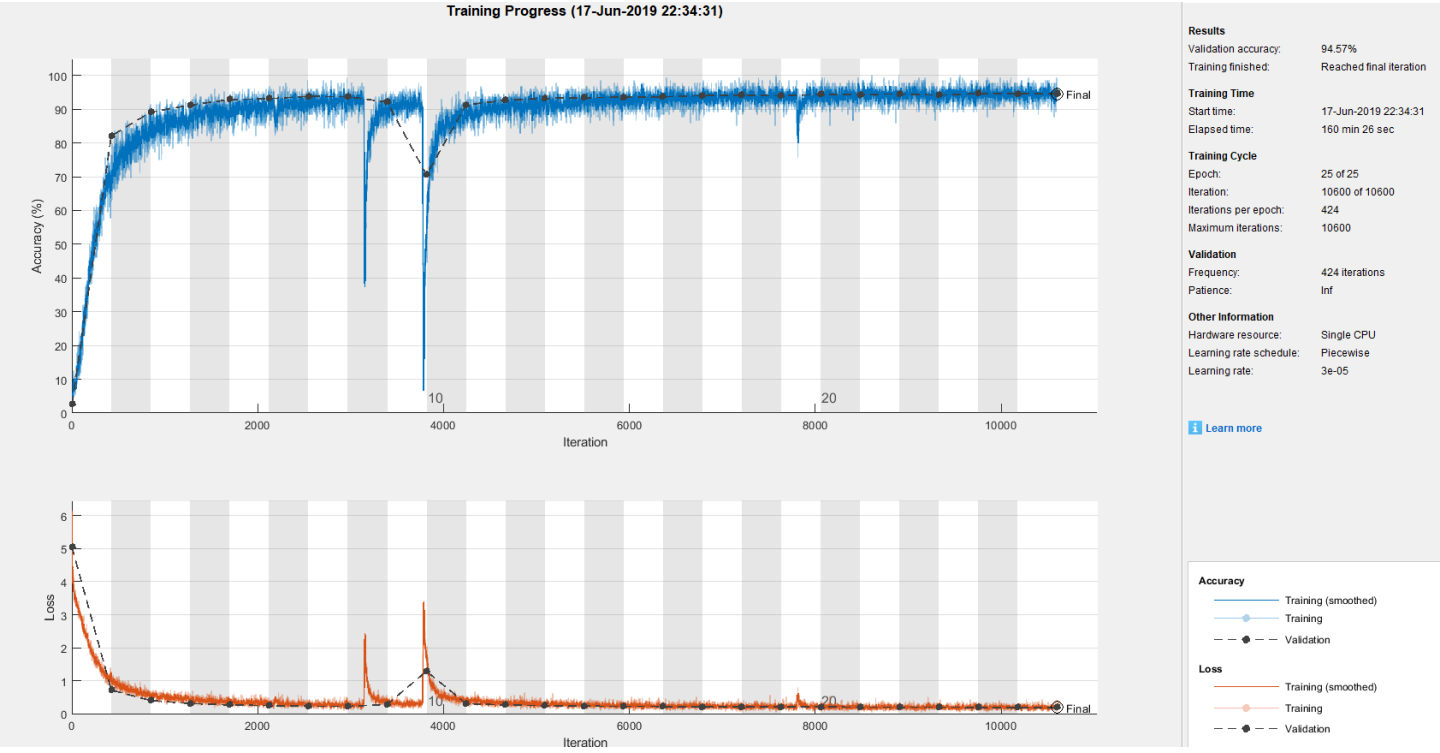


Fig4: The accuracy and loss distributions in train set

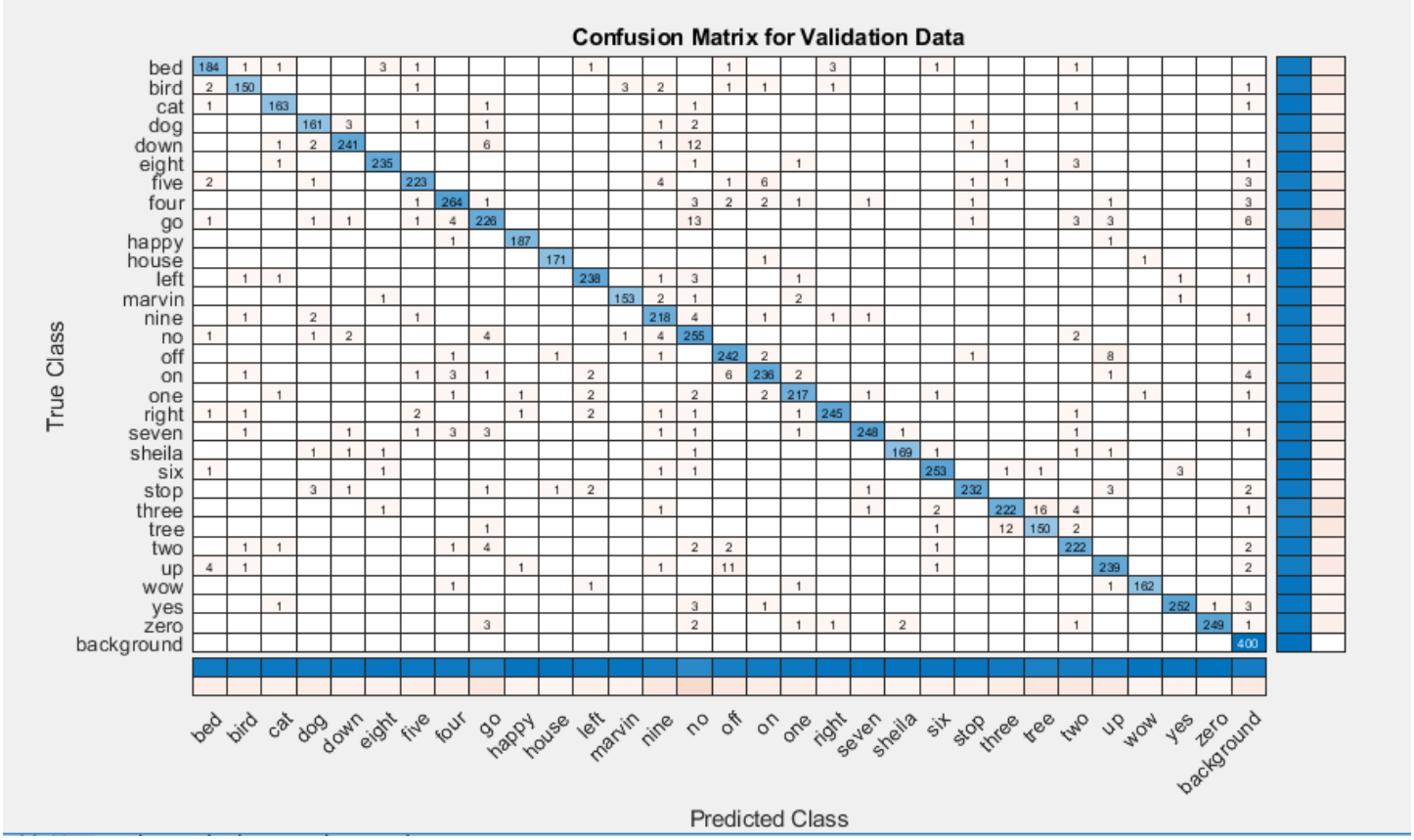


Fig5: The prediction chart of all the input command sets

CONCLUSIONS:

Using the dataset of around 60,000 one word commands set is being used with a variety of 21 subsets. So, we used deep learning to analyze the database and predict any further audio signals for verifications.

So, this is the codes used for this chapter requires MATLAB which is released after R2017b.

Deep learning is used to predict the one word commands which are in the training set and have an accuracy of more than 95%.

CHAPTER 5:

GENDER RECOGNITION USING SPEECH SIGNAL

INTRODUCTION:

In this chapter, we used a Bidirectional Long Short-Term Memory (BiLSTM) network.

Gender classification based on speech signals is an essential component of many audio systems, such as automatic speech recognition, speaker recognition and content-based multimedia indexing. This example uses long short-term memory networks, a type of recurrent neural network well-suited to study sequence and time-series data.

MY WORK:

I have taken dataset from Mozilla dataset for English from the site https://voice.mozilla.org/en. The dataset is having a total of 8, 95,795 inputs and is divided into 3 parts, training, testing and validation in the ration 60:20:20.

The dataset is then defined as either male or female and divided into age groups. This example is using deep learning and we are training the machine to test and validate the other inputs into the required outputs.

I have implemented this on the given dataset above and got an accuracy of 98% in training set. This is the same for testing set also.

So, accuracy greater than 90% is said to be a good program. This program is giving the gender and age range of the person speaking the one line sentence. The codes and the outputs are given from the next page. See the outputs and the codes for better understanding the feature extractions and implementation to fin the age of the person with the gender classification as well.

This is the project to help me check my assumptions in the first chapter, i.e. Spectrogram analysis of the speech signal. In which I have concluded the gender using the frequency spectrum and spectrogram. I wanted to check whether my conclusions were up to the mark, I have implemented this program to get the correct output.

MATLAB CODES USED:

- HELPERGETFEATURESEQUENCES FUNCTION:

```
function featureSequence = HelperGetFeatureSequences(segments,fParams,sParams)
featureSequence = zeros(sParams.SequenceLength,sParams.NumFeatures,0);
for indexS = 1:numel(segments)
    audioIn = segments{indexS};
    audioIn(isnan(audioIn)) = 0;
    [coeffs,delta,deltaDelta] = gtcc(audioIn,fParams.SampleRate, ...
                                     "WindowLength",numel(fParams.Window), ...
                                     "OverlapLength",fParams.OverlapLength, ...
                                     "NumCoeffs",12, ...
                                     "FilterDomain","Time");

    f0 = pitch(audioIn,fParams.SampleRate, ...
               "WindowLength",numel(fParams.Window), ...
               "OverlapLength",fParams.OverlapLength);

    hr = harmonicRatio(audioIn,fParams.SampleRate, ...
                       "Window",fParams.Window, ...
                       "OverlapLength",fParams.OverlapLength);
    [~,frequencyVector,~,S] = spectrogram(audioIn,fParams.Window, ...
        fParams.OverlapLength, numel(fParams.Window), fParams.SampleRate , ...
        "power" , "onesided");

    slope             = spectralSlope(S,frequencyVector);
    [skewness,spread] = spectralSkewness(S,frequencyVector);
    flux              = spectralFlux(S,frequencyVector);
    centroid          = spectralCentroid(S,frequencyVector);
    rolloff           = spectralRolloffPoint(S,frequencyVector);
    decrease          = spectralDecrease(S,frequencyVector);
    flatness          = spectralFlatness(S,frequencyVector);
    kurtosis          = spectralKurtosis(S,frequencyVector);
    features = [coeffs, delta, deltaDelta, f0, hr, slope, skewness, flux, ...
                spread, centroid, rolloff, decrease, flatness, kurtosis];
    features(~isfinite(features)) = 0;
    for index = 1: sParams.HopLength : size(features,1)-sParams.SequenceLength + 1
        F = features(index:index+sParams.SequenceLength - 1,:);
        featureSequence(:,:,end+1) = F;         %#ok
    end
end
```

- HELPERSEGMENTSPEECH FUNCTION:

```
function segments = HelperSegmentSpeech(audio,Fs)
audio               = audio ./ max(abs(audio)); % Normalize amplitude
audio(isnan(audio)) = 0;
WindowLength        = 50e-3 * Fs;
segments            = buffer(audio,WindowLength);
win          = hamming(WindowLength,'periodic');
signalEnergy = sum(segments.^2,1)/WindowLength;
centroid     = spectralCentroid(segments,Fs,'Window',win,'OverlapLength',0);

E = signalEnergy;
C = centroid;
T_E = mean(E)/2;
T_C = 5000;
isSpeechRegion = (E>=T_E) & (C<=T_C);
regionStartPos = find(diff([isSpeechRegion(1)-1, isSpeechRegion]));
```

```
regionLengths  = diff([regionStartPos, numel(isSpeechRegion)+1]);
isSpeechRegion  = isSpeechRegion(regionStartPos) == 1;
regionStartPos  = regionStartPos(isSpeechRegion);
regionLengths   = regionLengths(isSpeechRegion);
extension = 5;
startIndices = zeros(1,numel(regionLengths));
endIndices   = zeros(1,numel(regionLengths));
for index=1:numel(regionLengths)
   startIndices(index) = max(1, (regionStartPos(index) - extension) * WindowLength
+ 1);
   endIndices(index)   = min(numel(audio), (regionStartPos(index) +
regionLengths(index) + extension) * WindowLength);
end
activeSegment       = 1;
isSegmentsActive    = zeros(1,numel(startIndices));
isSegmentsActive(1) = 1;
for index = 2:numel(startIndices)
    if startIndices(index) <= endIndices(activeSegment)
        if endIndices(index) > endIndices(activeSegment)
           endIndices(activeSegment) =  endIndices(index);
        end
    else
        activeSegment = index;
        isSegmentsActive(index) = 1;
    end
end

if ~isempty(startIndices) &&  ~isempty(endIndices)
    numSegments = sum(isSegmentsActive);
    segments    = cell(1,numSegments);
    limits      = zeros(2,numSegments);
    speechSegmentsIndices  = find(isSegmentsActive);
    for index = 1:length(speechSegmentsIndices)
        segments{index} =
audio(startIndices(speechSegmentsIndices(index)):endIndices(speechSegmentsIndices(i
ndex)));
        limits(:,index) = [startIndices(speechSegmentsIndices(index)) ;
endIndices(speechSegmentsIndices(index))];
    end
else
    segments = {audio};
end
```

- MAIN CODE:

```
datafolder = fullfile('E:\SURYA-PRO\male-female detection\en');
ads =
audioDatastore(datafolder,'IncludeSubfolders',true,'FileExtensions','','LabelSource
','foldernames')
metadata = readtable(fullfile(datafolder,"all.csv"));
head(metadata)
csvFiles   = metadata.client_id;
[~,csvInd] = sort(csvFiles);

gender     = metadata.gender;
age        = metadata.age;

adsFiles   = ads.Files;
[~,adsInd] = sort(adsFiles);
```

```
gender     = gender(csvInd(adsInd));
age        = age(csvInd(adsInd));

ads.Labels = gender;

ads = shuffle(ads);
maleOrfemale = categorical(ads.Labels) == "male" | categorical(ads.Labels) ==
"female";
isAdult      = categorical(age) ~= "" & categorical(age) ~= "teens";
ads          = subset(ads,maleOrfemale & isAdult);

countEachLabel(ads)

ismale   = find(categorical(ads.Labels) == "male");
isfemale = find(categorical(ads.Labels) == "female");
numFilesPerGender = 3000;
ads = subset(ads,[ismale(1:numFilesPerGender) isfemale(1:numFilesPerGender)]);

ads = shuffle(ads);

countEachLabel(ads);

[audio,info] = read(ads);
Fs           = info.SampleRate;

timeVector = (1/Fs) * (0:numel(audio)-1);
figure
plot(timeVector,audio)
ylabel("Amplitude")
xlabel("Time (s)")
title("Sample Audio")
grid on

sound(audio,Fs)

audio        = audio ./ max(abs(audio)); % Normalize amplitude
windowLength = 50e-3 * Fs;
segments     = buffer(audio,windowLength);

win = hann(windowLength,'periodic');
signalEnergy = sum(segments.^2,1)/windowLength;
centroid = spectralCentroid(segments,Fs,'Window',win,'OverlapLength',0);

T_E            = mean(signalEnergy)/2;
T_C            = 5000;
isSpeechRegion = (signalEnergy>=T_E) & (centroid<=T_C);

CC = repmat(centroid,windowLength,1);
CC = CC(:);
EE = repmat(signalEnergy,windowLength,1);
EE = EE(:);
flags2 = repmat(isSpeechRegion,windowLength,1);
flags2 = flags2(:);

figure

subplot(3,1,1)
plot(timeVector, CC(1:numel(audio)), ...
     timeVector, repmat(T_C,1,numel(timeVector)), "LineWidth",2)
xlabel("Time (s)")
```

```
ylabel("Normalized Centroid")
legend("Centroid","Threshold")
title("Spectral Centroid")
grid on

subplot(3,1,2)
plot(timeVector, EE(1:numel(audio)), ...
     timeVector, repmat(T_E,1,numel(timeVector)),"LineWidth",2)
ylabel("Normalized Energy")
legend("Energy","Threshold")
title("Window Energy")
grid on

subplot(3,1,3)
plot(timeVector, audio, ...
     timeVector,flags2(1:numel(audio)),"LineWidth",2)
ylabel("Audio")
legend("Audio","Speech Region")
title("Audio")
grid on
ylim([-1 1.1])

regionStartPos = find(diff([isSpeechRegion(1)-1, isSpeechRegion]));

RegionLengths  = diff([regionStartPos, numel(isSpeechRegion)+1]);

isSpeechRegion = isSpeechRegion(regionStartPos) == 1;
regionStartPos = regionStartPos(isSpeechRegion);
RegionLengths  = RegionLengths(isSpeechRegion);

startIndices = zeros(1,numel(RegionLengths));
endIndices   = zeros(1,numel(RegionLengths));
for index=1:numel(RegionLengths)
   startIndices(index) = max(1, (regionStartPos(index) - 5) * windowLength + 1);
   endIndices(index)   = min(numel(audio), (regionStartPos(index) +
RegionLengths(index) + 5) * windowLength);
end

activeSegment       = 1;
isSegmentsActive    = zeros(1,numel(startIndices));
isSegmentsActive(1) = 1;
for index = 2:numel(startIndices)
    if startIndices(index) <= endIndices(activeSegment)
        if endIndices(index) > endIndices(activeSegment)
           endIndices(activeSegment) =  endIndices(index);
        end
    else
        activeSegment = index;
        isSegmentsActive(index) = 1;
    end
end
numSegments = sum(isSegmentsActive);
segments    = cell(1,numSegments);
limits      = zeros(2,numSegments);
speechSegmentsIndices  = find(isSegmentsActive);
for index = 1:length(speechSegmentsIndices)
    segments{index} = audio(startIndices(speechSegmentsIndices(index)): ...
                            endIndices(speechSegmentsIndices(index)));
    limits(:,index) = [startIndices(speechSegmentsIndices(index)) ; ...
                       endIndices(speechSegmentsIndices(index))];
end
```

```
figure

plot(timeVector,audio)
hold on
myLegend = cell(1,numel(segments)+1);
myLegend{1} = "Original Audio";
for index = 1:numel(segments)
    plot(timeVector(limits(1,index):limits(2,index)),segments{index});
    myLegend{index+1} = sprintf("Output Audio Segment %d",index);
end
xlabel("Time (s)")
ylabel("Audio")
grid on
legend(myLegend)

win = hamming(0.03*Fs,"periodic");
overlapLength = 0.75*numel(win);
featureParams = struct("SampleRate",Fs, ...
                  "Window",win, ...
                  "OverlapLength",overlapLength);

sequenceParams = struct("NumFeatures",50, ...
                  "SequenceLength",40, ...
                  "HopLength",20);

T = tall(ads)
segments = cellfun(@(x)HelperSegmentSpeech(x,Fs),T,"UniformOutput",false);
FeatureSequences =
cellfun(@(x)HelperGetFeatureSequences(x,featureParams,sequenceParams),...
                             segments,"UniformOutput",false);
FeatureSequences = gather(FeatureSequences);
featuresMatrix = cat(3,FeatureSequences{:});

sequencesMeans = zeros(1,sequenceParams.NumFeatures);
sequenceStds   = zeros(1,sequenceParams.NumFeatures);

for index = 1:sequenceParams.NumFeatures
    localFeatures             = featuresMatrix(:,index,:);
    sequencesMeans(index)     = mean(localFeatures(:));
    sequenceStds(index)       = std(localFeatures(:));
    featuresMatrix(:,index,:) = (localFeatures -
sequencesMeans(index))/sequenceStds(index);
end

features = cell(1,size(featuresMatrix,3));
for index = 1:size(featuresMatrix,3)
    features{index} = featuresMatrix(:,:,index).';
end

numSequences = cellfun(@(x)size(x,3), FeatureSequences);
mylabels     = ads.Labels;
gender       = cell(sum(numSequences),1);
count        = 1;
for index1 = 1:numel(numSequences)
    for index2 = 1:numSequences(index1)
        gender{count} = mylabels{index1};
        count = count + 1;
    end
end
```

```
layers = [ ...
    sequenceInputLayer(sequenceParams.NumFeatures)
    bilstmLayer(100,"OutputMode","sequence")
    bilstmLayer(100,"OutputMode","last")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];

options = trainingOptions("adam", ...
    "MaxEpochs",10, ...
    "MiniBatchSize",128, ...
    "Plots","training-progress", ...
    "Verbose",false, ...
    "Shuffle","every-epoch", ...
    "LearnRateSchedule","piecewise", ...
    "LearnRateDropFactor",0.1, ...
    "LearnRateDropPeriod",5);

net = trainNetwork(features,categorical(gender),layers,options);

trainPred = classify(net,features);

figure;
cm = confusionchart(categorical(gender),trainPred,'title','Training Accuracy');
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';

numFiles        = numel(numSequences);
actualGender    = categorical(ads.Labels);
predictedGender = actualGender;
counter         = 1;
for index = 1:numFiles
    predictions = trainPred(counter: counter + numSequences(index) - 1);
    predictedGender(index) = mode(predictions);
    counter = counter + numSequences(index);
end
figure
cm = confusionchart(actualGender,predictedGender,'title','Training Accuracy -
Majority Rule');
cm.ColumnSummary = 'column-normalized';
cm.RowSummary = 'row-normalized';
```

OUTPUTS:

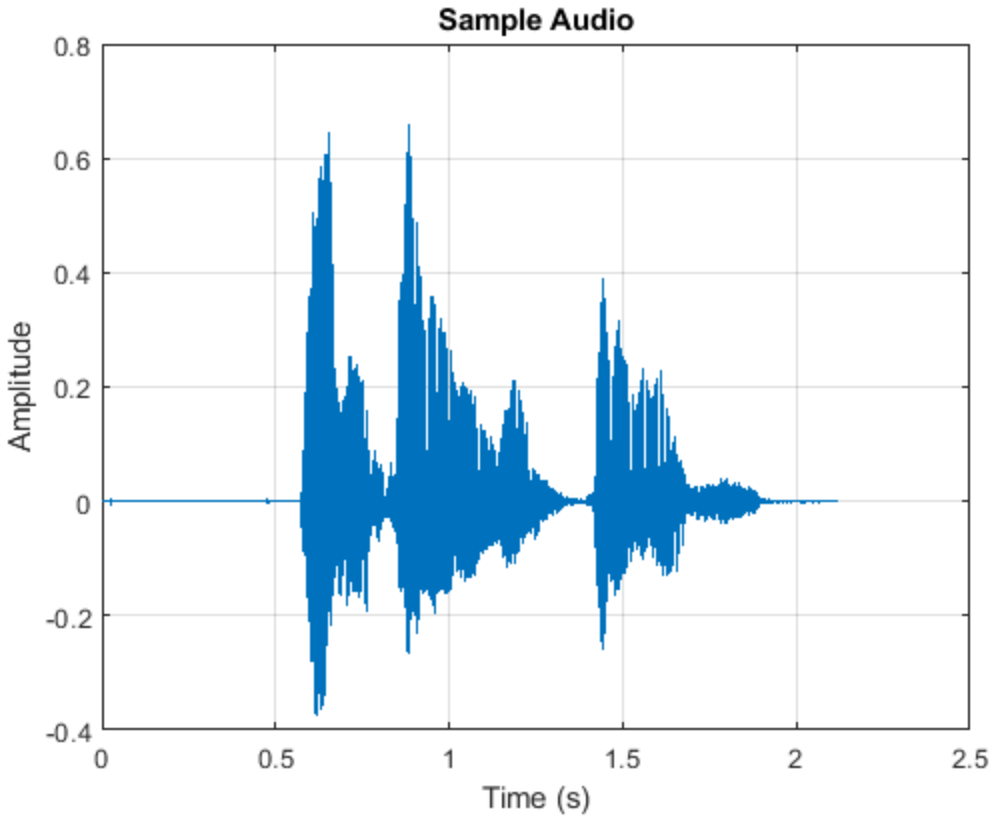


Fig1: Sample Audio

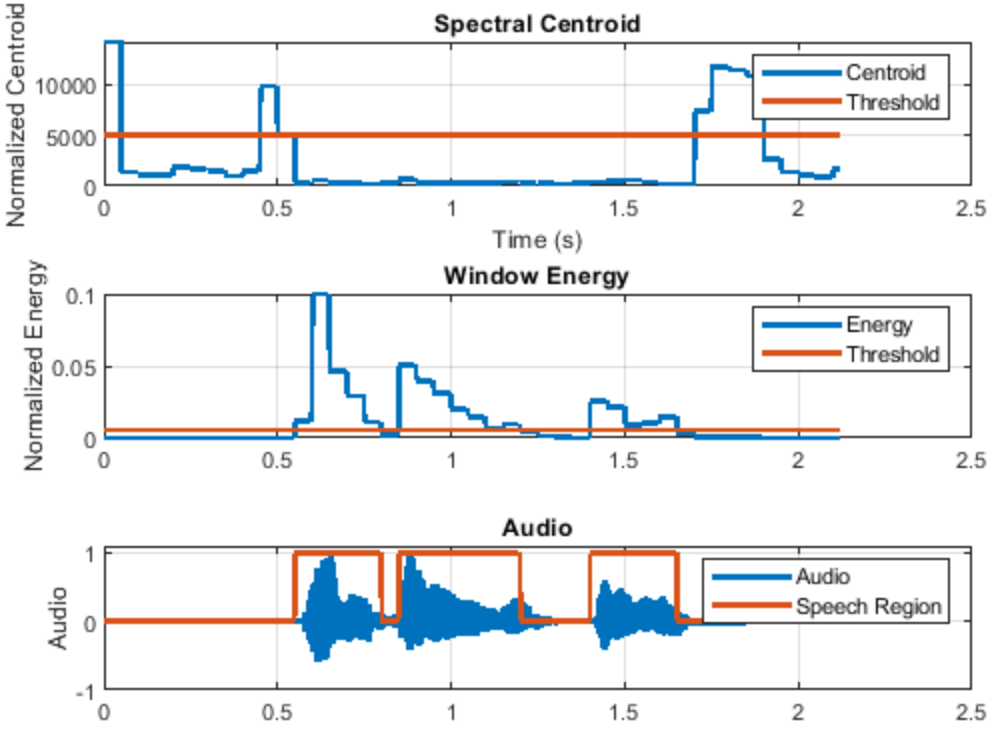


Fig2: Comparison of signal with threshold signal

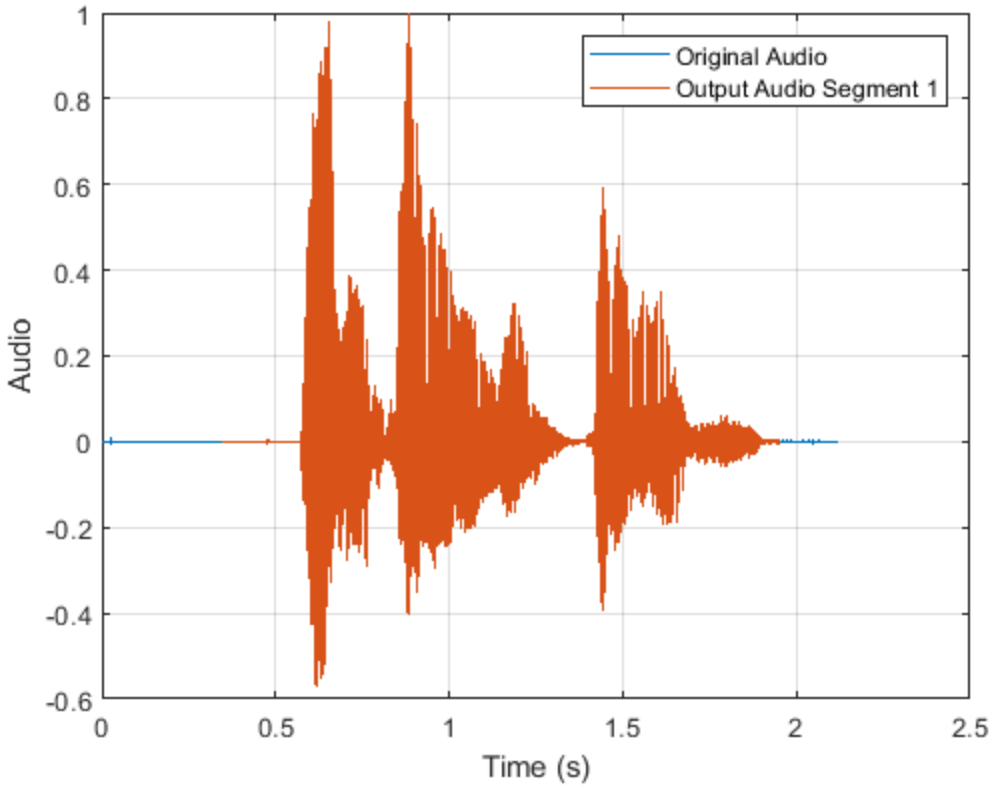


Fig3: Comparison of original and output audio segments

CONLCLUSIONS:

The database is giving 95% efficiency with genders and age classification. The output of the training set is given with sample inputs and outputs. In Fig2, Comparison of Energy with its threshold is shown. Similarly, in the same figure, centroid spectral density is compared with its threshold.

The output efficiency is shown in Fig4, which is showing that the accuracy of the training set is 95%. In Fig5, the percentage error and accuracy of the training set is provided.