Project Report on

REAL TIME SURROUNDING SOUND EVENT DETECTION USING DEEP LEARNING

Submitted by

Suraj.S.Hotkar(64)

Under the guidance of

Dr.S.H. Deshmukh

In partial fulfillment of the award of Bachelor of Technology (Computer Science and Engineering)

Department of Computer Science and Engineering

Maharashtra Institute of Technology,

Aurangabad (Maharashtra)

[2020-2021]

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Aurangabad

Date:

Suraj Hotkar(64)

CERTIFICATE

This is to certify that the project report entitled **"Real Time Surrounding Sound Event Detection Using Deep Learning"**, submitted by **Suraj Hotkar , Nitesh Kathale and Omkar Narsikar** is the bonafied work completed under our supervision and guidance in partial fulfillment for the award of Bachelor of Technology (**Computer Science and Engineering**) of Dr. Babasaheb Ambedkar Marathwada University, Aurangabad (M.S.).

Place: Aurangabad

Date:

Dr.S.H. Deshmukh
Guide

Dr. Smita L. Kasar
Head of Department

Dr. S. P. Bhosle

Principal
Maharashtra Institute of Technology

Aurangabad (M.S.) – 431 005

APPROVAL CERTIFICATE

This Project II report entitled **"Real Time Surrounding Sound Event Detection Using Deep Learning"** by **Suraj Hotkar(64)** is approved for Bachelor of Technology in Computer Science and Engineering, Maharashtra Institute of Technology under Dr. Babasaheb Ambedkar Marathwada University, Aurangabad (M.S.).

Place: Aurangabad

Date:

Examiner: _____

(Signature)

_____

(Name)

CONTENTS

TITLE	PAGE NO
Declaration	
Certificate	
List of Figures	i
List of Graphs	ii
List of Tables	iii
Abstract	iv
1.INTRODUCION	**1**
1.1 Necessity	1
1.2 Problem Definition	1
1.3 Objectives	1
1.4 Scope and Limitations	1
1.5 Applications	1
1.6 Organization and Project Plan	2
2. LITERATURE SURVEY	**4**
2.1 Literature	4
3. SYSTEM DEVELOPMENT	**9**
3.1 Proposed System	9
3.2 Functional Details of Block Diagram	9
3.3 Algorithms /Techniques/ Procedures used	14
3.4 Performance Evaluation Parameter	19
3.5 Implementation Details	19
4. PERFORMANCE ANALYSIS	**30**
4.1 Accuracy/ Testing	30
5. CONCLUSION	**34**
5.1 Conclusion	34
5.2 Future Scope	34
References	
Acknowledgement	
Appendix	

LIST OF FIGURES

Figure	Illustration	Page
1.1	Block Diagram of the Model	9
1.2	MFCC Block Diagram	10
1.3	Levenberg-Marquardt Algorithm Flowchart	17
1.4	Feed Forward Back Propagation Neural Network	21
1.5	GUI Design	30
1.6	Training of Neural Network	31
1.7	Confusion Matrix	33
1.8	System Percentage Accuracy	33

LIST OF GRAPHS

Figure	Illustration	Page
1.1	Distribution of Feedback for String E in Percentage	12
1.2	Distribution of Feedback for String B in Percentage	12
1.3	Distribution of Feedback for String G in Percentage	13
1.4	Distribution of Feedback for String D in Percentage	13
1.5	Distribution of Feedback for String A in Percentage	14
1.6	Distribution of Feedback for String E in Percentage	14
1.7	Perceptron Learning	18
1.8	Training Performance	31
1.9	Neural Network Training State	32
2.0	Neural Network Training Error Histogram	32

LIST OF TABLES

Figure	Illustration	Page
1.1	Audio Feature Extraction Techniques Used By Authors	5
1.2	Audience Feedback for String E	21
1.3	Target of String E	22
1.4	Audience Feedback for String B	23
1.5	Target of String B	23
1.6	Audience Feedback for String G	24
1.7	Target of String G	24
1.8	Audience Feedback for String D	25
1.9	Target of String D	26
2.0	Audience Feedback for String A	26
2.1	Target of String A	27
2.2	Audience Feedback for String E	28
2.3	Target of String E	28

ABSTRACT

We present in this project our work on real-time surrounding audio event detection for indoor events. As all, we know on Day to Day life, Sound is the most important to Humans and it is easy to identify for humans but when we move on computers or any systems which they don't have feelings or they don't know the sounds of different things. Humans also don't know but humans can learn things fastly as compared to systems or computers.

So, our project is based on Real-Time Surrounding Sound event detection using deep learning. We train our system to identify real-time surrounding sounds. Various audio features are used in recent researches in order to achieve automatic audio recognition system using machine learning algorithms. Recognizing audio features can be a challenging problem since it is a continuous task unlike discrete data, image or text recognition. Acquisition and recognition need to be in sync of time frame in order to achieve correct predictions.

The system trained on a standard dataset and we used the feed-forward neural network. In the system, softmax function has been used.

The aim of our project is to record the Real Time Sound and test it using machine learning. After testing this system and evaluating the performance we got 80.00% accuracy.

1. INTRODUCTION

The project topic is Real-time Surrounding Sound Event Detection using deep learning. Real-time means sound recorded in real-time and immediately test it. Now which basis real-time sound test that file this question occurs that's the normal question so, the answer is we trained our system so our system will identify that sound. Now how it happens for that we need some standard dataset, for this system we created our dataset and make it standardized.

Now how we made it standardized dataset. We used the Audacity software for cancelling background noise and set the frequency to 44100Hz. In the dataset, there are five different sounds and those sounds are Clear throat, door slam, drawer, Key drop, door Knock, all these sounds were recorded with the help of mobile and for each sound event, we took 20 samples so we record 100 samples and all these are for only training samples because for the testing we will record real-time sound and give to the system. We have five events, so the one event means one whole sound along with their 20 samples. Now there is the process for training the neural network for that system needs the target and for target system need features from the audio which is in the dataset. For this system mainly we will be using two features extraction techniques those are MFCC and LPCC These two feature extraction techniques are further described in detail.

1.1 Necessity

Normally, Humans are detecting the sounds but sometimes they confuse about two sounds but the system will detect the sound more efficiently and more accurately than Humans.

1.2 Problem Definition

The problem is defined as we record real-time sound and give to our trained system our system will identify the sound. The system will identify only that sound which is present in the dataset.

1.3 Objectives

The objectives of the proposed work are as follows,

1. To obtain set of audio features from an audio signal whereby, the rich set of audio features can be utilized.
2. To analyze psychological dimensions that categorizes audio into special classes.

1.4 Scope and Limitations

The scope of our project is that MFCC is majorly considered as a part of the Extraction technique to identify components of an audio signal and train a model using this feature may increase the accuracy. Also using Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) rather than Feed Forward Back Propagation Neural Network may enhance the performance.

1.5 Applications

1. It can be used in real time based projects for detecting sounds.
2. It can be used for knowing how different surrounding sounds are present in real-time.
3. Useful in entertainment industry

1.6 Organization and Project Plan

Sr. No.	ACTIVITY	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7
1	Project group formation							
2	Project work to be started in respective labs with Literature survey							
3	First review with PPT presentation							
4	Design Use-Case view as per project							
5	Design Block diagram as per project							
6	Second review with PPT presentation							
7	Selection of Technique/Algorithm to be used for project							
8	Third review with PPT presentation							

Sr. No.	ACTIVITY	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13
9	Implementation coding as per project						
10	Testing, Troubleshooting with different techniques						
11	Created Soft copy of project and then final hard copy						

2. LITERATURE SURVEY

2.1 Literature

2.1.1 Time and frequency domain representation

Techniques for the automatic description of music recordings are based on the computation of time and frequency representations of audio signals. We summarize here the main concepts and procedures to obtain such representations.

The frequency of a simple sinusoid is defined as the number of times that a cycle is repeated per second, and it is usually measured in cycles per second, or Hertz (*Hz*). As an example, a sinusoidal wave with a frequency $f = 440$ *Hz* performs 440 cycles per second. The inverse of the frequency f is called the period T ($f = 1/T$), which is measured in seconds and indicates the temporal duration of one oscillation of the sinusoidal signal. In time domain, analog signals $x(t)$ are sampled each Ts seconds to obtain digital signal representations $x[n]$, where $n = i \cdot Ts$, $i = 0, 1, 2, ...$ and $fs = 1$

Ts is the sampling rate in samples per second (*Hz*). According to the Nyquist-Shannon sampling theorem, a given audio signal should be at least sampled to the double of its maximum frequency to avoid the so-called *aliasing*, i.e. the introduction of artifacts during the sampling process.

Time domain signals are suitable to extract descriptors related to the temporal evolution of the waveform $x[n]$, such as the location of major changes in signal properties. The frequency spectrum of a time-domain signal is a representation of that signal in the frequency domain. It can be generated via the Fourier Transform (FT) of the signal, and the resulting values are usually presented as amplitude and phase, both plotted versus frequency. For sampled signals $x[n]$ we use the Discrete version of the Fourier Transform (DFT). Spectrum analysis is usually carried out in short segments of the sound signal (called *frames*), in order to capture the variations in frequency content along time (Short-Time Fourier Transform - STFT). This is mathematically expressed by multiplying the discrete signal $x[n]$ by a window function $w[n]$, which typically has a bell-shaped form and is zero-valued outside of the considered interval [1]

2.1.2 Audio Information

Audio is sound within the acoustic range available to humans. An audio frequency (AF) is an electrical alternating current within the 20 to 20,000 hertz (cycles per second) range that can be used to produce acoustic sound. In computers, audio is the sound system that comes with or can be added to a computer many methods of extracting features from audio signals have been researched for audio content analysis. Mel-frequency cepstral coefficients (MFCCs) are particularly popular for extracting

the power spectrum of an audio signal. The basic process is to take the spectrogram of the audio signal, convert it to a Mel scale ($Mel\ scale$ = 2595 $*$ log10 (1 +$frequency$700)) , take 8 the log of the powers at each Mel- frequency, and apply the discrete cosine transform to generate the Mel-frequency cepstrum (MFC). The MFCCs are the amplitudes of the resulting cepstrum.

This process has many parameters that may be adjusted for results customized to a specific function. MFCCs are most often used as features in speech recognition and analysis. Another, simpler, timbre feature extraction method is the Zero-Crossing Rate, or the rate at which a signal changes from a positive to a negative value and back.

These expertly defined methods of extracting audio information have had many recorded successes, and have improved performance of audio analysis across various problem domains.

However, these methods were not designed for musical data. They were designed for speech audio, which is only a single contribution to a musical track (more in choral/a Capella pieces, and less in instrumental pieces). Despite being designed for speech audio, research conducted with these methods of feature extraction has proved that they do extract some meaningful information from music audio. In fact, many papers have been published using these methods to perform musical analysis and classification tasks.

When analyzing musical audio, the raw audio signal is most often transformed into its spectrogram representation prior to its analysis. A spectrogram is generated by analyzing the existing frequency components of an audio signal over a given frame of time of the signal. These frequencies are then analyzed for their individual magnitudes which are then translated to values on a two dimensional matrix. The dimensions of the spectrogram are the time domain and the frequency domain, and the values of the spectrogram are the magnitudes of the associated frequency at that time [2

2.1.3 Comparison of audio feature extraction techniques used by author

Table 1.1: Audio Feature Extraction Techniques Used By Authors [3]

Author/Year	Feature extraction techniques	Classifiers	Data	Result
Chachada and Kuo(2014)	MFCC Local Discriminant Bases(LDB)	K-Nearest Neighbor(KNN) Support vector Machines(SVM)	37 classes of artificial (instrumental) and natural sounds (Example:Bells, Clapping,Thunder and etc.)	MFCC features have performed than LDB features by using KNN in the domain of artificial and natural sounds
Kumar,Pandya and Jawahar (2014)	MFCC	K-Nearest Neighbor(KNN) Bayes Network (BNs) Support Vector Machines(SVM)	10 Indian ragas(melody) consisting of 170 tunes	In recognition of raagas (melody), KNN and BNs achieved more than 70% of classification accuracy and SVM achieved more than 80% of classification accuracy by using MFCC
Bormane & Dusane (2013)	MFCC LPC ZCR	Wavelet Packet Transform (WPT)	4 classes of musical Instruments (Example: Sitar, Piano and Guitar)	Keyboard Instruments- More than 60% of recognition rate
Dave (2013)	MFCC LPC PLP	Support Vector Machine(SVM) Artificial Neural Network(ANN)	Music and Speech Signals	Compared to MFCC and LPC, PLP is more useful in the domain of speech signals
Rocha , Panda and Piava (2013)	Standard audio features (SA)(spectral features and MFCC) Melodic audio features (MA)	K-Nearest Neighbour(kNN) Bayes Network (BNs) Support Vector Machines(SVM) C4.5 decision tree Naïve Bayes(NB)	903 datasets of emotional music	BNs: 40%-62% NB: 39%-48% KNN: 40%-60% C4.5 decision tree: 34%-60% SVM:45%-64%

2.1.4 Deep Network

The function of Artificial Neural Networks (ANNs) is inspired by the biological neurons in the human brain. The human brain is comprised of billions of neurons, each of which is connected up to 10,000 other neurons. These neurons receive, process, and transmit information necessary for various biological functions, from basic muscle movements to complex organ operations. ANNs attempt to model something similar to this function without being constrained by the real-world interactions of the neurons in the brain. Inputs to the ANN, which includes a bias term, are applied through weighted connections to an activation function, which produces an output value from the ANN.

In a biological neuron, impulses are received via dendrites from across the synapse, or gap, between neuron cells. The impulses are carried by the axon connection to the axon terminals before being transmitted to the next neuron cell. *X* is the input to the neuron, or the 'impulse' received by the neuron. A weighted matrix, *W*, is applied to the input, *X*, before reaching the 'cell body' where the weighted input and bias are evaluated. An activation function, shown in the right of Figure X as *f*, is used to constrain the value of the output, such as the sigmoid function ($(x) =11+e-x$) to constrain the output range to [0, 1] or TANH ($\tanh(x) = 2\sigma(2x) - 1$, a scaled sigmoid) to constrain the output to the range [-1, 1]. The basic structure of an ANN uses multiple layers made up of these artificial neurons. The initial layer is the input layer, where the input to the network is applied. The middle layers are called hidden layers. A neural network can contain any number of hidden layers with any number of artificial neurons per layer. The final layer of the network is the output layer, where the final output of the network is produced [2].

2.1.5 Study of Research paper related to this project

Paper Title, Author, Journal/ conference	Method / Approach used	Dataset used	Advantages	Drawback(s)	Future Scope
Anomaly Detection For Environmental Noise Monitoring BY DUC H. PHAN(2018)	RNN PCA	RealTime Database Used	High Accuracy	RNN does Not provide Best Solution	Neural Network
Sound Event Recognition in Unstructured Environments using Spectrogram Image Processing By Jonathan William Dennis	MFCC-HMM using HTK SER Method(SIF Method	50 Sound Classes	Strong database	Very large training database and SIF	Time Complexity
Automatic Sound Detection And Recognition For Noisy Environment (Published on 2000)	SNR HMM	6 classes 800 samples	Strong Database	Statistical Approach	Accuracy

Literature survey based on various research paper:

1. Anomaly Detection For Environmental Noise Monitoring BY DUC H. PHAN(2018)

During the research found that this system based on support vector machine, replicator neural network, and principal component analysis, typical measurements are approximately generated by a multivariate Gaussian distribution, and anomalies are input samples which are unlikely to be present under the corresponding normative distribution. In this the octave-band sound pressure level is available as raw data Secondly, the non stationary nature of sound levels in residential areas and their daily patterns are illustrated. Thirdly, different anomaly detection algorithms applied to a continuously monitoring sound pressure level data set are reported. Lastly, the extension of robust PCA anomaly detection by introducing piecewise constant mean and covariance parameters in multivariate Gaussian distribution of the generative model produces an anomaly detection that can adapt to the dynamics of residential noise level.

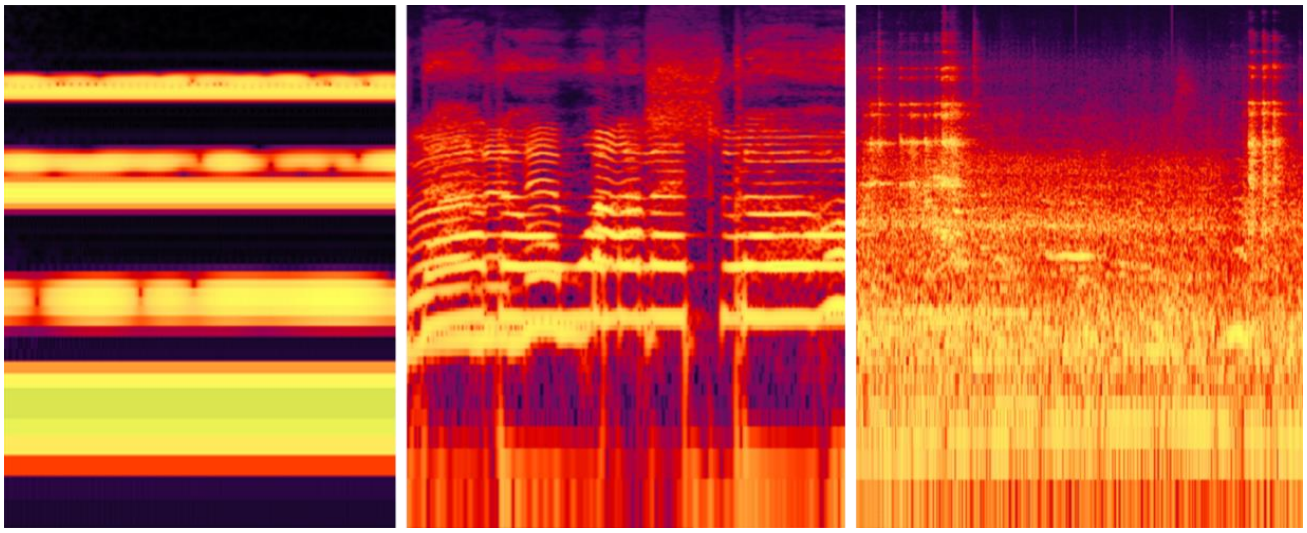
Based on this approach some limitations found. First support vector machine, replicator neural network, and principal component analysis based anomaly detection shows low performance in the collected data because these standard algorithms are unable to exploit the daily patterns. Second for this system need protect speech privacy, the algorithm should only be applied on coarse measures such as noise intensity, octave band, or one-third-octave band measurements over intervals considerably larger than phoneme duration, but not directly to the raw audio. Third the normal and anomaly definitions are time-dependent at a given monitored area, because noise-level patterns change over time .Fourth, if octave bands or one-third-octave bands are used, the algorithm has to work on high-dimensional data. Fifth for this system the cost for Hardware and infrastructure is very high. Sixth the octave band is if you go for higher frequency you don't get much accuracy.

2. Sound Event Recognition in Unstructured Environments using Spectrogram Image Processing By Jonathan William Dennis (2014):

During the research found that the approach taken is to interpret the sound event as a two-dimensional spectrogram image, with the two axes as the time and frequency dimensions. This enables novel methods for SER to be developed based on spectrogram image processing, which are inspired by techniques from the field of image processing.

The challenge in such environments are the adverse effects such as noise, distortion and multiple sources, which are more likely to occur with distant microphones compared to the close-talking microphones that are more common in ASR. In addition, the characteristics of

acoustic events are less well defined than those of speech, and there is no sub-word dictionary available like the phonemes in speech. Therefore, the performance of ASR systems typically degrades dramatically in these challenging unstructured environments, and it is important to develop new methods that can perform well for this challenging task. In this thesis, the approach taken is to interpret the sound event as a two-dimensional spectrogram image, with the two axes as the time and frequency dimensions. The drawbacks of using this approach First there is a wide variation in the time-frequency structure of different sound events, and this information may not be best captured by an two-dimensional spectrogram image. Second Here, two of the basic challenges are the classification of visual scenes and the detection of objects in cluttered images. Third Sounds are “transparent” : One challenge posed in the comparison between visual images and spectrograms is the fact that visual objects and sound events do not accumulate in the same manner. To use a visual analogy, one could say that sounds are always “transparent” [4] whereas most visual objects are opaque. When encountering a pixel of a certain color in an image, it can most often be assumed to belong to a single object. Discrete sound events do not separate into layers on a spectrogram: Instead, they all sum together into a distinct whole. That means that a particular observed frequency in a spectrogram cannot be assumed to belong to a single sound as the magnitude of that frequency could have been produced by any number of accumulated sounds or even by the complex
interactions between sound waves such as phase cancellation. This makes it difficult to separate simultaneous sounds in spectrogram representations



Three examples of difficult scenarios of spectrogram analysis. (Left): Two similar tones cause uneven phase cancellations across frequencies. (Middle): Two simultaneous voices with similar pitch are difficult to tell apart. (Right): Noisy and complex auditory scenes make it particularly

difficult to distinguish sound events

3) Automatic Sound Detection And Recognition For Noisy Environment (Published on 2000):

This paper addresses the problem of automatic detection and recognition of impulsive sounds, such as glass breaks,human screams, gunshots, explosions or door slams. A complete detection and recognition system is described and evaluated on a sound database containing more than 800 signals distributed among six different classes.Emphasis is set on robust techniques,allowing the use of this system in a noisy environment. The detection algorithm, based on a median filter, features a highly robust performance even under important background noise conditions. In the recognition stage, two statistical classifiers are compared, using Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM), respectively. It can be shown that a rather good recognition rate (98% at 70dB and above 80% for 0dB signal-to-noise ratios) can be reached, even under severe gaussian white noise degradations.

The database used in the experiments reported in this paper contains 822 sounds of 6 different classes associated to intrusion or aggression situations : 314 door slams, 88 glass breaks, 73 human screams, 62 explosions, 225 gun shots and 60 other stationary noises. The sounds were taken from different sound libraries (BBC, Warner, Noisex-92 [2]) and there is a lot of variability within a same class (differences of quality, differences in signal length, different signal energy levels, etc). Some of the signals were also manually recorded. All signals were digitized and sampled at 44.1 kHz

During the research found this system use a Gaussian Mixture Model (GMM) to form statistical representations of normal events, thresholding the likelihood of the incoming data based on selected anomalies returned by the GMM. the drawbacks of using this approach for SER are twofold. Firstly, there is a wide variation in the time-frequency structure of different sound events, and this information may not be best captured by an HMM which assumes independence between adjacent observations in time. Secondly, the frame-based features only capture the sound event information within a narrow time window, and commonly represent the full frequency spectrum. This causes problems in mismatched conditions, where the features may contain

elements from noise or multiple sources and it is challenging to separate them. Therefore, while such systems typically have a high recognition accuracy in clean conditi

3. SYSTEM DEVELOPMENT

3.1 Proposed System

The Proposed System has execution by creating database i.e. recording the audio samples.

These audio samples are then preprocessed to make audio in a standard format and particular frequency range .After preprocessing audio, the feature extraction of audio attributes is carried out. For generating target neurons audience are made to listen to the samples resulting into target with highest rated parameter.

To train the neural network the audio features are given input to the input layer of neural network henceforth resulting into firing the highest rated parameter as '1' and others as '0'. For the testing of neural network the target generated by the audience questionnaire and the machine generated target are checked, so the number of correctly identified samples will give the accuracy of the neural network.

The below block diagram shows the flow of execution of the proposed system

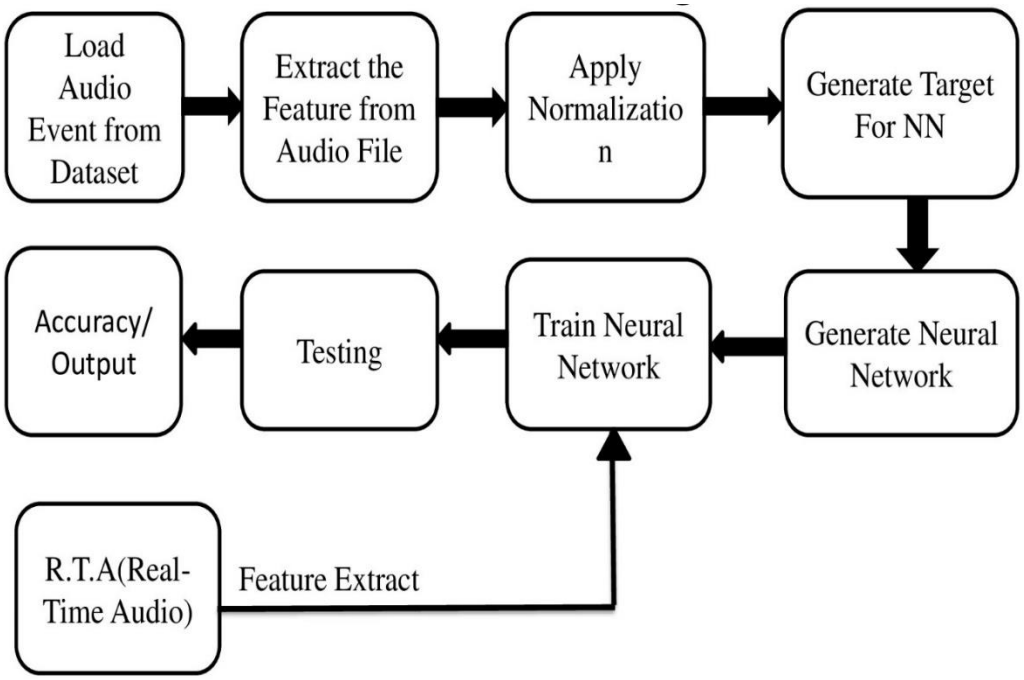


Figure 1.1: Block Diagram of the Model

Firstly, load the dataset after loading the dataset give the number of audio events as five because we took five audio events and total samples per audio event are 20. Then Extract the features from audio file extraction feature is important because the computer knows only numbers so in feature extraction from audio file these feature extraction technique get some random values. There are two types of Feature extraction technique one is MFCC and other is LPCC. These

two also known as audio descriptors. Once the system will get features from an audio file then the system will normalize them. Now, Normalization is a technique in which random values get converted into zeros and ones why is it necessary because the computer only knows binary values so normalization and feature extraction is so important.

Now the system needs the target for a neural network so for a neural network we have to set some target so the neural network gets the right target and the neural network trained in the right way. Once to the system, we will give a target to the system. The system will ready to generate a neural network. When System generate a neural network the system has to train it and the system train that neural network which we give them as the timing of generating neural network. We trained the neural network using the Feed-Forward back propagation neural network. After completion of training, the neural network is ready for testing the files, but the project is based on real-time, so the system will record real-time sound. So after re-record, the sound feature extraction is necessary. After extracting features directly the system give the file to the testing because the neural network was trained.

Functional Details Of Blocks

3.1.1 Audio Database

i. Characteristics attribute

We recorded the sound using a mobile phone. Then using Audacity Software Cancelling Background noise from each file and save in the .Wav Format. Technical characteristics of sound samples are as follows,

Sampling frequency of the audio files in audio database is 44100 Hz. Duration of each audio is around 0.5 minute. Bit rate is 705-768kbps and the audio file format is '.wav' for better sound quality.

In the dataset, we took 5 different sounds those are Clear throat, door slam, drawer, Key Drop, Knocking. Now for each sound event, we took 20 samples so in total we took 100 audio samples. The system will train all samples because for testing we record in real-time audio and then that audio system will check it.

3.1.2 Feature Analysis

i. List of Audio Descriptors

Audio descriptor-

MFCC (Mel-frequency cepstral coefficient).

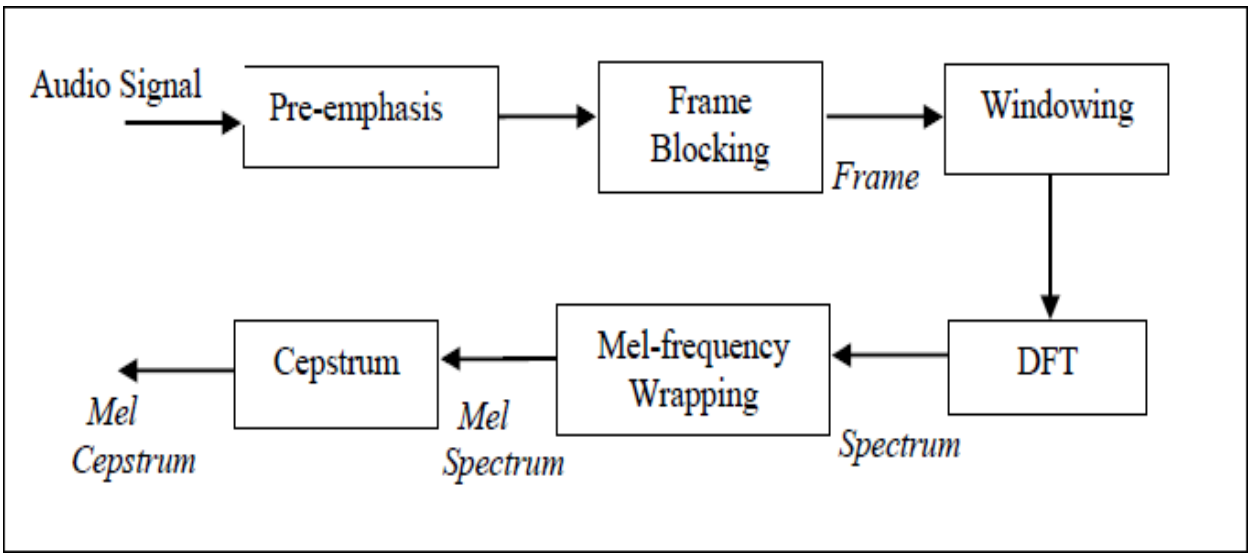


Figure 1.2: MFCC Block Diagram

Figure 1.2 shows the block diagram of MFCC.As shown in the figure, the audio signal is passed into process called pre-emphasis. The function of the pre-emphasis process is to overcome the high frequency part that was suppressed during the sound production. Next frame blocking is done in that the input speech signal is segmented into frames of 15~20 ms with overlap of 50% of the frame size. Overlapping is used to produce continuity within frames. The next step is windowing where each individual frame is then windowed by using Hamming window in order to minimize the signal discontinuities at the beginning and end of each frame.

Then, the signal will be processed by using Discrete Fourier Transform (DFT). DFT convert the sampled function from its original domain (often time or position along a line) to the frequency domain. Therefore, each frame of N samples from the time domain is converted into the frequency domain. The output of DFT is defined in spectrum.

In the next step, the spectrum is passed through a process called mel-frequency wrapping and expressed in the mel-frequency scale. This mel scale is used to approximate the human auditory system's response more closely than linearly-spaced frequency bands.

Cepstrum inverses the Fourier transform of the logarithm of the estimated spectrum of an audio signal. The result is expressed in mel cepstrum and is referred to as the mel-scale cepstral coefficients or MFCC

Linear prediction cepstral coefficients (LPCC)

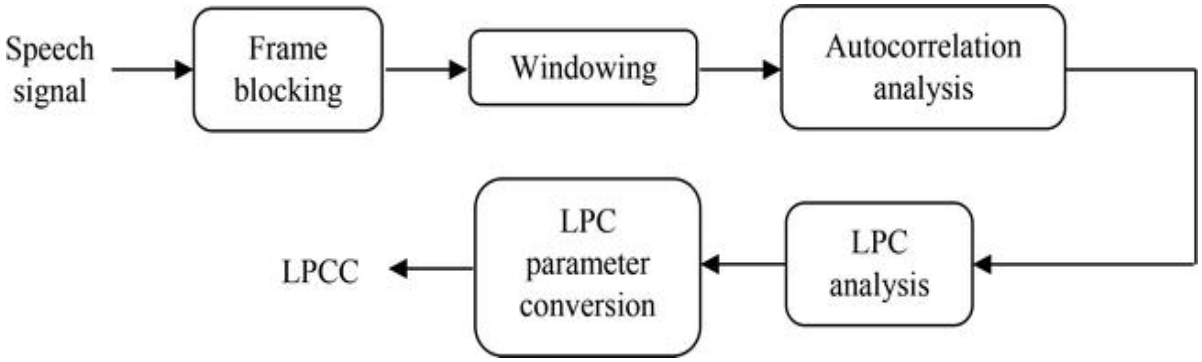


Figure 1.3: LPCC Block Diagram

MFCC and LPCC these are features extraction techniques from audio signals. So these two features extraction technique is very important.

3.2 Details Of Algorithms/Techniques Used

3.2.1 Perceptron Neural Networks

Rosenblatt created many variations of the perceptron. One of the simplest was a single-layer network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. The training technique used is called the perceptron learning rule. The perceptron generated great interest due to its ability to generalize from its training vectors and learn from initially randomly distributed connections. Perceptron is especially suited for simple problems in pattern classification. They are fast and reliable networks for the problems they can solve. In addition, an understanding of the operations of the perceptron provides a good basis for understanding more complex network

Neuron Model

A perceptron neuron, which uses the hard-limit transfer function hardlim, is shown below.

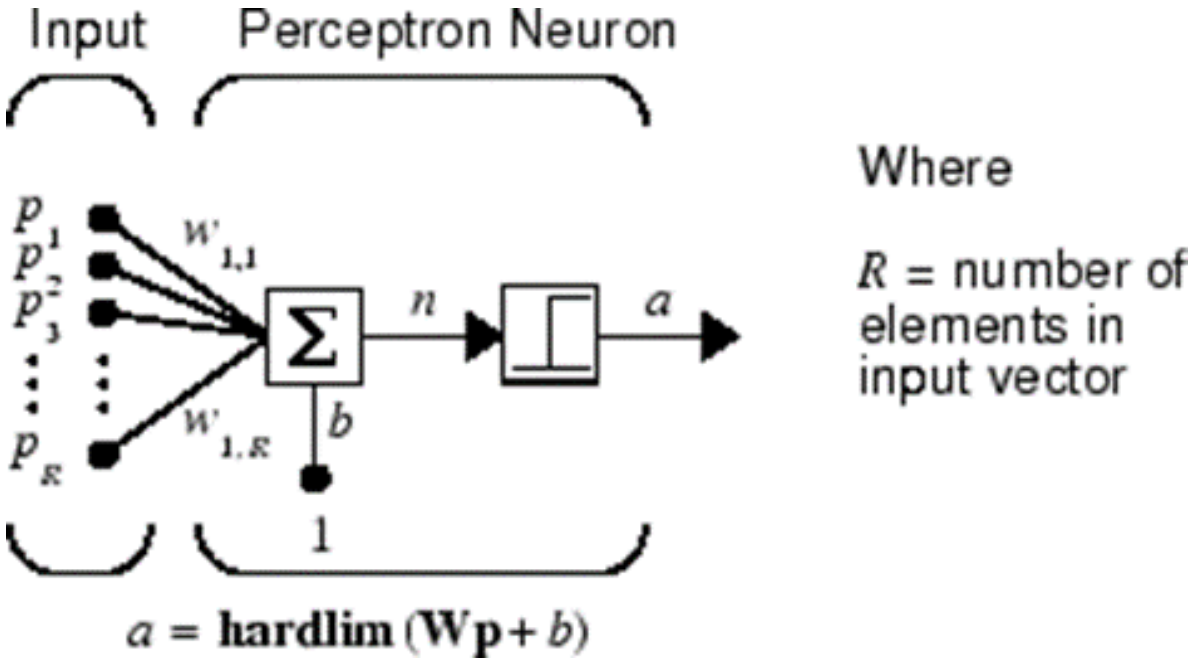


Figure 1.3: Perceptron neuron for hard-limit transfer function

Each external input is weighted with an appropriate weight w1j, and the sum of the weighted inputs is sent to the hard-limit transfer function, which also has an input of 1 transmitted to it through the bias. The hard-limit transfer function, which returns a 0 or a 1, is shown below.

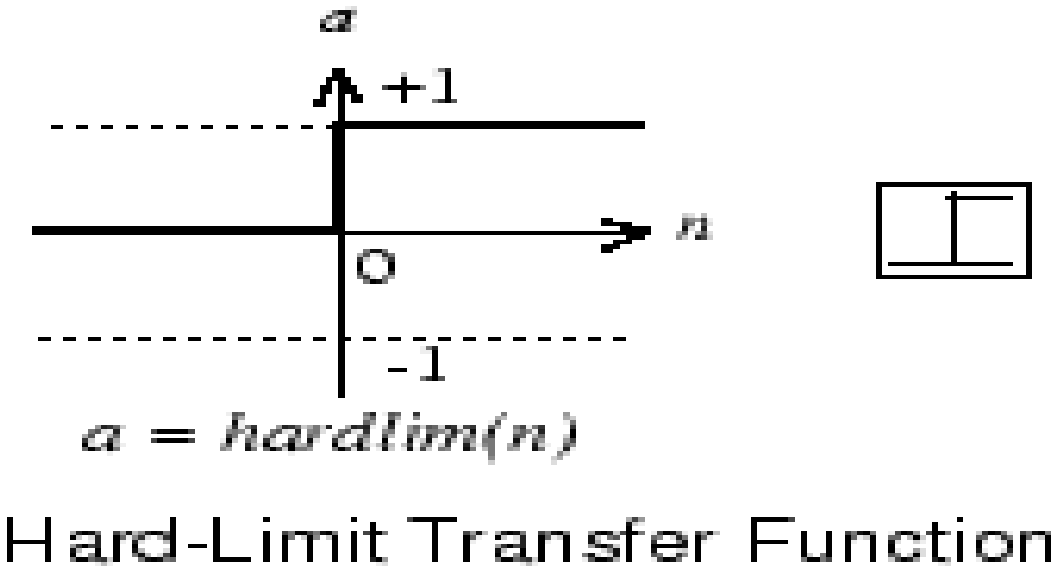


Figure 1.4: hard-limit transfer function

The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0; otherwise it produces a 0.

The hard-limit transfer function gives a perceptron the ability to classify input vectors by dividing the input space into two regions. Specifically, outputs will be 0 if the net input n is less than 0, or 1 if the net input n is 0 or greater. The following figure 1.5 show the input space of a two-input hard limit neuron with the weights w1,1 = −1, w1,2 = 1 and a bias b = 1.

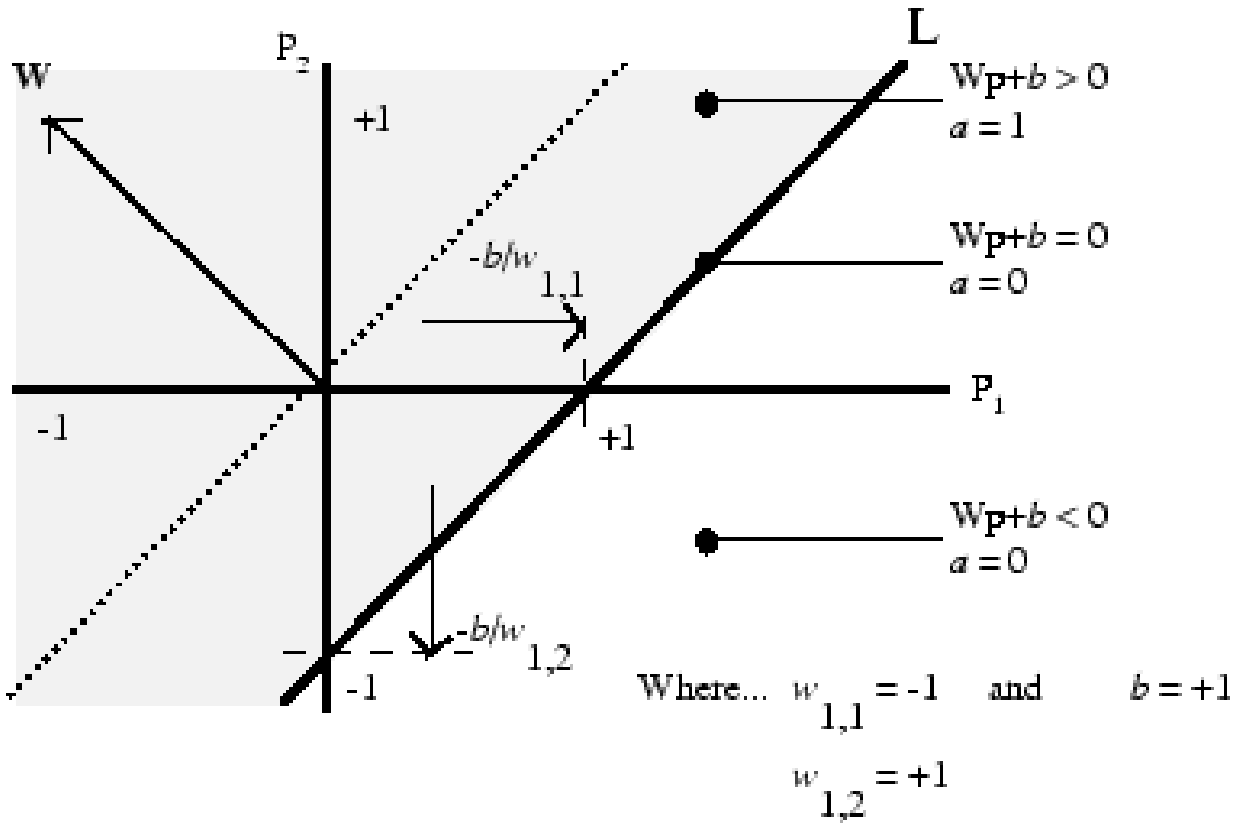


Figure 1.5: Input space of a two-input hard limit neuron

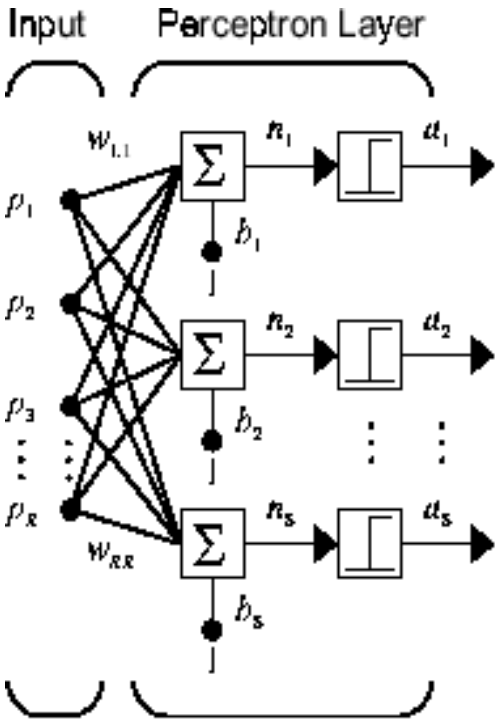
Two classification regions are formed by the decision boundary line L at Wp + b = 0. This line is perpendicular to the weight matrix W and shifted according to the bias b. Input vectors above and to the left of the line L will result in a net input greater than 0 and, therefore, cause the hard-limit neuron to output a 1. Input vectors below and to the right of the line L cause the neuron to output 0. You can pick weight and bias values to orient and move the dividing line so as to classify the input space as desired.

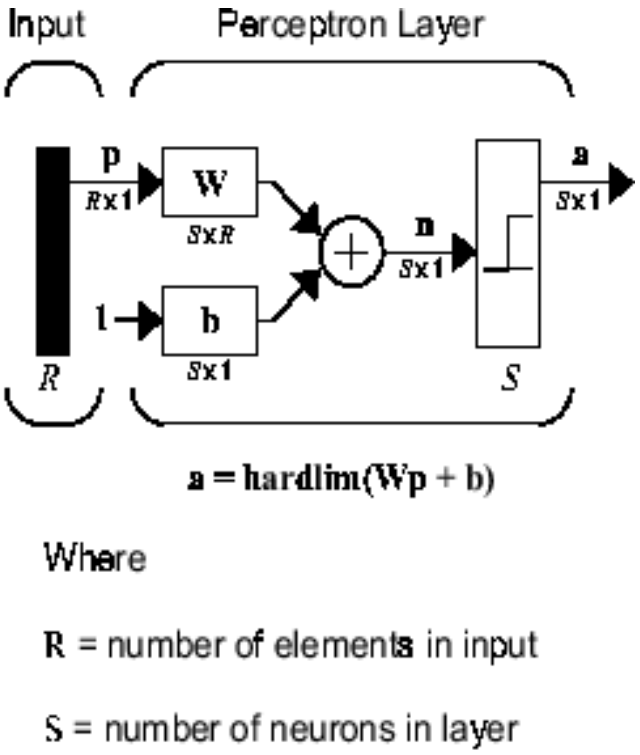
Hard-limit neurons without a bias will always have a classification line going through the origin. Adding a bias allows the neuron to solve problems where the two sets of input vectors are not located on different sides of the origin. The bias allows the decision boundary to be shifted away from the origin, as shown in the plot above

You might want to run the example program nnd4db. With it you can move a decision boundary around, pick new inputs to classify, and see how the repeated application of the learning rule yields a network that does classify the input vectors properly.

3.2.2 Perceptron Architecture

The perceptron network consists of a single layer of S perceptron neurons connected to R inputs through a set of weights wi,j, as shown below in two forms. As before, the network indices i and j indicate that wi,j is the strength of the connection from the jth input to the ith neuron.





The perceptron learning rule described shortly is capable of training only a single layer. Thus only one-layer networks are considered here. This restriction places limitations on the computation a perceptron can perform.

3.2.3 Perceptron Learning Rule (learnp)

Perceptron's are trained on examples of desired behavior. The desired behavior can be summarized by a set of input, output pairs

p1t1,p2t1,…,pQtQ

where p is an input to the network and t is the corresponding correct (target) output. The objective is to reduce the error e, which is the difference t − a between the neuron response a and the target vector t. The perceptron learning rule learnp calculates desired changes to the

perceptron's weights and biases, given an input vector p and the associated error e. The target vector t must contain values of either 0 or 1, because perceptrons (with hardlim transfer functions) can only output these values.

Each time learnp is executed, the perceptron has a better chance of producing the correct outputs. The perceptron rule is proven to converge on a solution in a finite number of iterations if a solution exists.

If a bias is not used, learnp works to find a solution by altering only the weight vector w to point toward input vectors to be classified as 1 and away from vectors to be classified as 0. This results in a decision boundary that is perpendicular to w and that properly classifies the input vectors,

There are three conditions that can occur for a single neuron once an input vector p is presented and the network's response a is calculated:

CASE 1. If an input vector is presented and the output of the neuron is correct (a = t and e = t – a = 0), then the weight vector w is not altered.

CASE 2. If the neuron output is 0 and should have been 1 (a = 0 and t = 1, and e = t – a = 1), the input vector p is added to the weight vector w. This makes the weight vector point closer to the input vector, increasing the chance that the input vector will be classified as a 1 in the future.

CASE 3. If the neuron output is 1 and should have been 0 (a = 1 and t = 0, and e = t – a = –1), the input vector p is subtracted from the weight vector w. This makes the weight vector point farther away from the input vector, increasing the chance that the input vector will be classified as a 0 in the future.

The perceptron learning rule can be written more succinctly in terms of the error e = t – a and the change to be made to the weight vector Δw:

CASE 1. If e = 0, then make a change Δw equal to 0. CASE 2. If e = 1, then make a change Δw equal to pT. CASE 3. If e = –1, then make a change Δw equal to –pT.

3.2.3 Training (train)

If sim and learnp are used repeatedly to present inputs to a perceptron, and to change the perceptron weights and biases according to the error, the perceptron will eventually find weight and bias values that solve the problem, given that the perceptron can solve it. Each traversal through all the training input and target vectors is called a pass.

The function train carries out such a loop of calculation. In each pass the function train proceeds through the specified sequence of inputs, calculating the output, error, and network adjustment for each input vector in the sequence as the inputs are presented.

Note that train does not guarantee that the resulting network does its job. You must check the new values of W and b by computing the network output for each input vector to see if all targets are reached. If a network does not perform successfully you can train it further by calling train again with the new weights and biases for more training passes, or you can analyze the problem to see if it is a suitable problem for the perceptron.

3.2.4 Feed Forward Neural Network

A Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The opposite of a feed forward neural network is a recurrent neural network, in which certain pathways are cycled. The feed forward model is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.

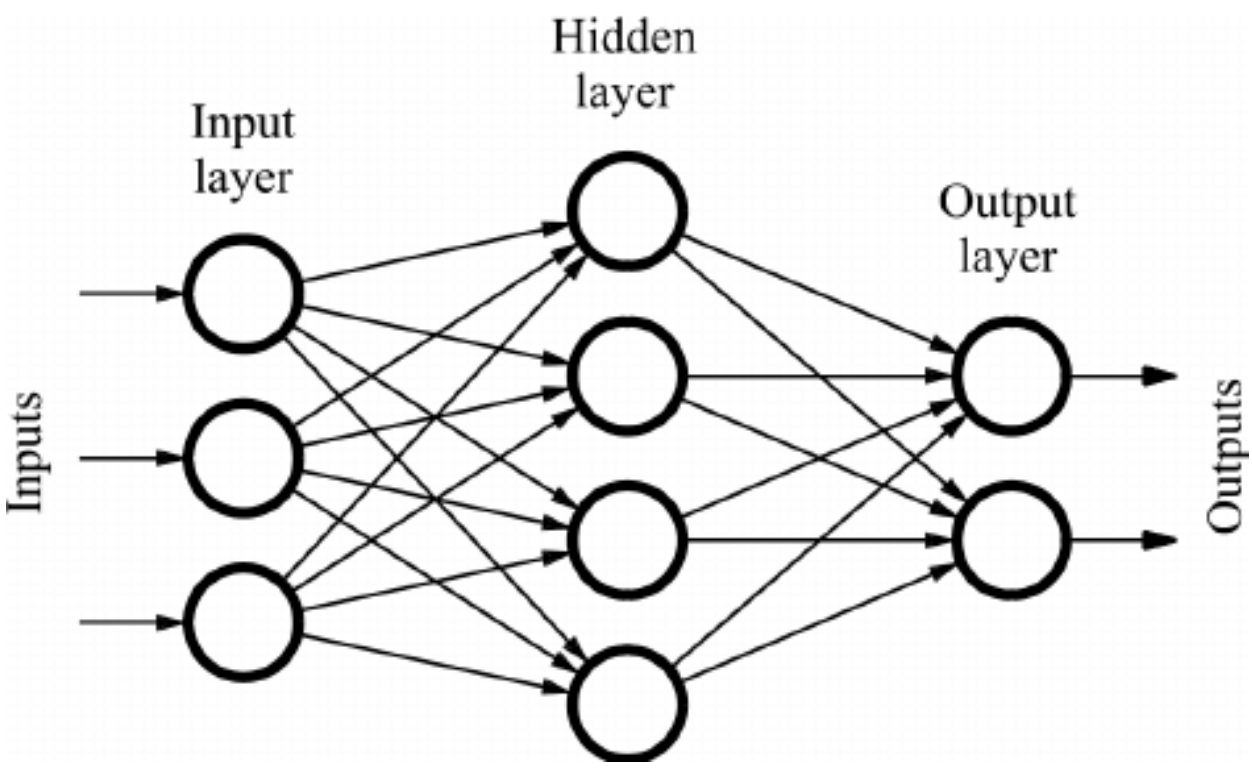


Figure 1.6: Sample of a feed-forward neural networ

3.2.5 Working of Feed Forward Neural Network

Feedforward neural networks were among the first and most successful learning algorithms. They are also called deep networks, multi-layer perceptron (MLP), or simply neural networks. As data travels through the network's artificial mesh, each layer processes an aspect of the data, filters outliers, spots familiar entities and produces the final output.

Feedforward neural networks are made up of the following:

Input layer: This layer consists of the neurons that receive inputs and pass them on to the other layers. The number of neurons in the input layer should be equal to the attributes or features in the dataset.

Output layer: The output layer is the predicted feature and depends on the type of model you're building.

Hidden layer: In between the input and output layer, there are hidden layers based on the type of model. Hidden layers contain a vast number of neurons which apply transformations to the inputs before passing them. As the network is trained, the weights are updated to be more predictive.

Neuron weights: Weights refer to the strength or amplitude of a connection between two neurons. If you are familiar with linear regression, you can compare weights on inputs like coefficients. Weights are often initialized to small random values, such as values in the range 0 to 1.

To better understand how feedforward neural networks function, let's solve a simple problem — predicting if it's raining or not when given three inputs

x1 - day/night

x2 - temperature

x3 – month

Let's assume the threshold value to be 20, and if the output is higher than 20 then it will be raining, otherwise it's a sunny day. Given a data tuple with inputs (x1, x2, x3) as (0, 12, 11), initial weights of the feedforward network (w1, w2, w3) as (0.1, 1, 1) and biases as (1, 0, 0

3.2.6 Here's how the neural network computes the data in three simple steps:

1. **Multiplication of weights and inputs:** The input is multiplied by the assigned weight values, which this case would be the following:

$$(x1^* w1) = (0 * 0.1) = 0$$

$$(x2^* w2) = (1 * 12) = 12$$

$$(x3^* w3) = (11 * 1) = 11$$

2. Adding the biases: In the next step, the product found in the previous step is added to their respective biases. The modified inputs are then summed up to a single value.

$$(x1^* w1) + b1 = 0 + 1$$

$$(x2^* w2) + b2 = 12 + 0$$

$$(x3^* w3) + b3 = 11 + 0$$

$$\text{weighted\_sum} = (x1^* w1) + b1 + (x2^* w2) + b2 + (x3^* w3) + b3 = 23$$

3. Activation: An activation function is the mapping of summed weighted input to the output of the neuron. It is called an activation/transfer function because it governs the inception at which the neuron is activated and the strength of the output signal.

4. Output signal: Finally, the weighted sum obtained is turned into an output signal by feeding the weighted sum into an activation function (also called transfer function). Since the weighted sum in our example is greater than 20, the perceptron predicts it to be a rainy day

The image below illustrates this process more clearly.

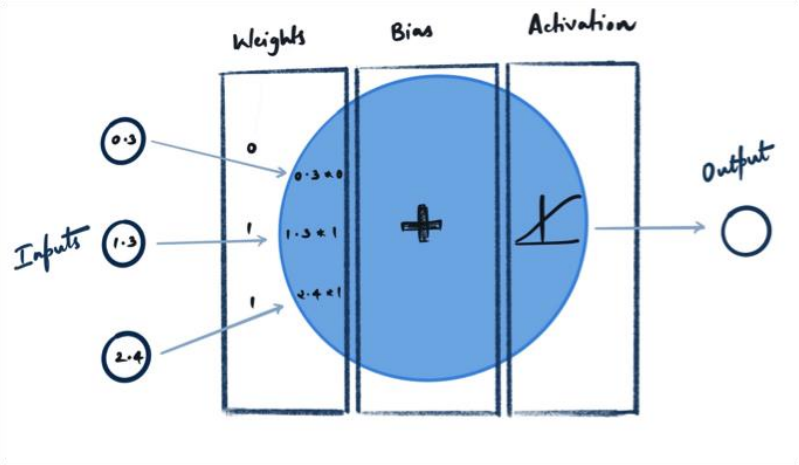


Figure 1.7: Compute data process in Neural Network

There are several activation functions for different use cases. The most commonly used activation functions are relu, tanh and softmax. Here's a handy cheat sheet:

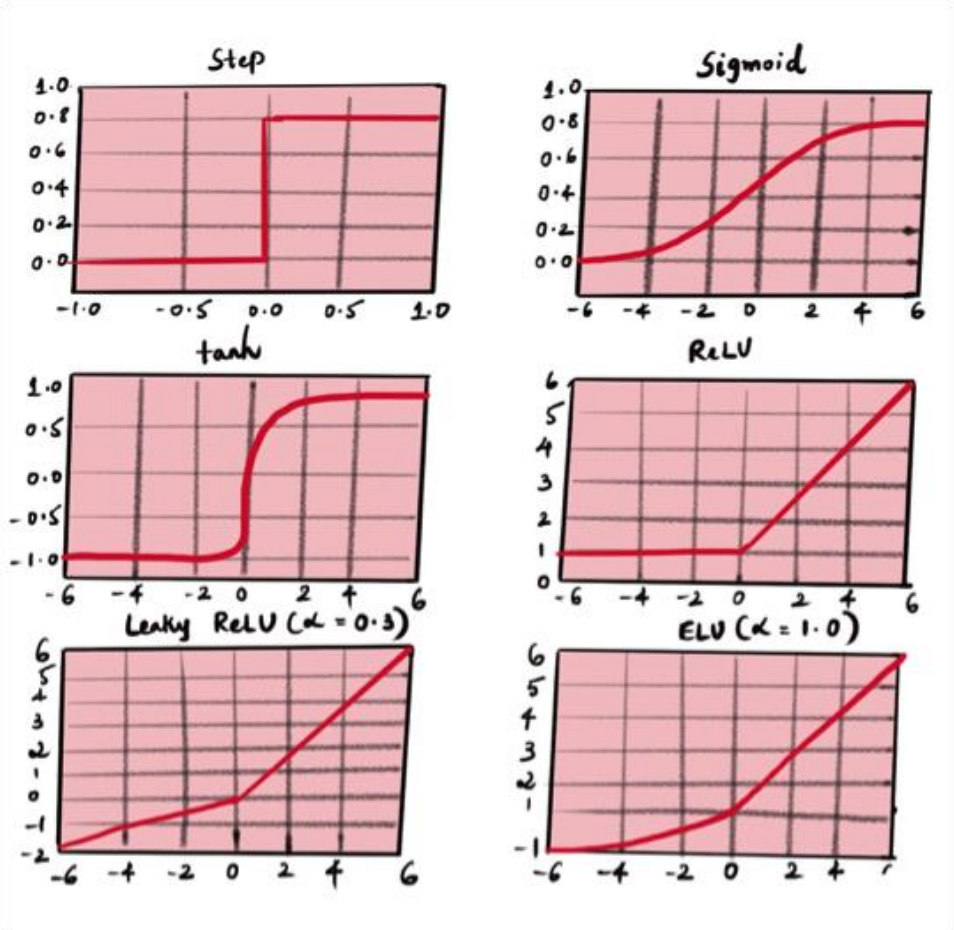


Figure 1.8: Some activation function with their graphs

Calculating the Loss

In simple terms, a loss function quantifies how “good” or “bad” a given model is in classifying the input data. In most learning networks, the loss is calculated as the difference between the actual output and the predicted output.

Mathematically:

$$loss = y_{predicted} - y_{original}$$

The function that is used to compute this error is known as loss function J(.).

Different loss functions will return different errors for the same prediction, having a considerable effect on the performance of the model.

Gradient Descent

Gradient descent is the most popular optimization technique for feedforward neural networks. The term "gradient" refers to the quantity change of output obtained from a neural network when the inputs change a little. Technically, it measures the updated weights concerning the change in error. The gradient can also be defined as the slope of a function. The higher the angle, the steeper the slope and the faster a model can learn.

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function; the procedure is then known as gradient ascent.

"A gradient measures how much the output of a function changes if you change the inputs a little bit.A gradient simply measures the change in all weights with regard to the change in error. You can also think of a gradient as the slope of a function. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematical terms, a gradient is a partial derivative with respect to its inputs.

3.2.7 Neural Network Learning –Trainlm

Trainlm is a net work training function that updates weight and bi as values according to Levenberg- Marquardt optimization.

Syntax:

net.trainFcn = 'trainlm' [net,tr] = train (net,...)

The Levenberg-Marquardt optimization

The procedure used to carry out the learning process in a neural network is called the

optimization algorithm.

The picture below represents a state diagram for the training process of a neural network with the Levenberg-Marquardt algorithm.

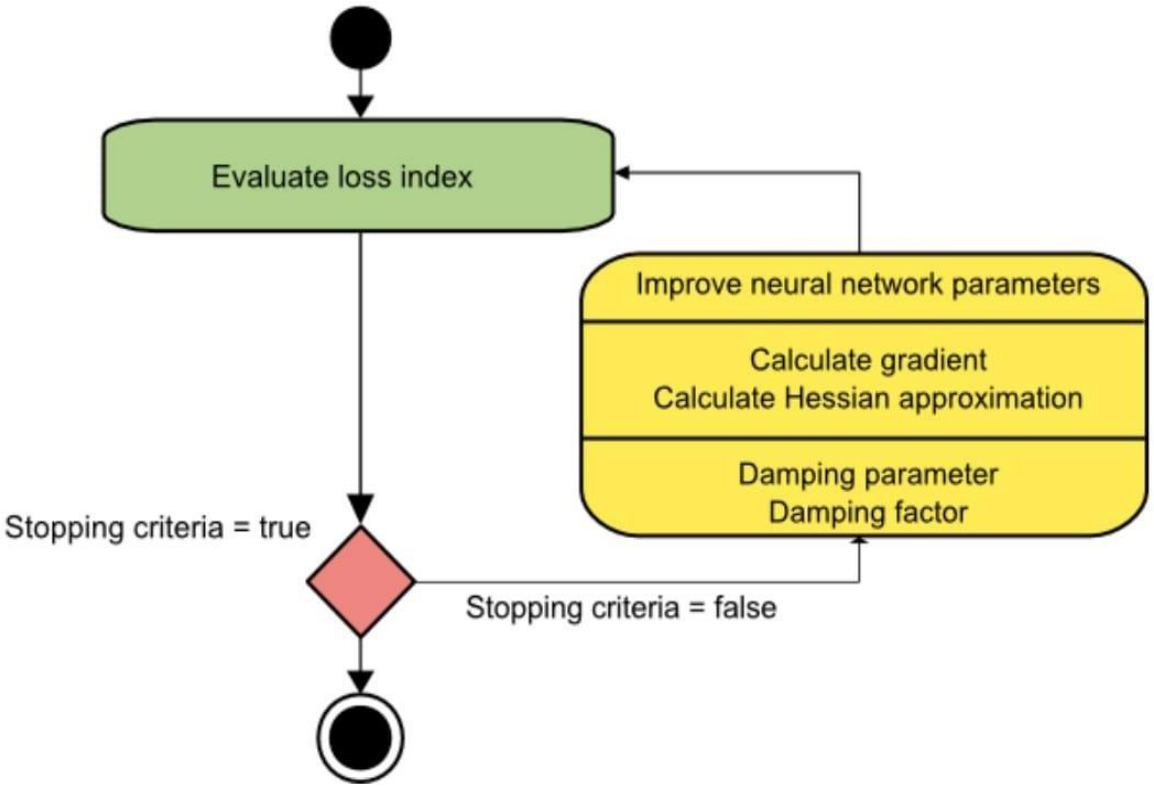


Figure 1.9: Levenberg-Marquardt Algorithm Flowchart

The first step is to calculate the loss, the gradient and the Hessian approximation.

The loss function depends on the adaptive parameters (biases and weights) in the neural network. We can conveniently group them together into a single n-dimensional weight vector.

We can calculate the first and second derivatives of the loss function. The first derivatives are grouped in the gradient vector, whose elements can be writtenas-

$$\nabla_i f(\mathbf{w}) = \frac{\partial f}{\partial w_i},$$

Similarly, the second derivatives of the loss function can be grouped in the Hessianmatrix,

$$\mathbf{H}_{i,j} f(\mathbf{w}) = \frac{\partial^2 f}{\partial w_i \partial w_j},$$

Then the damping parameter is adjusted so as to reduce the loss at eachiteration

TrainlmAlgorithm

Trainlm is often the fastest back propagation algorithm in the toolbox and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. The syntax to use this algorithm is as follows-

net.trainFcn = 'trainlm' sets the network trainFcn property.[net,tr] = train(net,…) trains the network with trainlm.

trainlm supports training with validation and test vectors if the network's NET.divideFcn property is set to a data division function.Validation vectors are used to stop training. There different types of validation such as when- Training stops when any of these conditions occurs or if the maximum number of epochs (repetitions) is reached or if the maximum amount of time is exceeded or if the performance is minimized to the goal, performance gradient falls below min_grad and if the validation performance has increased more than max_fail times since the last time it decreased (when usingvalidation).

3.3 Performance Evaluation Parameters

$$\frac{\text{Number of samples correctly Identified for audio attributes}}{\text{Total number of test attributes}} \times 100$$

3.4 Implementation Details

3.4.1 Overview

For creating the system we use Matlab software. Also, we need a different toolbox for implementation As I told you earlier for this system we created a dataset and standardized it. Now at the time of implementation for getting better accuracy, we download the one dataset from Kaggle and also check the accuracy for the downloaded dataset but system get better accuracy with our created dataset.

For this System, we needed three GUI so we created one to train all the data and also you can check the accuracy but the system needs real-time audio so the second GUI system record the audio in real-time and converted into .wav file, and third GUI system extract features from real- time audio and check it. Now how we made it standardized dataset. We used the Audacity software for cancelling background noise and set the frequency to 44100Hz. In the dataset, there are five different sounds and those sounds are Clear throat, door slam, drawer, Key drop, door Knock, all these sounds were recorded with the help of mobile and for each sound event, we took 20 samples so we record 100 samples.

When the system runs the GUI 2 will open, we suppress the record button. After suppressing recording will start for five minutes. After five minutes recording will stop automatically and that file saved in .wav format. The sound which records in real-time the system change frequency to 44100 Hz and our training samples also have same frequency. GUI 2 is only for recording the sound in real-time and change it into .wav format. Once recorded the real-time sound then load that file into GUI 3 and extract the features from that file because the feature extraction technique is very important for extracting a feature from audio now in the system there are only two types of feature extraction technique we used one is MFCC and another is LPCC. Now when GUI 3 will open then suppress Load real-time audio then load that .wav file. Then load that folder, when we recorded our real-time audio that audio save in that folder it happened by default it means in the system we were set the path for that and every time in that folder file is re-write it means the file is same but every time audio is different, different means it stores new real-time recording each time which we recorded in GUI 2. As I said earlier in GUI 3 when we suppressed button for loading the real-time audio. After that, we need to select MFCC or LPCC feature extraction technique and then we suppress extract features then features extraction process is completed.

3.4.2 Preprocessing training data

We created dataset for this project. Now recording is done on mobile phones. We created a dataset for this project. The recording was recorded on mobile phones. After completing we were used Audacity software to set the frequency to 44100 Hz

3.4.3 Training of neural network

Feed Forward neural network is used for training purpose. Neural network is as follows,

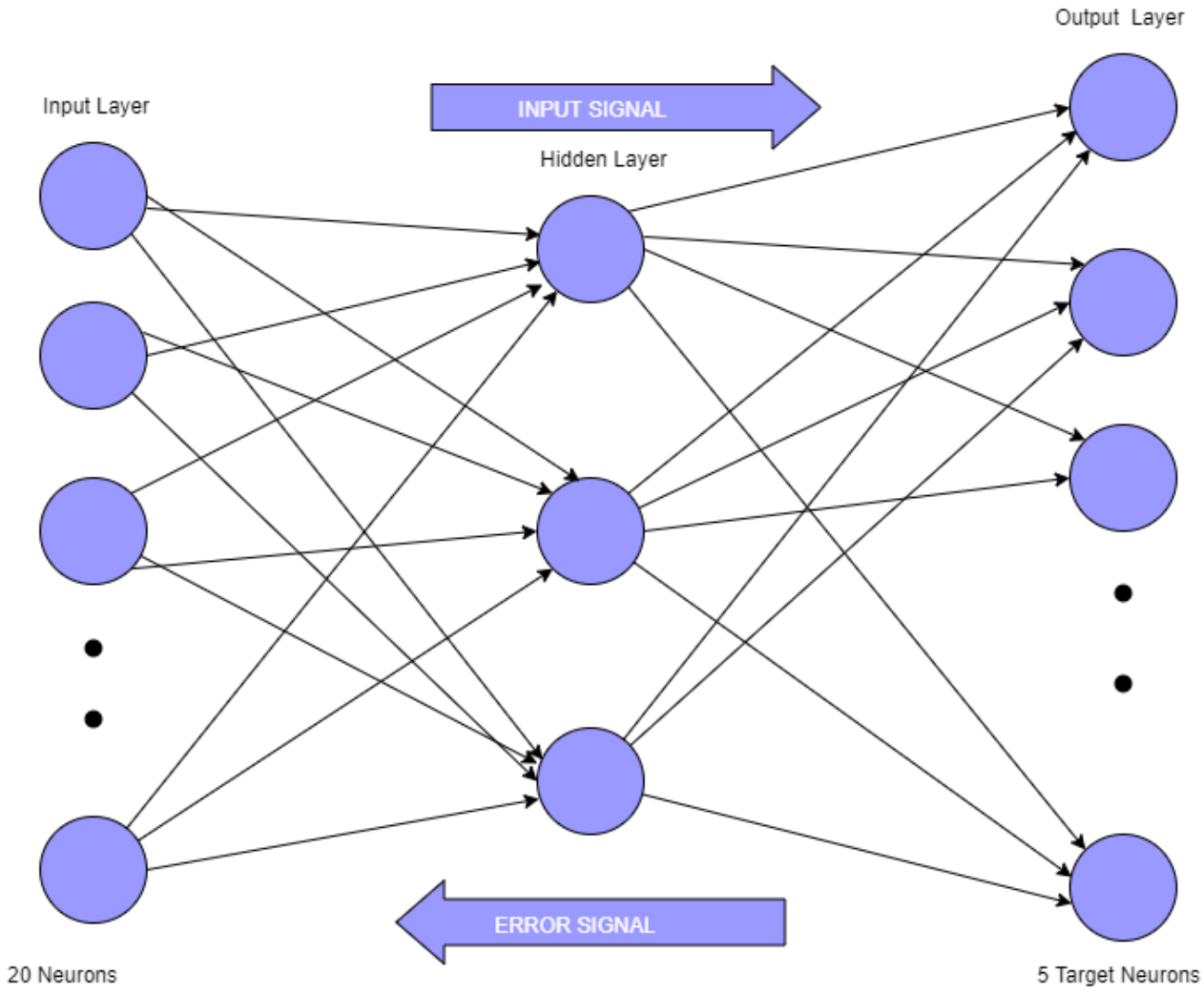
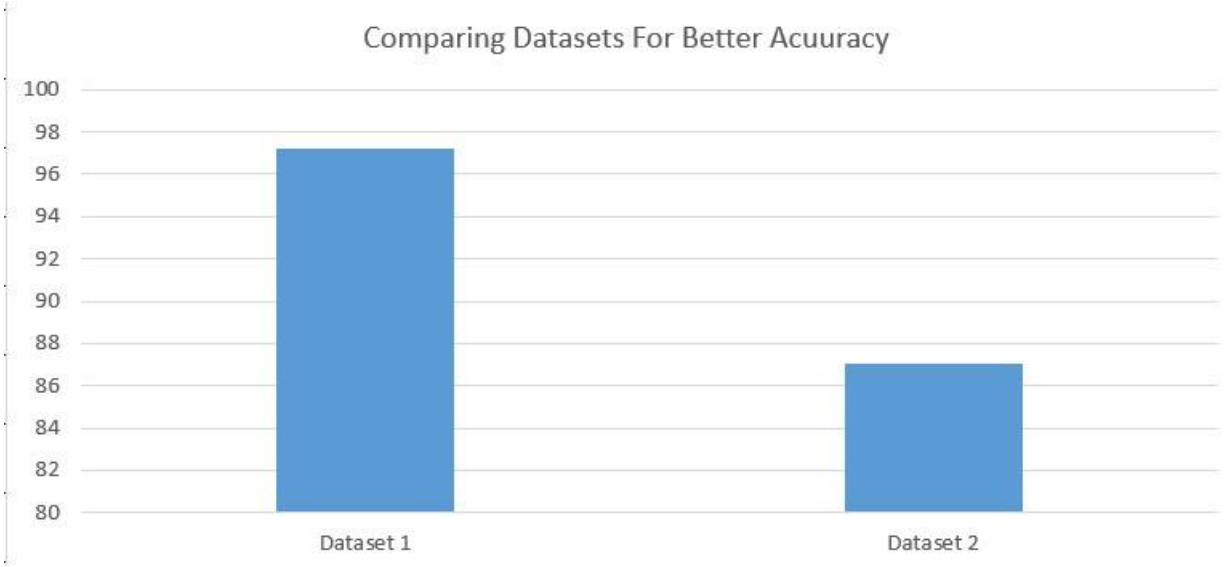


Figure 1.10: Feed Forward Neural Network

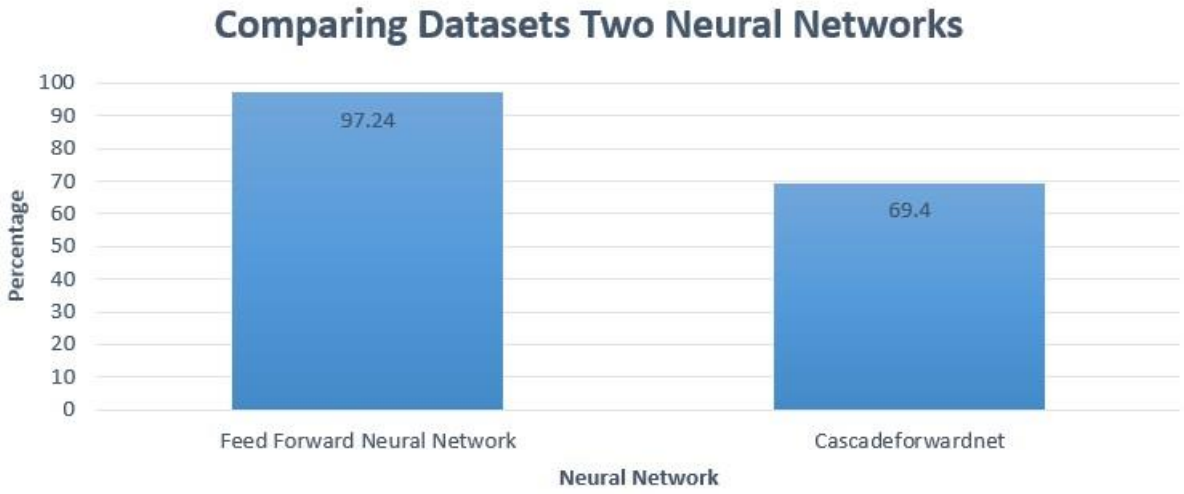
following are the graph which show the comparison between two datasets.



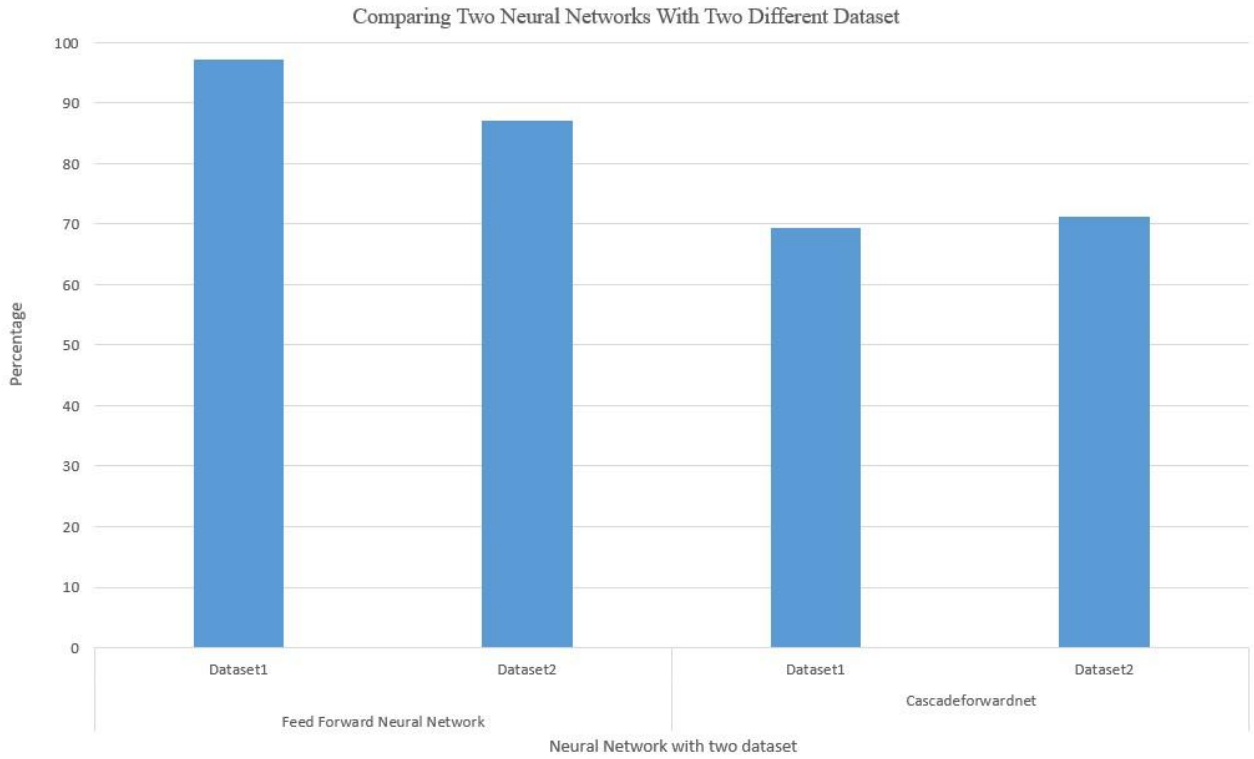
Graph 1.1: Comparing training of dataset for better accuracy

Now as we see in above graph there is two datasets namely as dataset1 and dataset 2. Dataset 1 is that dataset which we created for this project and dataset 2 is standard dataset but after training datasets dataset 1 gives better accuracy than dataset 2 .This is training accuracy.

following are the graph which show the comparison between two Neural Networks.



Graph 1.2: Comparing Neural Networks for better accuracy



Graph 1.3: Comparing Neural Networks along with two datasets for better accuracy

4 PERFORMANCE ANALYASIS

4.1 Accuracy / Testing

The recorded notes, major chords and minor chords when generated the target and then trained to the feed forward back propagation neural network gives the output. Then the audio produced from other set of strings which possess some different quality from the previous recorded audio generates the target. This target is then given for the training of the neural network. The output built after training is further compared with human feedback which results to accuracy of the system.

$$\text{System accuracy} = \frac{\text{Number of correctly identify samples}}{\text{Total Number of samples}} * 100$$

The accuracy evaluated by the system is 80.00%

4.2 GUI and Graphs

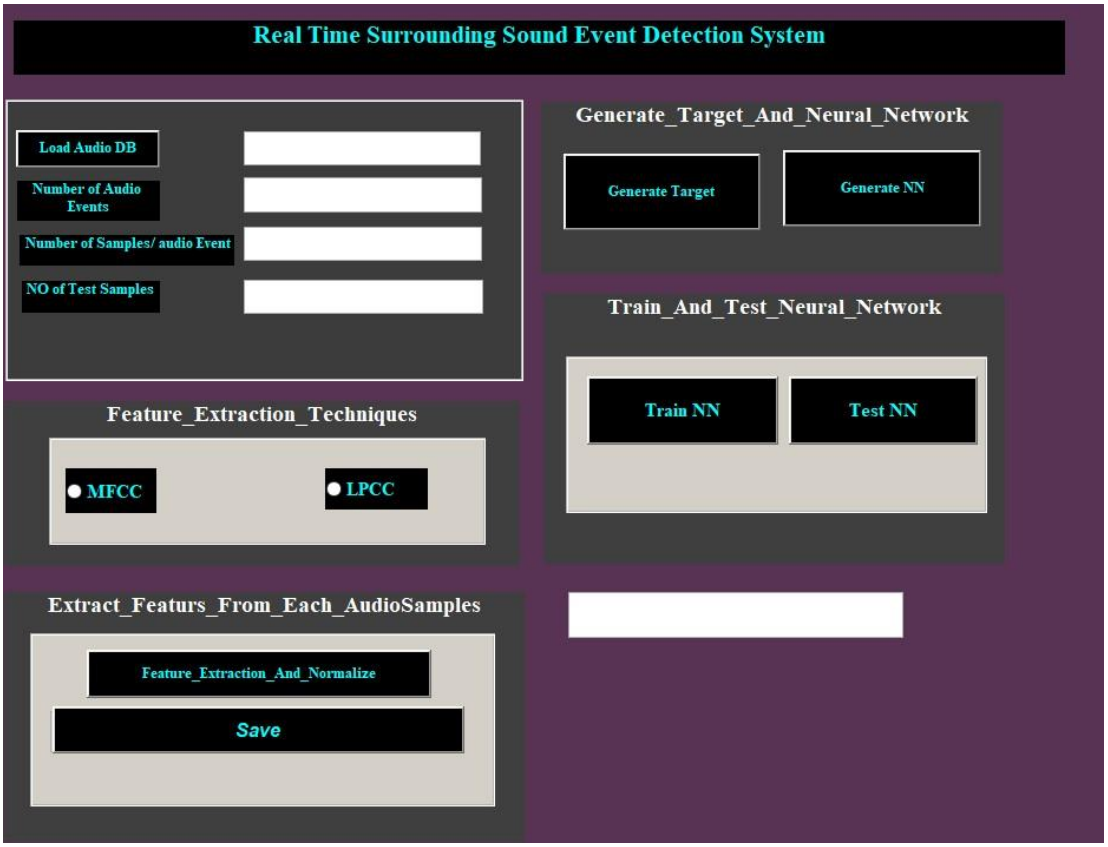


Figure 1.11:GUI 1

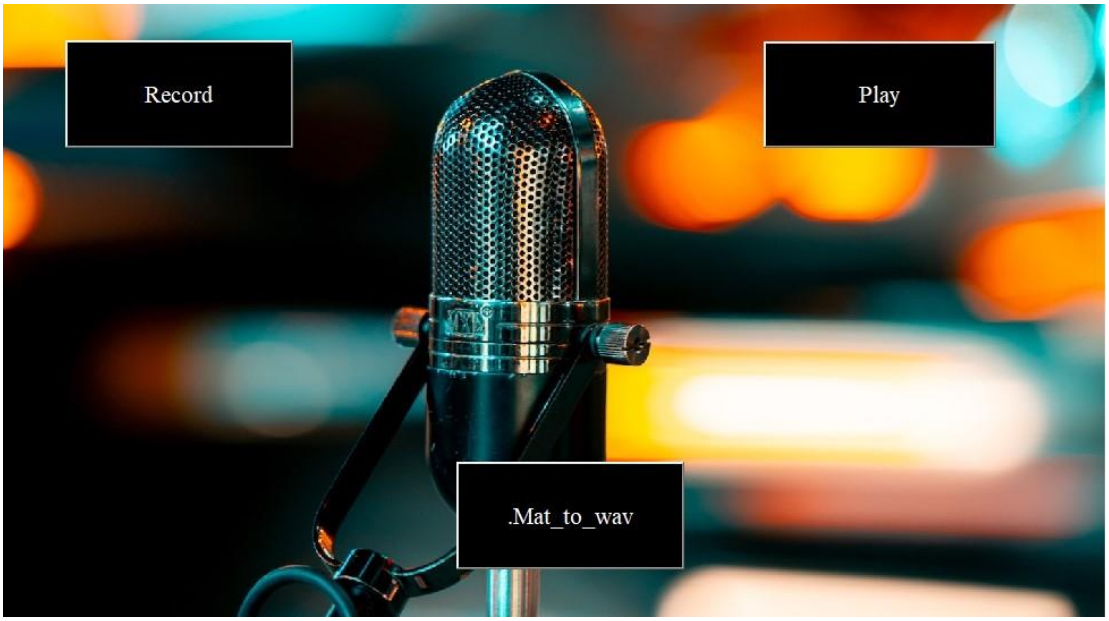


Figure 1.12:GUI 2

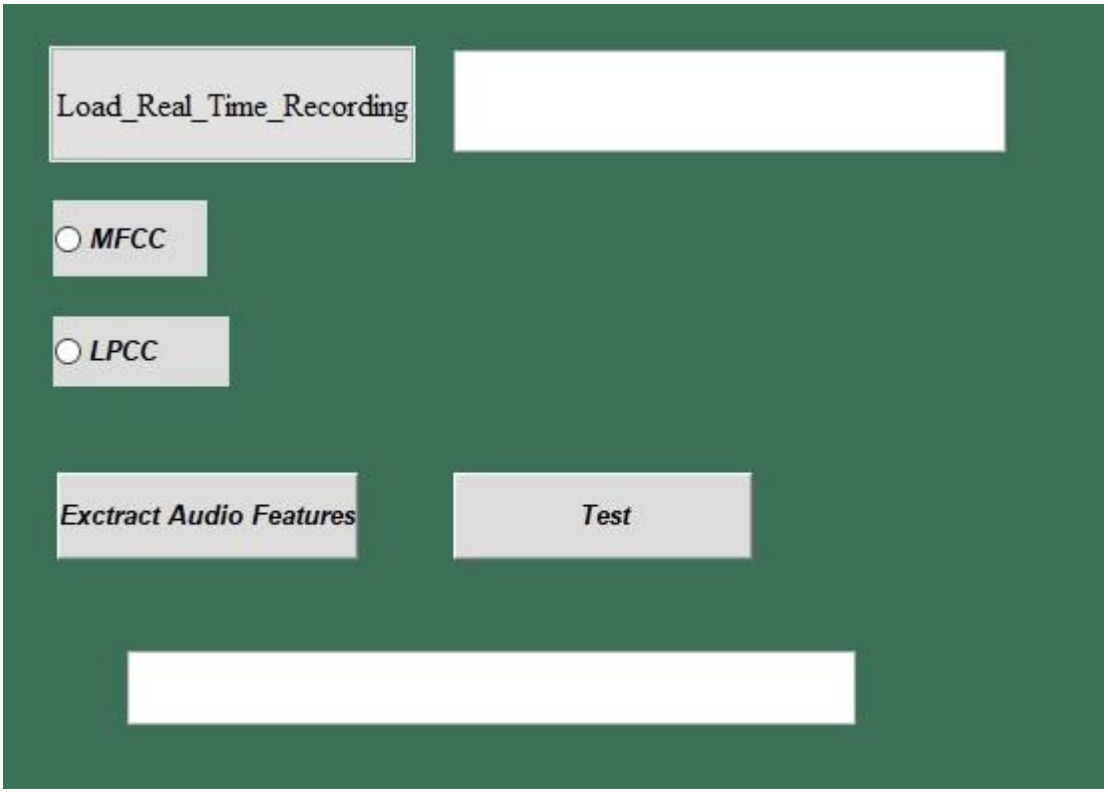


Figure 1.13:GUI 3

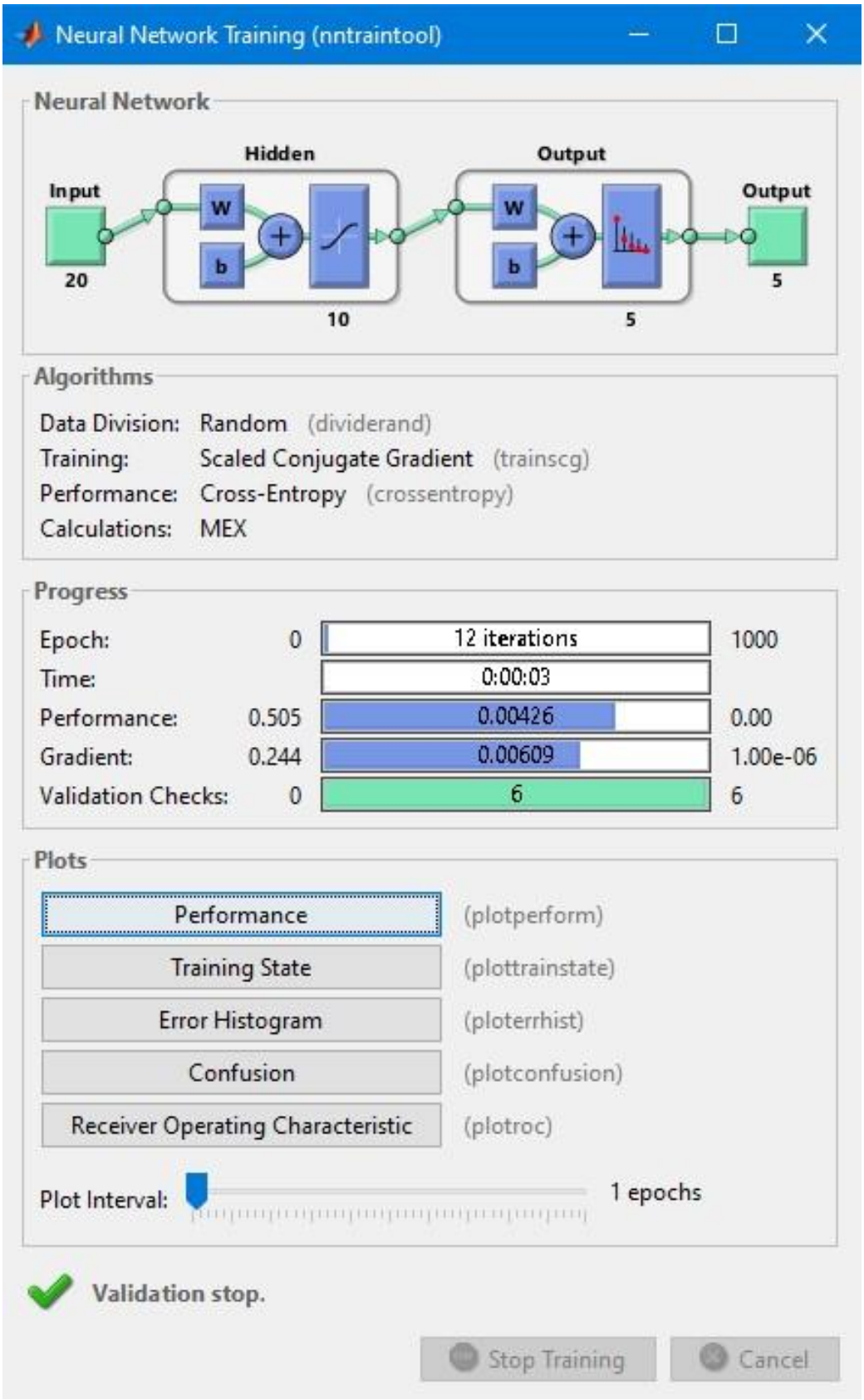
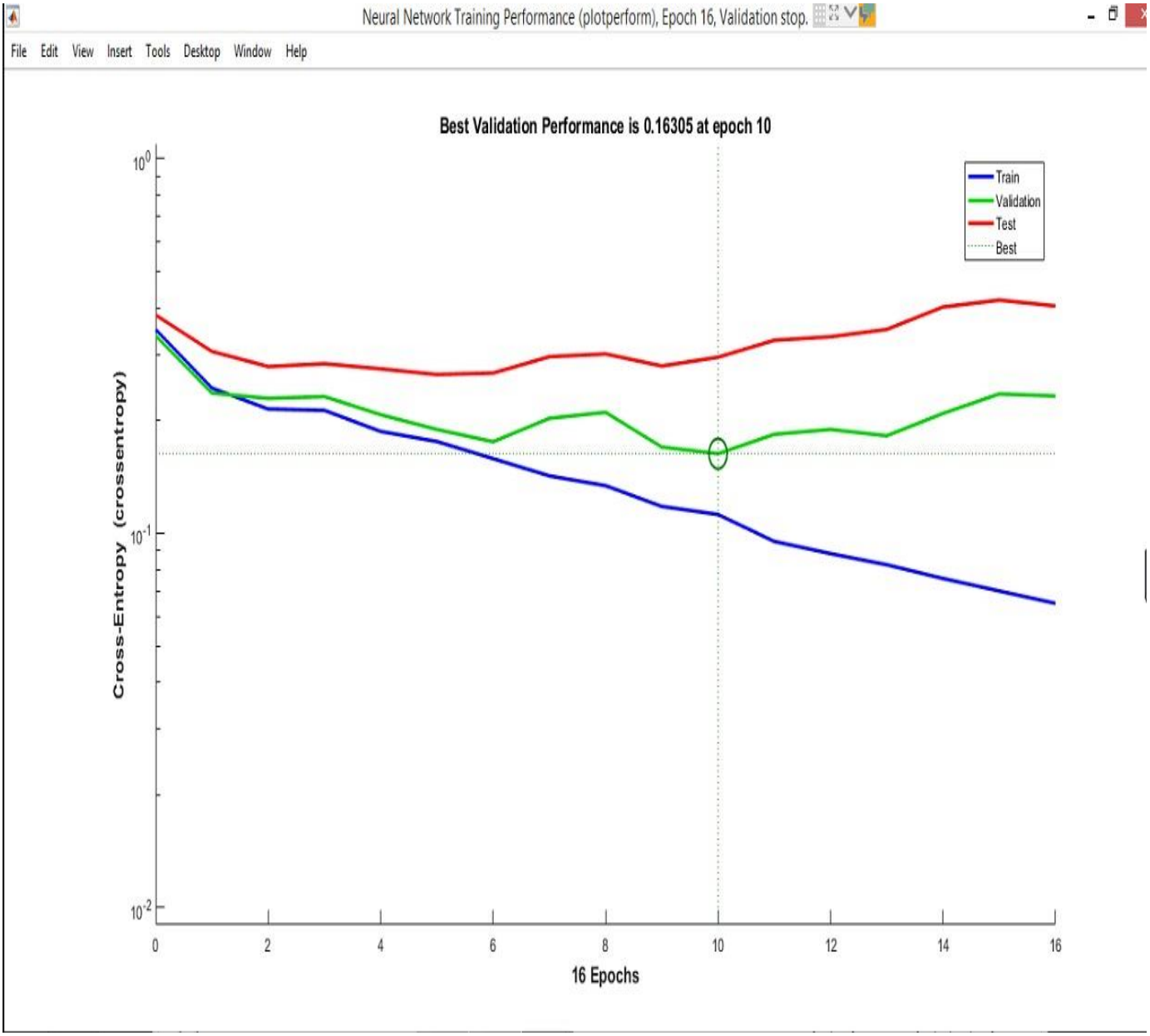
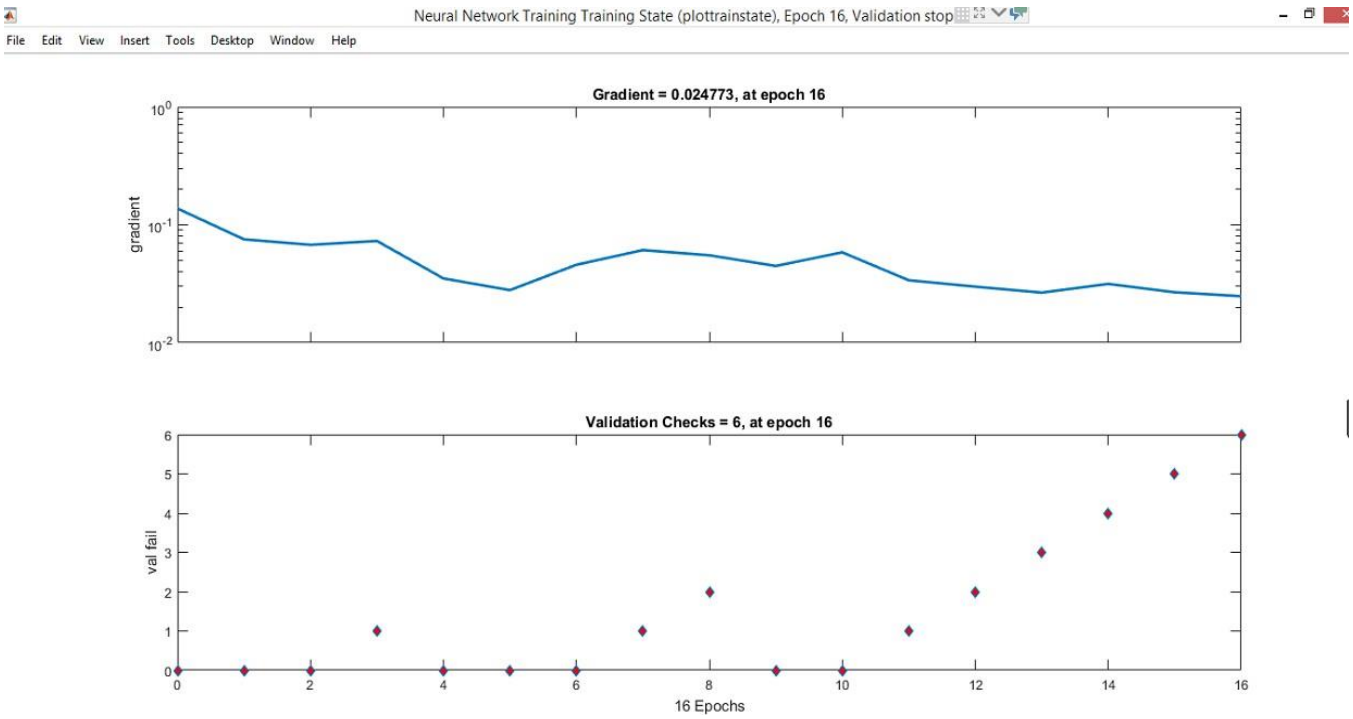


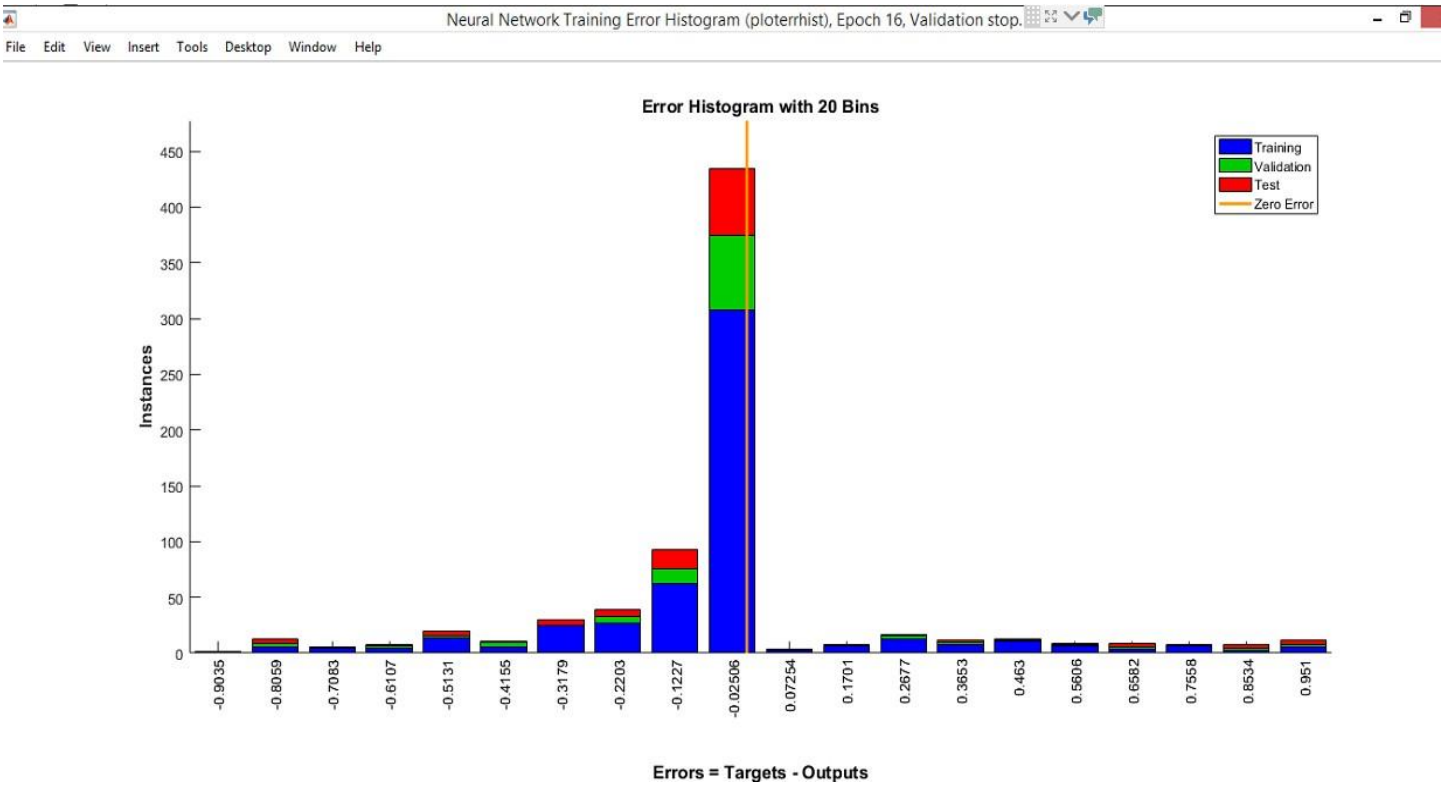
Figure 1.14: Training of Neural Network



Graph 1.4: Training Performance



Graph 1.5: Neural Network Training State



Graph 1.6: Neural Network Training Error Histogram

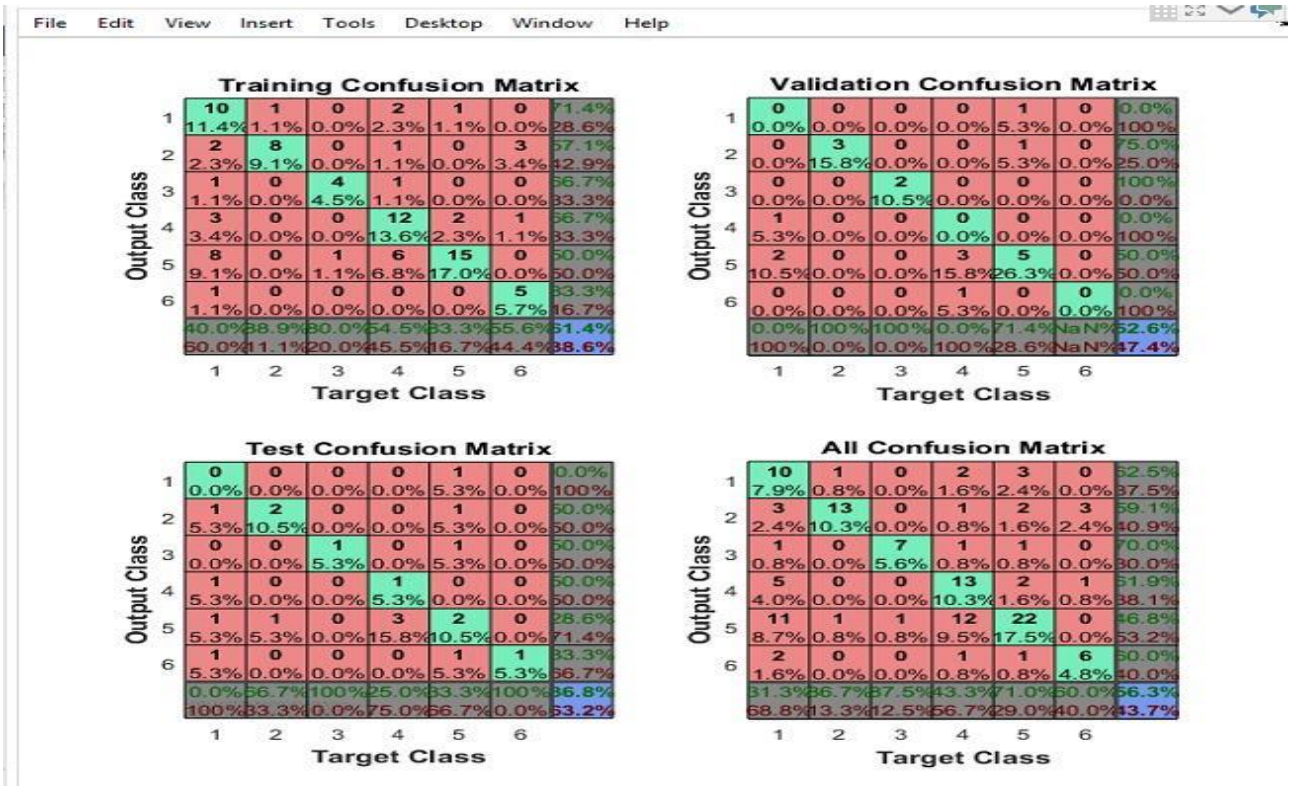


Figure 1.15: Confusion Matrix

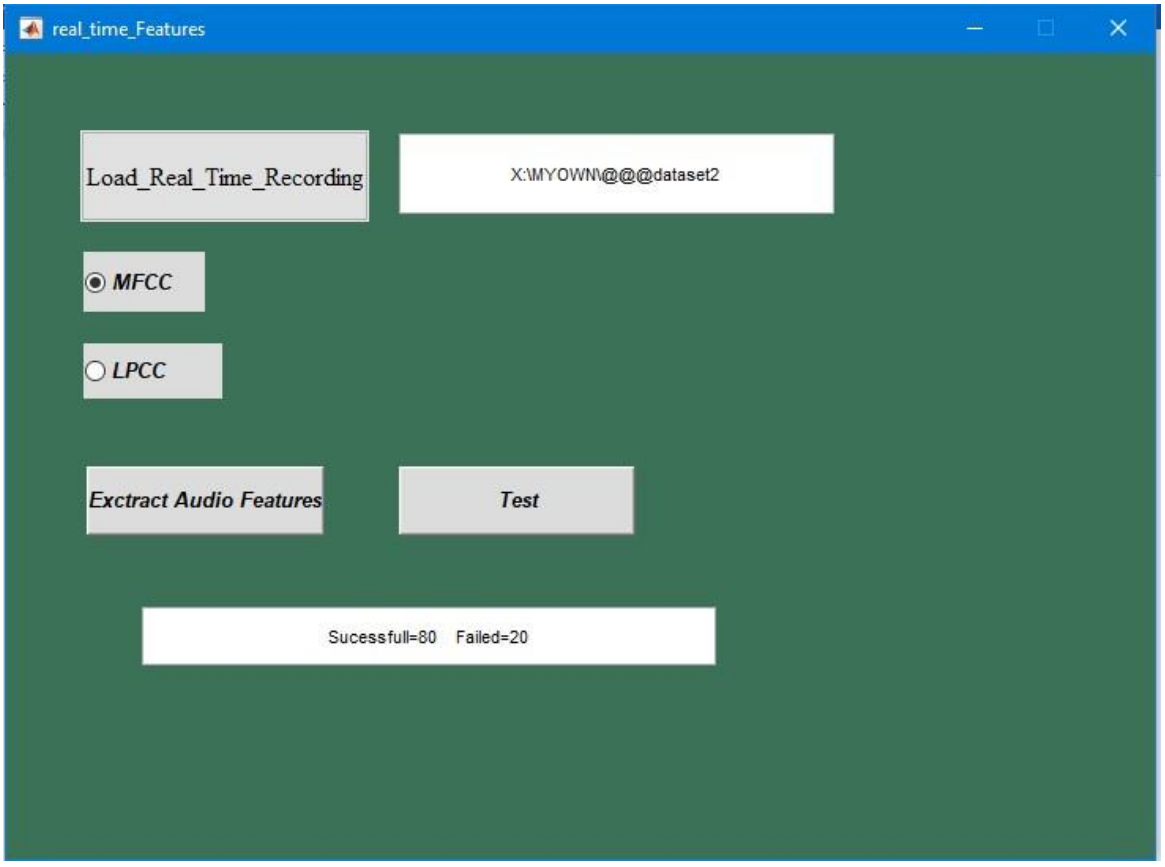


Figure 1.16: System Percentage Accuracy

5 CONCLUSION

5.1 Conclusion

We have identified the real-time surrounding sound event detection by creating the dataset and extracted audio features using MFCC feature extraction technique. Normalization is performed on MFCC features and neural network is trained from the target neurons generated which we recorded from the real-time and that file give to the system and that is the target for the system. Around we test 10 neural networks from that we analyze which neural network gives the best accuracy finally we used feed-forward neural network and our system got the percentage accuracy of 80.00%.

5.2 Future Scope

MFCC is majorly considered as a part of Feature extraction technique. MFCC along with feature extraction may increase the accuracy.

Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) other than Feed Forward Neural Network may enhance the performance. In this research the only thing we looked at was five sounds are taken. To extend or improve this research multiple aspects could be investigated. Of course it is possible to use various sounds we only used five for this system. Adding more sounds to the knowledge of surrounding sounds which is present in room is a useful thing to do. It is also possible to use other or more sounds and try to find out the very minute sounds which humans are just ignored. This project is not only detecting the sounds but also it depends on signal processing and signal processing is the technology of the future .

REFERENCES

Bibliography

[1] E. Gómez and J. Urbano M. Schedl, "Music Information Retrieval:Recent Developments and Applications," *Foundations and Trends R in Information Retrieval*, vol. 8, no. 978-1-60198-2, 2014.

[2] Daigneau Madeleine, "Advanced Music Audio Feature Learning with Deep Networks," *Rochester Institute of Technology*, 2017.

[3] Kohshelan A/L Sundararajoo, "Improvement of Audio Feature ExtractionTechniques in Traditional Indian String Musical Instrument," *University Tun Hussein Onn Malaysia*, August 2015.

[4] https://www.ideals.illinois.edu/bitstream/handle/2142/101116/PHAN-THESIS-2018.pdf?sequence=1&isAllowed=y

[5] www.datascience.com. [Online]. https://www.datascience.com/blog/k-means-clustering

[6] https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

ACKNOWLEDGEMENT

The portion of success is brewed by the efforts put in by many individuals. It is constant support provided by people who gave us the initiative, who inspired us at each step of our endeavor that eventually helped us in our goal.

We wish to express our deep gratitude and heartily appreciation for the invaluable guidance of our professors throughout the span of preparing this seminar. We are indebted to our college **Principal Dr. S. P. Bhosle.**

We are also thankful to our **HOD Dr. S. L. Kasar** and our **Project Guide Dr.S.H.Deshmukh** for their invaluable and elaborate suggestions. Also we are thankful to **Prof. Shambhavi Shete** for her contribution. Their excellent guidance made us to complete this task successfully within a short duration.

The inspiration behind the every aspect of life constructs a way to get success, which we have got from all the professors of the department.

No thanks giving would be complete without mentioning our parents and friends, without their constant support and encouragement, this assignment would have not been successful.

SURAJ HOTKAR(64)