

Human Voice Recognition in the Regional Language

A Project Work Submitted in Partial Fulfillment
of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

ANUP BURNWAL (Roll No. 10700116043)
&
VIVEK KUMAR YADAV (Roll No. 10700116002)

Under the supervision of
Dr. Alok Ranjan Pal



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING & MANAGEMENT, KOLAGHAT

(Affiliated to MAKAUT, WB)

Purba Medinipur – 721171, West Bengal, India

July, 2020



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING & MANAGEMENT, KOLAGHAT

(Affiliated to MAKAUT, WB)

Purba Medinipur – 721171, West Bengal, India

CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this project entitled **Human Voice Recognition in the Regional Language** submitted by ANUP BURNWAL and VIVEK KUMAR YADAV, to the Department of Computer Science & Engineering, is carried out under my direct supervision and guidance.

The project work has been prepared as per the regulations of West Bengal University of Technology and I strongly recommend that this project work be accepted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science & Engineering.

Supervisor

Dr. Alok Ranjan Pal

Asst. Prof., Dept. of CSE

Signed by HOD(CSE)

Dr. Tapas Kr. Maity

Head, Department of CSE



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING & MANAGEMENT, KOLAGHAT

(Affiliated to MAKAUT, WB)

Purba Medinipur – 721171, West Bengal, India

Certificate by the Board of Examiners

This is to certify that the project work entitled **Human Voice Recognition in the Regional Language** submitted by ANUP BURNWAL & VIVEK KUMAR YADAV to the Department of Computer Science and Engineering of College of Engineering of Management, Kolaghat has been examined and evaluated.

The project work has been prepared as per the regulations of West Bengal University of Technology and qualifies to be accepted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science & Engineering.

Project coordinator

Board of Examiners

ACKNOWLEDGEMENT

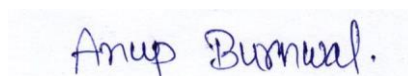
It gives us great pleasure to find an opportunity to express our deep and sincere gratitude to our project guide Dr. Alok Ranjan Pal. We do very respectfully recollect his constant encouragement, kind attention and keen interest throughout the course of our work. We are highly indebted to him for the way he modeled and structured our work with his valuable tips and suggestions that he accorded to us in every respect of our work.

We are extremely grateful to the Department of Computer Science & Engineering, CEMK, for extending all the facilities of our department.

We humbly extend our sense of gratitude to other faculty members, laboratory staff, library staff and administration of this Institute for providing us their valuable help and time with a congenial working environment.

Last but not the least; we would like to convey our heartiest thanks to all our classmates who time to time have helped us with their valuable suggestions during our project work.

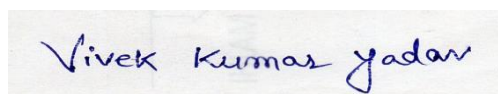
Date: 9th July' 2020



ANUP BURNWAL

University Roll:10700116043

University Registration No: 161070110006



VIVEK KUMAR YADAV

University Roll:10700116002

University Registration No: 161070110047

ABSTRACT

This project ‘Human Voice Recognition in the Regional Language’ is basically speech recognition system to recognize a phone from a set of phones in a regional Language (here it is Hindi).

Speech recognition is widely used nowadays in many aspects of our life be it e-commerce, defence, health sectors, education sectors etc. Speech recognition mechanism comes under the domain of Natural Language Processing(NLP).

The aim here is to recognize the human voice in regional language(Hindi) from a specific set of phones of that language. Under this project we will be focusing on six different phones of the language (which are आगे (āge), पीछे (peechhe), बाएँ (bāen), दाएँ (dāen), चलो (chalo), रूको (ruko)) and then use this recognition system to operate a toy vehicle based on Arduino through on-spot human voice.

For the above approach, a test audio of human voice will be fed to the trained module through microphone and the closeness of that audio is checked with the above given six phones and prediction is done for the phone which has the highest value of probability for the given audio.

For making the training module, a Recurrent Neural Network was made which was fed with a MFCC value of around 400 audio files (manually collected) consisting of the above mentioned phones and a model was trained.

The final results are quite encouraging giving an accuracy of (80-85) % when tested on the files present in the system whereas gives promising on-spot testing for other phones except for the phone āge (आगे) as this phone is somewhat similar in structure to other phones like (dāen, bāen).

Table of contents

Sl. No.	Content	Page no.
1	Introduction	1
2	Theoretical Background	2
3	History	10
4	A Brief Survey	13
5	Pre-processing Work	18
6	Proposed Approach	21
7	Results	30
8	Few Close Observations	39
9	Future Scope	40
10	Conclusion	41
	Reference	42

1. INTRODUCTION

The Speech is the most common & primary mode of communication among human beings. It is the most natural and efficient form of exchanging information among humans. Human voice conveys much more information such as gender, emotion and identity of the speaker.

Speech recognition is simply the ability of a software to recognise speech. Anything that a person says, in a language of their choice, must be recognised by the software. Speech recognition technology can be used to perform an action based on the instructions defined by the human. The human needs to train the speech recognition system by storing speech patterns and vocabulary of their language into the system. By doing so, they can essentially train the system to understand them when they speak.

The speech recognition system however is not something new. It has been decades since the first speech recognition model was proposed and in today's world with advancement of technology and resources many complex recognition system has been built. Be it Google (OK Google), Alexa or anything else they are able to recognise complex sentences and give results according to that. In field of machines or automobiles also speech recognition has stretched its leg and these can be operated based on human voice. But one thing common in these is that they use English as a language for operation or some language that is not so complex or has that huge different meanings for the same word, for example the word 'haath' can be expressed for different meanings thus any fully operational system on these complex language is difficult to make. Thus this project is towards recognising a speech in complex regional language i.e. Hindi.

The proposed approach is making a speech recognition system which can predict human voice for a specific domain of phones (that are आगे (āge), पीछे (peechhe), बाएँ (bāen), दाएँ (dāen), चलो (chalo), रुको (ruko)). The expectation from the system is to make a promising model for predicting the model which can predict voices from human which was thought to be fed to an Arduino based toy vehicles which can be operated from the human voice. This proposed approach can be used to operate some basic home appliances like A.C. or microwave in a regional language.

2. THEORETICAL BACKGROUND

2.1 Types of Speech Recognition

Speech recognition system can be divided into the number of classes based on their ability to recognize that words and list of words they have. A few classes of speech recognition are classified as under

2.1.1 Isolated speech

The Isolated word has sample windows. It accepts single word or single utterances at a time. Isolated utterance might be a better name of this work.

2.1.2 Connected speech

The Connected word system are similar to isolated words but allow separate utterance to be “run together minimum pause between them.

2.1.3 Continuous speech

It allows user to speak almost naturally, while the computer will examine the content. there are special methods used to determine utterance boundaries and various difficulties occurred in it.

2.1.4 Spontaneous speech

A System with spontaneous speech ability should be able to handle a variety of natural speech feature such as words being run together.

2.2 Sampling of Voice Signals

2.2.1 Definition

Sampling rate or sampling frequency defines the number of samples per second (or per other unit) taken from a continuous signal to make a discrete or digital signal. For time-domain signals like the waveforms for sound (and other audio-visual content types), frequencies are measured in hertz (Hz) or cycles per second.

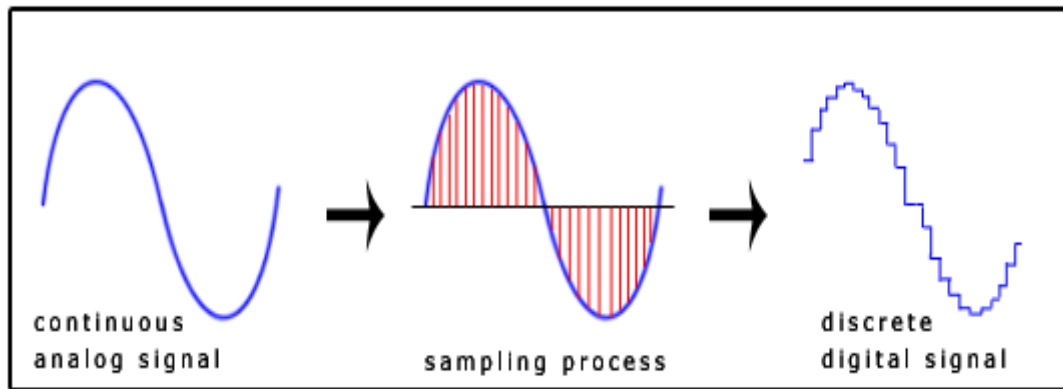


Fig 1(a). Sampling of an audio signal

The **Nyquist–Shannon sampling theorem** (**Nyquist principle**) states that perfect reconstruction of a signal is possible when the sampling frequency is greater than twice the maximum frequency of the signal being sampled. For example, if an audio signal has an upper limit of 20,000 Hz (the approximate upper limit of human hearing), a sampling frequency greater than 40,000 Hz (40 kHz) will avoid aliasing and allow theoretically perfect reconstruction.

#Sampling rate of human speech: 16k samples/sec.

#Sampling rate of recorded audio: 44k samples/sec.

2.3 Fast Fourier Transform(FFT)

Fourier analysis of a periodic function refers to the extraction of the series of sines and cosines which when superimposed will reproduce the function. This analysis can be expressed as a Fourier series.

The Discrete Fourier Transform(DFT) is given by:

$$DFT = \sum_{t=0}^{t-1} g(x)e^{-2\pi ift}$$

where $g(x)$ is time-series audio signal, t is range of time and f is the frequency at which the wave is rotated around a circle (cycle/second).

In terms of complexity the DFT gives an order of $O(n^2)$ which is huge as the range of time increases.

The fast Fourier transform is a mathematical method for transforming a function of time into a function of frequency. The Fast Fourier Transform (FFT) is an algorithm that determines Discrete Fourier Transform of an input significantly faster than computing it directly. The FFT reduces the number of computations needed for a problem of size N from $O(N^2)$ to $O(N \log(N))$.

One important application is for the analysis of sound. It is important to assess the frequency distribution of the power in a sound because the human ear exercises that capacity in the hearing process.

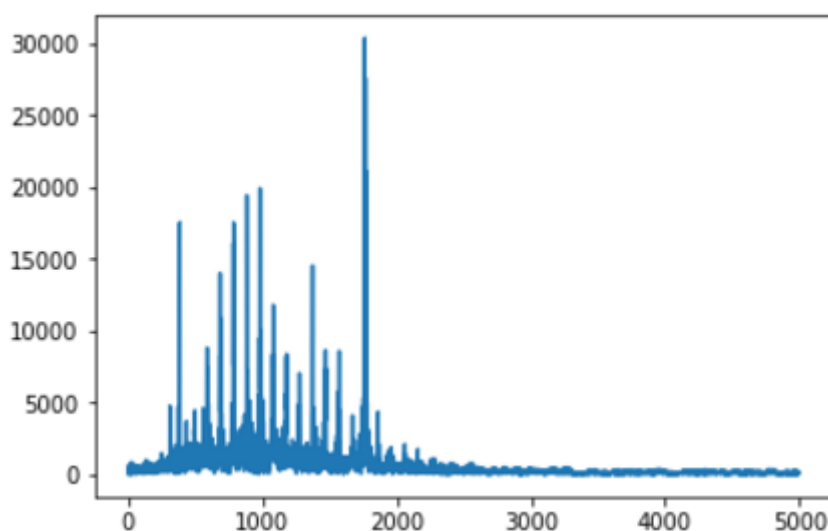


Fig 1(b). Discrete Fourier Transform of an audio signal

2.4 Short-Time Fourier Transform(STFT)

The Short-time Fourier transform (STFT), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.

In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This reveals the Fourier spectrum on each shorter segment. One then usually plots the changing spectra as a function of time, known as a spectrogram or waterfall plot.

The usual mathematical definition of the STFT is

$$Xn(\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n - mR)e^{-j\omega n}$$

where

$Xn(\omega)$ = DFT of windowed data centered at time n

$w(n)$ = window function at time n

$x(n)$ = input signal at time n

mR = hop size between two successive window

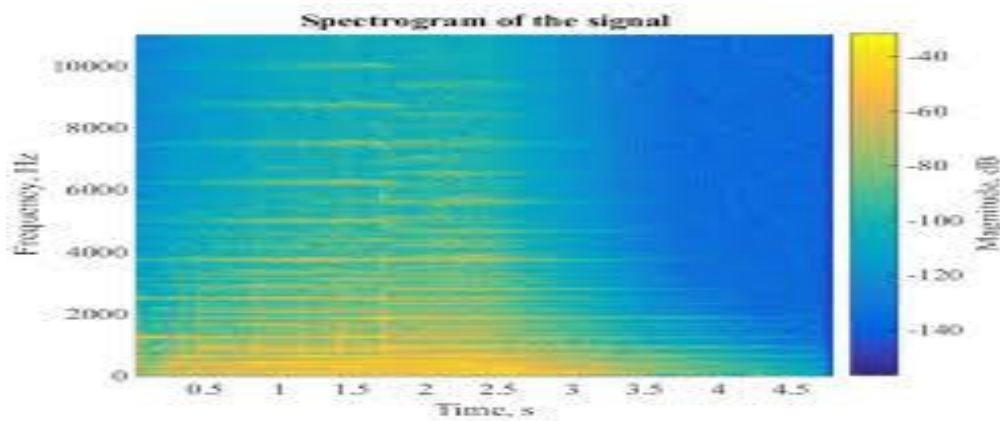


Fig 1(c). Spectrogram of an audio signal

2.5 Mel-Frequency Cepstrum

In sound processing, the Mel-frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency.

Mel-frequency Cepstral Coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the Mel-frequency

cepstrum is that in the MFC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

MFCCs are commonly derived as follows:

- Take the Fourier transform of (a windowed excerpt of) a signal.
- Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows.
- Take the logs of the powers at each of the Mel frequencies.
- Take the discrete cosine transform of the list of Mel log powers, as if it were a signal.

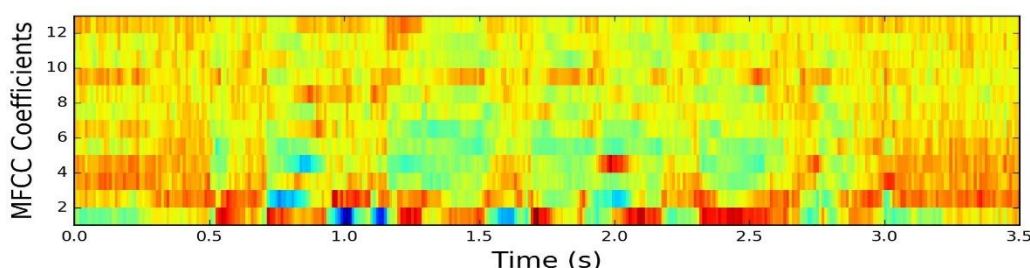


Fig 1(d). MFCC of an audio signal

2.4.1 Application of MFCC

MFCCs are commonly used as features in speech recognition systems, such as the systems which can automatically recognize numbers spoken into a telephone.

MFCCs are also increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.

2.5 Neural Networks

A neural network is a type of machine learning which models itself after the human brain. This creates an artificial neural network that via an algorithm allows the computer to learn by incorporating new data.

While there are plenty of artificial intelligence algorithms these days, neural networks are able to perform what has been termed deep learning. While the basic unit of the brain is the neuron,

the essential building block of an artificial neural network is a perceptron which accomplishes simple signal processing, and these are then connected into a large mesh network.

The computer with the neural network is taught to do a task by having it analyse training examples, which have been previously labelled in advance. A common example of a task for a neural network using deep learning is an object recognition task, where the neural network is presented with a large number of objects of a certain type, such as a cat, or a street sign, and the computer, by analysing the recurring patterns in the presented images, learns to categorize new images.

2.5.1 Recurrent Neural Network

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.

Recurrent neural networks were based on David Rumelhart's work in 1986. Hopfield networks - a special kind of RNN - were discovered by John Hopfield in 1982. In 1993, a neural history compressor system solved a "Very Deep Learning" task that required more than 1000 subsequent layers in an RNN unfolded in time.

2.5.1.1 Fully recurrent

Basic RNNs are a network of neuron-like nodes organized into successive layers. Each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer. Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. Nodes are either input nodes (receiving data from outside of the network), output nodes (yielding results), or hidden nodes (that modify the data enroute from input to output).

2.5.1.2 Long short-term memory

Long short-term memory (LSTM) is a deep learning system that avoids the vanishing gradient problem. LSTM is normally augmented by recurrent gates called “forget gates”. LSTM prevents back propagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier. Problem-specific LSTM-like topologies can be evolved. LSTM works even given long delays between significant events and can handle signals that mix low and high frequency components.

2.5.1.3 Activation Functions

Activation function of a node, in a Neural Network, defines the output of the node given an input or set of inputs. It is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. It also introduces nonlinearity in a model which is essential as without a non-linear activation function in the network, a Neural Network, no matter how many layers it had, would behave just like a single-layer perceptron, because summing these layers would give us just another linear function. Several activation functions exist, like sigmoid, tanh (hyperbolic tangent), ReLu, softmax etc. This project uses ReLu (Rectified Linear Unit) activation function in all the hidden layers and Softmax for the final classification dense layer. The reason for these choices was because ReLu tends to not cause vanishing gradient problem when used in very deep neural networks, as opposed to sigmoid & tanh. Furthermore, ReLu allows models to learn faster and perform better when compared to other activation functions. The outer layer could have had a sigmoid activation function, but softmax works better for a multi-class classification problem as it returns probabilities which sum up to 1, whereas sigmoid outputs in the range of 0 to 1, which is better suited for binary classification problems.

2.5.2 Backpropagation

Backpropagation is the fundamental building block of any CNN. It is used to effectively train a neural network through a method called chain rule. After each forward pass through a network, backpropagation performs a backward pass while adjusting the model’s parameters (weights and biases).

2.5.3 Loss Function

A Loss Function tells us “how good” our model is at making predictions for a given set of parameters. The cost function has its own curve and its own gradients. The slope of this curve tells us how to update our parameters to make the model more accurate. Put simply, the Loss is used to calculate the gradients, and gradients are used to update the weights of a Neural Network. In the RNNs used in this project, categorical_crossentropy loss function was used, as multi-class classification had to be done. For binary classification problems, binary_crossentropy is used.

2.5.4 Optimizers

Optimizers update the weight parameters to minimize the Loss Function. Loss function guides the terrain, telling the optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum. This project uses the Adam optimizer.

2.5.5 Overfitting

Overfitting refers to the problem of the algorithm becoming “too smart” and essentially memorizing the training dataset. This causes the algorithm to not properly predict the results when tested against a test dataset. The training accuracy becomes much, much higher than the validation accuracy. Overfitting is a huge issue that very deep Neural Networks face that can be mitigated (to an extent) by using Dropouts, Data Augmentations and other methods.

3. HISTORY

Voice recognition technology has become a part of our everyday lives, and with the increasing popularity of home assistant devices, it is more familiar than ever. Predictions indicate that voice technology will soon be common place in our workspaces, too.

Of course, voice recognition technology is nothing new. There have been key developments throughout the decades, as voice recognition evolved to today's recognisable iteration.

1950s: The first speech recognition systems were focused on numbers, not words. In 1952, Bell Laboratories designed the “Audrey” system which could recognize a single voice speaking digits aloud. Ten years later, IBM introduced “Shoebox” which understood and responded to 16 words in English.

Across the globe other nations developed hardware that could recognize sound and speech. And by the end of the year, the technology could support words with four vowels and nine consonants.

1960s: In 1961, IBM invented the Automatic Call Identification System, enabling engineers to talk to and receive spoken answers from a device, paving the first steps for the speech recognition.

1970s: Speech recognition made several meaningful advancements in this decade. This was mostly due to the US Department of Defence and DARPA. The Speech Understanding Research (SUR) program they ran was one of the largest of its kind in the history of speech recognition. Carnegie Mellon's “Harpy” speech system came from this program and was capable of understanding over 1,000 words which is about the same as a three-year-old's vocabulary.

Also significant in the '70s was Bell Laboratories' introduction of a system that could interpret multiple voices.

1980s: The '80s saw speech recognition vocabulary go from a few hundred words to several thousand words. One of the breakthroughs came from a statistical method known as

the “Hidden Markov Model (HMM)”. Instead of just using words and looking for sound patterns, the HMM estimated the probability of the unknown sounds actually being words.

IBM created a voice-activated typewriter called Tangora, which could handle a 20,000-word vocabulary under the lead of Fred Jelinek. In this era, neural networks emerged as an attractive model for Automatic speech recognition.

1990s: Speech recognition was propelled forward in the 90s in large part because of the personal computer. Faster processors made it possible for software like Dragon Dictate to become more widely used.

BellSouth introduced the voice portal (VAL) which was a dial-in interactive voice recognition system. This system gave birth to the myriad of phone tree systems that are still in existence today.

2000s: By the year 2001, speech recognition technology had achieved close to 80% accuracy. For most of the decade there weren't a lot of advancements until Google arrived with the launch of Google Voice Search. Because it was an app, this put speech recognition into the hands of millions of people. It was also significant because the processing power could be offloaded to its data centres. Not only that, Google was collecting data from billions of searches which could help it predict what a person is actually saying. At the time Google's English Voice Search System included 230 billion words from user searches.

In **2002**, Microsoft integrated speech recognition technology into their office products.

In **2005**, Giuseppe Riccardi developed Variational Bayesian (VB) to solve the problem of adaptive learning in speech recognition and proposed learning algorithm for ASR.

In **2006**, The NSA using speech recognition to isolate keywords when analysing recorded conversations.

In **2007**, Google launched GOOG-412, a telephoned directory service that paved the way for their other voice recognition products.

2010s: In 2011 Apple launched Siri which was similar to Google's Voice Search. The early part of this decade saw an explosion of other voice recognition apps. And with Amazon's Alexa, Google Home we've seen consumers becoming more and more comfortable talking to machines.

2015: Song.W & Cai.J has developed end to end speech recognition using hybrid CNN and RNN. They have used hybrid convolutional neural networks for phoneme recognition and HMM for word decoding. Their best model achieved an accuracy of 26.3% frame error on the standard core test dataset for TIMIT. Their main motto is to replace GMM-HMM based automatic speech recognition with the deep neural networks. The CNN they used consists of 4 convolutional layers. The first two layers have max pooling and the next two densely connected layers with a softmax layer as output. The activation function used was ReLu. They implemented a rectangular convolutional kernel instead of square kernel.

Between **2014** and **2017**, the race to the top for voice recognition products intensified. Microsoft introduced Cortana and Amazon gave us the Echo, a voice-controlled speaker powered by Alexa.

Today, some of the largest tech companies are competing to herald the speech accuracy title.

In **2016**, IBM achieved a word error rate of 6.9 percent.

In **2017** Microsoft usurped IBM with a 5.9 percent claim. Shortly after that IBM improved their rate to 5.5 percent. However, it is Google that is claiming the lowest rate at 4.9 percent.

4. A BRIEF SURVEY

Initial speech recognition systems were on isolated word recognition designed to perform special task. But recently in the last decade, all over the world, statistical approach (e.g., hidden Markov models (HMMs), or conditional random fields (CRF)) & ANN (artificial neural networks) based technologies are gaining attention. It is expected that human neural network like models may ultimately be able to solve the complexities of speech recognition system and provide human-like performance. A survey on major research works in the development of automatic speech recognition system in many regional languages is given below.

4.1 Survey on Tamil & Telugu Automatic Speech Recognition System

Sararwathi et al have built a language model for Tamil speech recognition system. she presents a novel technique for building a syllable based continuous speech recognizer when un annotated transcribed train data is available. Two different segmentation algorithms to segment the speech and the corresponding text into comparable syllable like unit have been presented. A group delay based two level segmentation algorithm is proposed to extract accurate syllable units from the speech data. A rule based text segmentation algorithm is used to automatically annotate the text corresponding to the speech into syllable units.

Dharun and Karnan have developed a system for voice recognition in Tamil word and numeral using Mel-Frequency Cepstral Coefficients (MFCC) and Dynamic Time Warping (DTW)

Krishnaveni present a Continuous Speech Recognition (CSR) system for Tamil language using Hidden Markov Model (HMM) approach. The most powerful and widely used MFCC feature extraction is used as a front-end for the proposed system. The monophone based acoustic model is chosen to recognize the given set of sentences from medium vocabulary. The results are found to be satisfactory with 92% of word recognition accuracy and 81% of sentence accuracy for the developed system.

Vimala C. and V. Radha present a speaker independent isolated speech recognition system for Tamil language. The experiments furnish high-quality word accuracy of 88% for trained and test utterances spoken by the speakers. The performance evaluation of the system is done based on the Word Error Rate (WER) which gives 0.88 WER for the above research work.

An Isolated Tamil word Recognizer was built using the HTK tool by A. Akila. The recognizer was trained with the grain names in Tamil spoken by 2 female speakers. The result obtained from the recognizer was having good accuracy rate for isolated Tamil words.

4.2 Survey on Assamese & Bengali Automatic Speech Recognition System

In 2010, K.K Sarma worked for the development of numeral speech recognition system for Assamese language. Gender and mood variations were given consideration during the recording of speech signals of 10 numeral digits at 8 KHz in mono channel mode.

In 2011, M. P. Sarma have proposed the design of an optimal feature extraction block and ANN based architecture for speech recognition.

M.Banerjee worked to study the effect of triphone based acoustic modelling over monophone based acoustic models in the context of continuous speech recognition in Bengali. Triphone clusters have been generated using decision tree based techniques. These triphone clusters have then been used to generate tied-state triphone based acoustic models to be used in a continuous speech recognizer.

S. Mandal, in their paper introduce the SPHINX3-based Bengali Automatic Speech Recognition (ASR) system Shruti-II and an E-mail application based on it. This ASR system converts standard Bengali continuous speech to Bengali Unicode. Due to the limited availability of access to computer, visually impaired community can use speech as an input method for various computer-based applications. This paper also demonstrates an application based on Shruti-II which is made for the visually challenged people. Those people can send E-mail by using this system.

Sunitha. K.V.N and Kalyani. N have built a speech recognition system that uses syllable as the basic unit. The experiment is conducted on CIIL Telugu corpus and achieved good results in recognizing the words that were not used for training. For training they have used 300 words and for testing they recorded 100 new words and 80% of the words were recognized correctly.

Neural network approach has been proposed by M. R. Hassan for Bengali phoneme recognition. A Bengali speech recognizer is built by training the HTK toolkit that can recognize any word in the dictionary. After acoustic analysis of speech signal, the words are recognized. Technically this work presents training the toolkit and builds a segmented speech recognizer

of Bengali. The paper contains the training procedure of the toolkit along with different steps of building a recognizer with the HTK toolkit.

4.3 Survey on Oriya & Urdu ASR System

Mohanty and Swain have made such effort for Oriya language. Mohanty and Swain [36] have come forward to apply the benefit of automatic speech recognition systems to society by developing an isolated speech recognizer for Oriya language so that visually impaired students can attempt the closed ended questions such as fill-in-blanks, dichotomous, ranking scale, multiple choice and rating scale questions, well during their exams.

Raza worked for the development of a HMM based Large Vocabulary Automatic Spontaneous Urdu speech recognition system with the help of Sphinx 3 trainer and decoder.

Sarfraz have worked on the development of LVCSR for Urdu language using CMU Sphinx Open Source Toolkit. They used a corpus of training data recorded in noisy environment so as to improve the robustness of speech recognition.

4.3 Survey on Kannada ASR System

M. A. Anusuya and K.K Katti proposed a new scheme for recognition of isolated words in Kannada Language speech, based on the Discrete Wavelet Transform (DWT) and Principal Component Analysis (PCA). Initially the Discrete wavelet transform of a speech signal is computed and then LPC coefficients are calculated.

Sarika Hegde have developed an Isolated Word Recognition (IWR) system for identification of spoken words for the database created by recording the words in Kannada language. The developed system is tested and evaluated with a performance of 79% accuracy.

M. A. Anusuya and K.K Katti used statistical method to remove the silence from the speech signal. This method is applied on vector quantization technique to identify the minimum speech patterns that are required while creating the training set of the speech samples. This paper also discusses the importance and efficiency of the algorithms used in vector. Also speech recognition accuracies for speaker dependent and speaker independent methods have been evaluated and tabulated in the tables given below. The paper shows the importance of the statistical method analysis of the signal than the normal analysis.

Shiva Kumar C worked to provide a low cost alternative for the literate deaf people. The project describes Isolated Word Recognition of Kannada Digits. The system reads the spoken speech

signal. Wavelet transform of the speech signal is taken and MFCC (Mel Frequency Cepstral Coefficients) are calculated followed by Vector Quantization. Euclidean Distance measure is used to correlate the test speech signal with pre-recorded speech signals from the speech database. The nearest match is identified and its respective text equivalent is displayed. The project is carried out for Kannada digits, which can be extended to words later. The programming is done using Matlab.

Sarika Hegde, in 2013 developed an Isolated Word Recognizer for identification of spoken words for the database created by recording the words in Kannada Language. Support Vector Machine (SVM) algorithm is used for designing the classifier model. They have analysed the variation in the performance of classification for words ending with same phonetics and found that the classification accuracy using SVM with Mel-Frequency Cepstrum Coefficients (MFCC) is good and accuracy has an affect due to the similar phonetic sounds in different words.

4.3 Survey on Gujarati ASR System

A technique for fast bootstrapping of initial phone models of a Gujarati language is presented by Himanshu N. Patel. The training data for the Gujarati language is aligned using an existing speech recognition engine for English language. This aligned data is used to obtain the initial acoustic models for the phones of the Gujarati language.

Speech recognition of Gujarati Language is presented by Patel Pravin and Harikrishna Jethva. Neural network was used for developing the system. They have used joint features derived from the modified group delay function and MFCC for Continuous speech recognition.

M.K. Deka have proposed an approach for Speech Recognition using LPCC (Linear Predictive Cepstral Coefficient) and MLP (Multilayer Perceptron) based Artificial Neural Network with respect to Assamese and Bodo Language. A new simplified approach has been made for the design and implementation of a noise robust speech recognition using Multilayer Perceptron (MLP) based Artificial Neural Network and LPC-Cepstral Coefficient. Cepstral matrices obtained via Linear Prediction Coefficient are chosen as the eligible features. Here, MLP neural network based transformation method is studied for environmental mismatch compensation.

4.3 Survey on Malayalam ASR System

Syama R. and Suma Mary Idikkula built an isolated word and speaker independent speech recognition system for Malayalam. Microsoft Visual Studio was used for compiling HTK and Active Perl as interpreter. Accuracy of this system was just 62%.

Vimal Krishnan developed a small vocabulary (5 words) speech recognition using 4 types of wavelet for feature extraction and Artificial neural network technique (ANN) is used for classification and recognition purpose. By using this method, they have achieved a recognition rate of 89%.

Raji kumar et.al have presented recognition of the isolated question words from Malayalam speech query using DWT and ANN. A recognition accuracy of 80% has been reported.

A small vocabulary speech recognizer has been developed by Anuj Mohamed and K.N Ramachandran Nair using Hidden Markov Models and MFCC. The system has produced 94.67%-word accuracy.

Sonia Sunny worked on the speech recognition of isolated 20 words of Malayalam language and reported results in three papers i.e. comparative study of two wavelet based feature extraction methods; comparison of LPC and DWT for speech recognition; and the accuracy of the speech recognizer with DWT and ANN.

4.PRE-PROCESSING WORK

Pre-processing of speech signals is considered a crucial step in the development of a robust and efficient speech recognition system. In the pre-processing of speech signals we perform some statistical outlier detection to segregate the silence/unvoiced part of the speech signal from the voiced portion.

The noise free speech signals help to determine various parameters accurately and to improve the overall system performance for subsequent offline analysis process.

The audio files collected where in different formats (such as.mp3, opus, .m4a etc.). All these files where converted to a uniform format (.wav) for making the training module.

We plotted 4 different graphs for each phone, which are:

1. **Time series graph:** Audio data can be thought of 1-D vector having numerical values for each sample. Basically these values are nothing but the amplitude of the sample at some instantaneous point of time. Time series plot is a 2D plot of these numerical values with respect to time. This plot gives the insights of the visual structure of the audio files, the unvoiced data, and similarity and dissimilarity can be inferred from this plot. This helped us to find the amplitude value below which the audio waves are discarded and not used as they don't have any useful information.

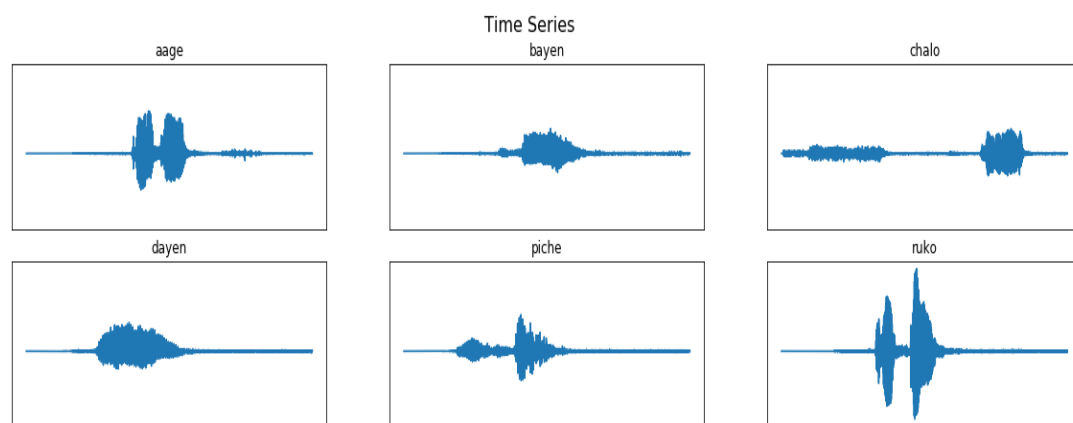


Fig 2(a) Time Series plot of the selected phones before processing

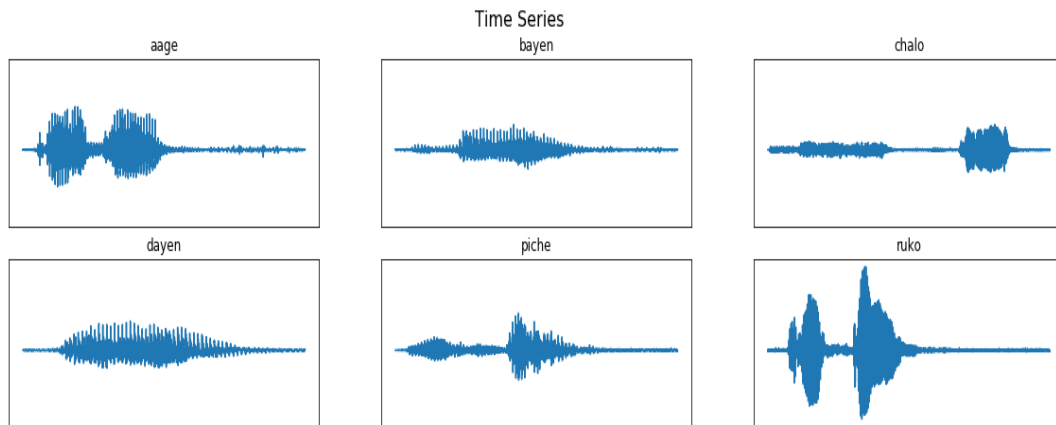


Fig 2(b) Time Series plot of selected phones after processing.

2. **Fast Fourier transform:** Fast Fourier Transform basically transform the signal from time domain to frequency domain such that the constituent frequency of a signal can be fetched. The higher frequency components of the wave basically consist of noise in the audio. Noise are the unwanted waves that distort the signal.

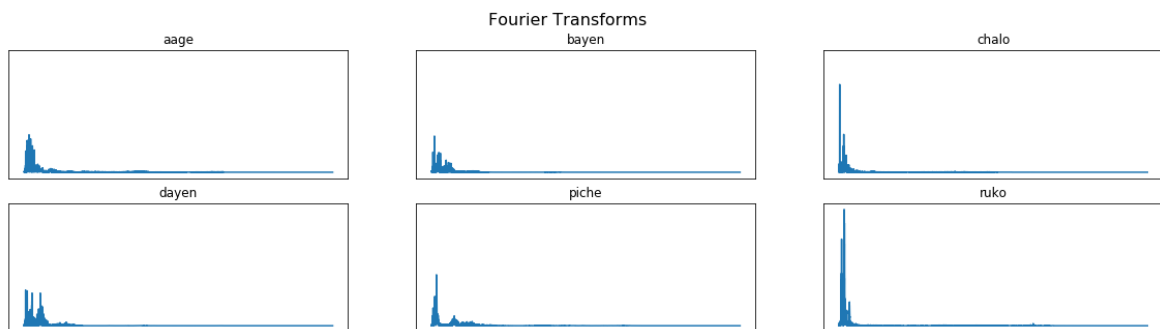


Fig 2(c). FFT plot of the selected phones before pre-processing

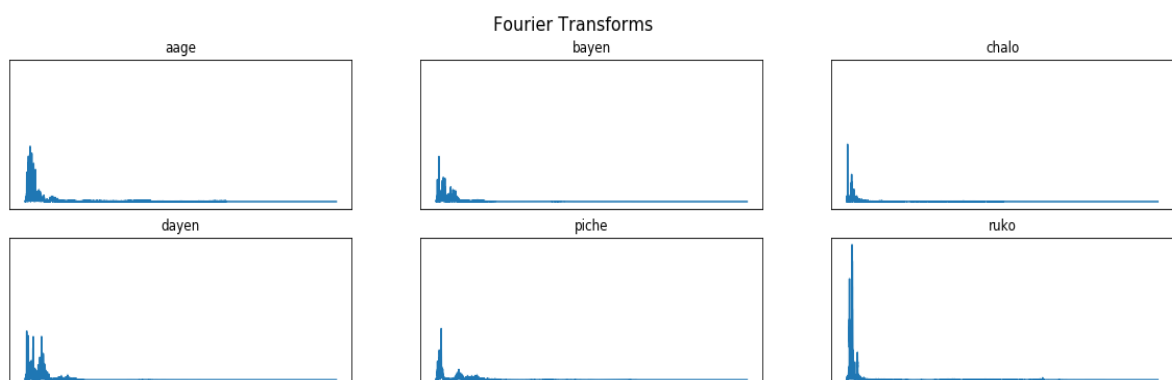


Fig 2(d). FFT plot of the selected phones after pre-processing

3. **MFCC graph:** MFCC gives the co-efficient by applying the Mel-scale to the audio signal transformed after DFT as the human ear perceives the frequencies in a similar way. These co-efficient help in analysing the voice based on the movement of human vocal tract.

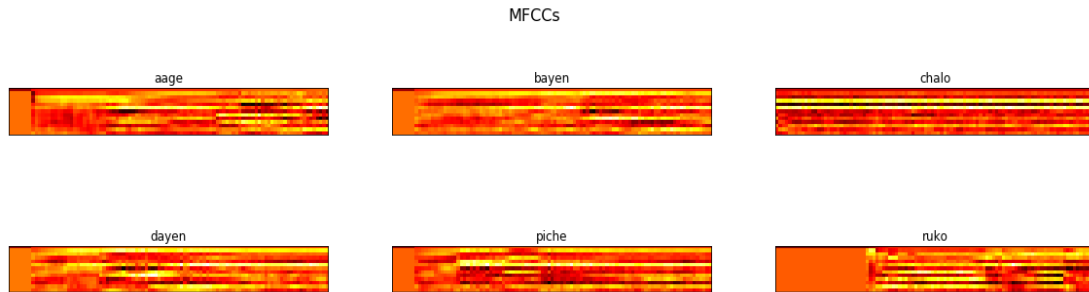


Fig 2(e). MFCC plot of the selected phones before pre-processing

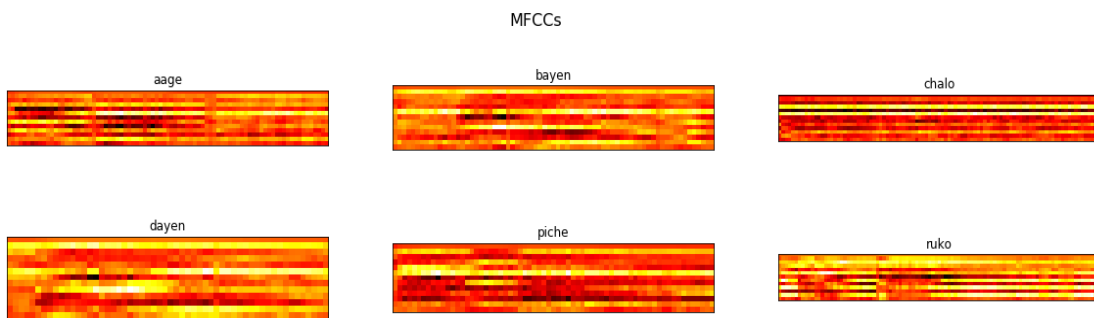


Fig 2(f). MFCC plot of the selected phones after pre-processing

After taking the insights it was seen that all the audio samples are not sampled at the same frequency instead they have varying sampling frequency, thus all the audio files were sampled at the same frequency in this step.

After that the unvoiced signals having low amplitude were discarded making an envelope over the audio of real part of the signal.

5. PROPOSED APPROACH

Voice recognition is a technique in computing technology by which specialized software and systems are created to identify, distinguish and authenticate the voice of an individual speaker.

Although there are many applications developed in the voice recognition domain like Siri & Google assistant. But these applications can work in English language only.

Besides that, many works are done or are under process in Indian regional languages but again making a complex system to attain a result somewhat comparable to a Speech Recognition System in English is a difficult task.

Thus the purpose of this project will not cover the entire language but will focus on few phones of that language.

The purpose of selecting this project is that we want to develop a voice recognition system which can work in a regional language (Hindi). This project will focus only on the six phones of the language (that are आगे (āge), पीछे (peechhe), बाएँ (bāen), दाएँ (dāen), चलो (chalo), रूको (ruko)). Basically the system will predict any of the audio phones from the mentioned above and will give the name of phone as a text.

Further this recognition system will be used to operate an Arduino based vehicle which will be fed with the text received from the recognition system and it will take action accordingly.

ALGORITHM_1: SPEECH_RECOGNITION_SYSTEM

Input: Voice Sample

Output: A system which can predict these phones.

Step_1: Collecting voice of different peoples for the required phones & save them in a directory to make a repository.

Step_2: Pre-process those audio files by removing unwanted voice signals from each file.

Step_3: Split the pre-processed audio files into two parts (files for training & testing).

Step_4: Feed the training part of audio files into the training module so that our model will be trained to recognize those phones.

Step_5: Feed the audio files in testing folder into the trained module, and test it for accuracy.

Step_6: Feed the Voice Sample to the trained model, a vector having probability value for each of the phones is generated.

Step_6: Use the vector index with maximum probability to find the name of the phone.

Step_7: Display the name of the phone.

Step_8: Stop

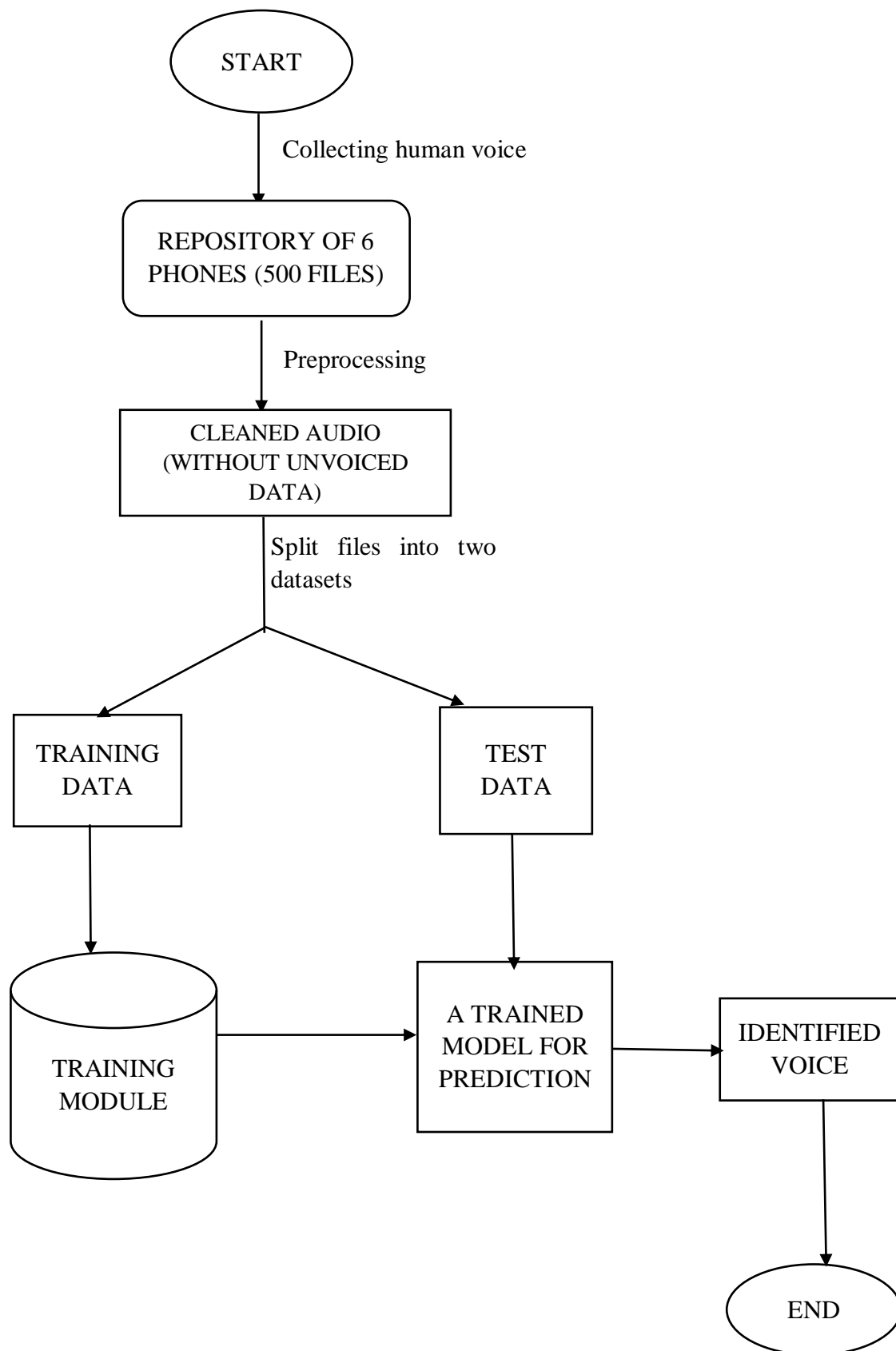


Fig.3(a) Approach for Making a Speech Recognition System

ALGORITHM_2: PREPROCESSING_STEPS

Input: The manually collected repository of audio files.

Output: Audio files after removing all unwanted signals.

Step_1: Convert all the audio files in the same format(.wav)

Step_2: Plot graphs to gather insights about the phones. These are Time-Series, MFCC, FFT plots

Step_3: Analysing plots, make an envelope over the audio to discard all unwanted, unvoiced part of the audio.

Step_4: Save the cleaned audio files in the repository.

Step_5: Stop.

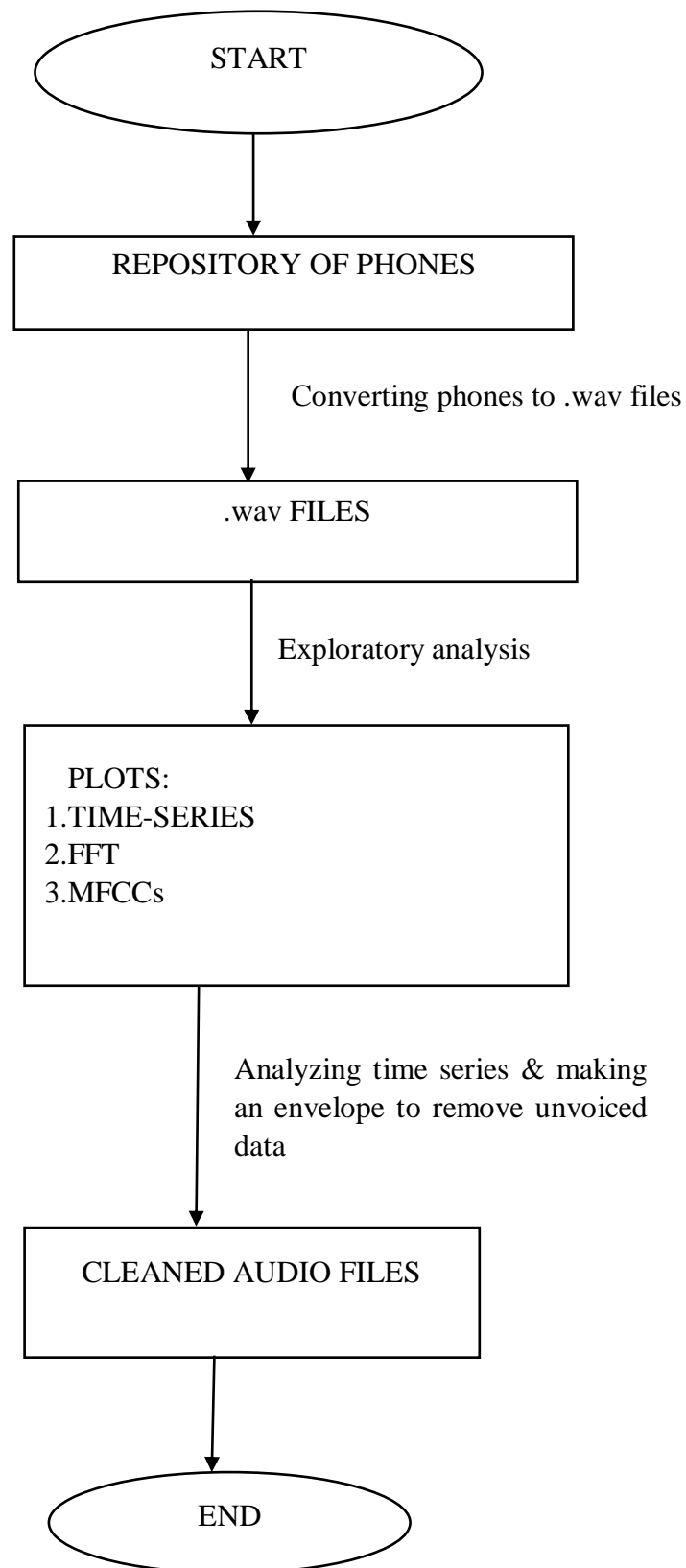


Fig 3(b). Preprocessing Steps

ALGORITHM_3: TRAINING_MODULE

Input: Repository of pre-processed audio files.

Output: A supervised trained model which can further be used for testing.

Step_1: Initialise Sample_number with the number of files in the repository

Step_2: For I=1 to Sample_number, repeat

Step_2.1: Random selection of an audio file from the repository of pre-processed audio files.

Step_2.2: Random selection of 1/10th of a second from that random audio file.

Step_2.3: Extracting the mfcc features for that 1/10th second of audio.

Step_2.4: Append those features in a 2 dimensional vector.

Step_2.5: Append the name of the phone in a vector

Step_3: Feed the recurrent neural network of our training module with the two vectors formed in Step_2.4 and Step_2.5

Step_4: Save the trained Recurrent Neural Network model

Step_5: Stop

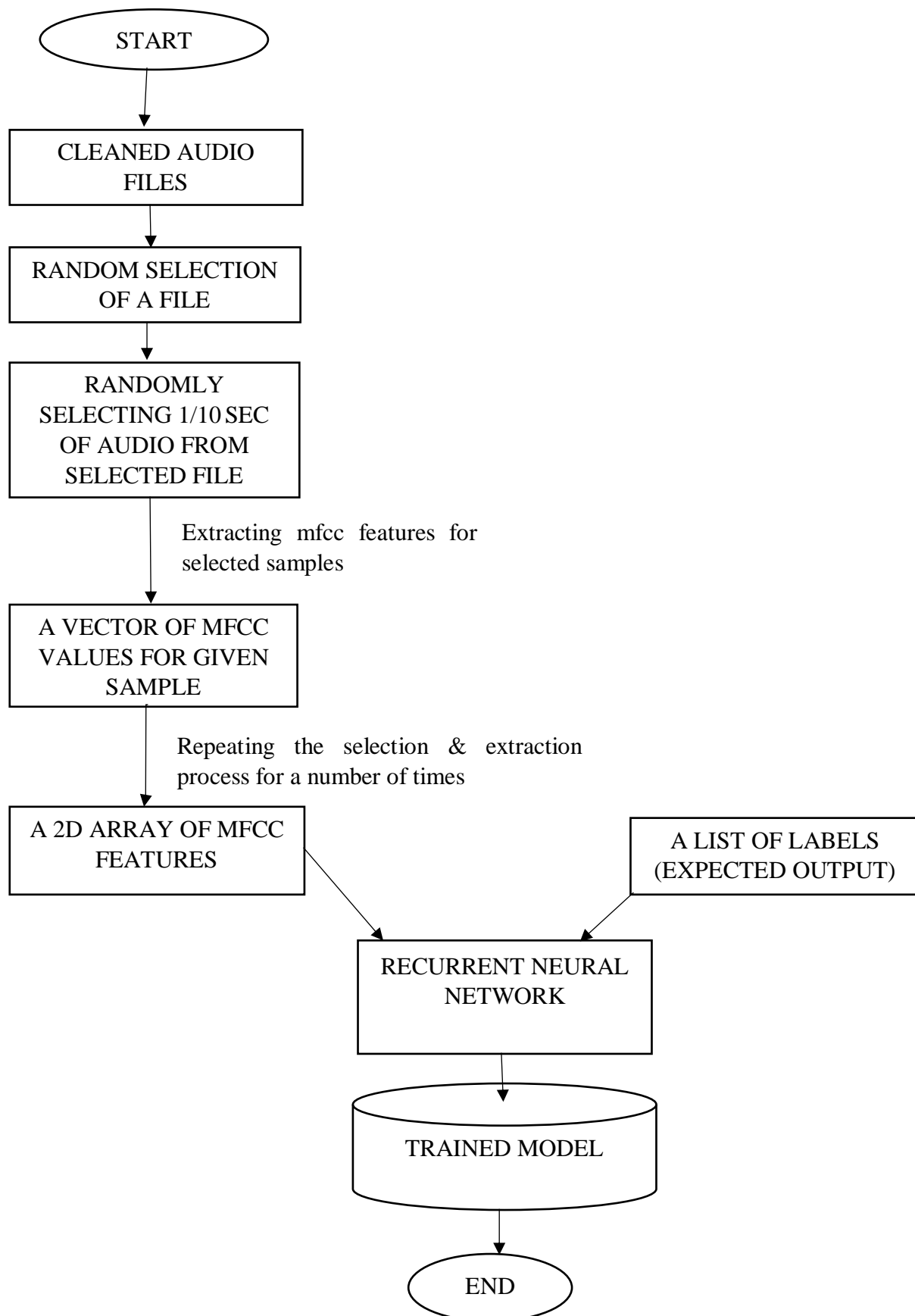


Fig 3(c). Training Module

ALGORITHM_4: ONSPOT_TESTING

Input: Human voice for a particular phone among those six phones through microphone.

Output: Name of the phone predicted by the Speech Recognition System.

Step_1: Save the voice taken as input in .wav format.

Step_2: Perform pre-processing and clean the voice.

Step_3: Feed the cleaned voice into the trained model.

Step_4: Test for closeness of the voice with one of the phone among the six used for training.

Step_5: Display the phone name having maximum probability.

Step_6: Stop

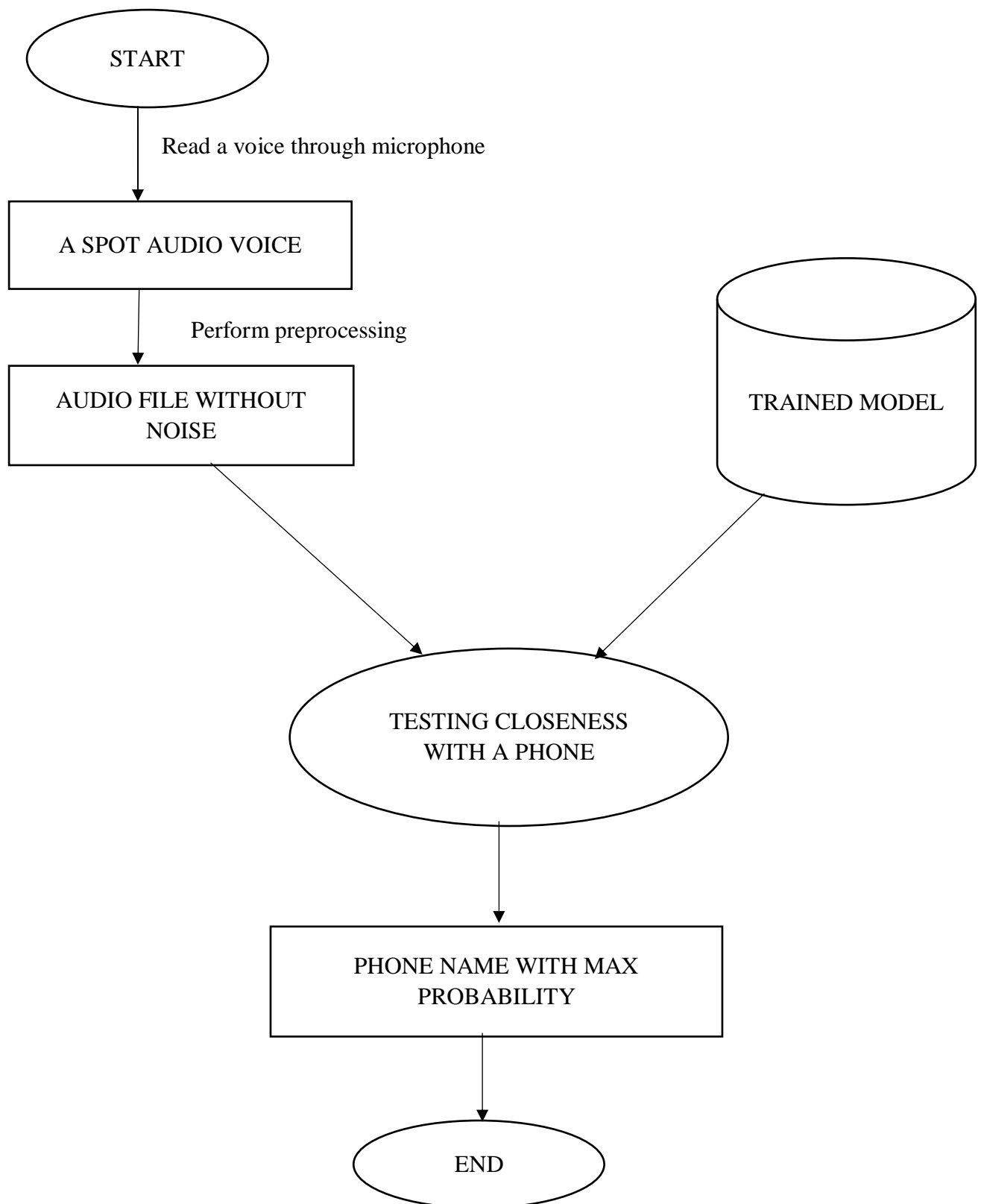


Fig 3(d). On-spot Testing

6. RESULT AND CORRESPONDING EVALUATION

Just by tweaking the features selecting method and changing envelope threshold value, we trained five models with same recurrent network.

6.1 Common Terms Used

1) **Feature Selection:** The MFCC feature selection selects m^* number of mfcc filters used (20 for our case), where m is dependent on the sampling frequency, thus signals with different frequencies have different shapes. Thus an arbitrary value m^* between the maximum value and minimum value of m considering all samples was taken as fixed as the shape to be fed to neural network. If the row size, m of the given sample is less than the m^* , then (m^*-m) zero valued rows were padded and if the audio sample contains $m > m^*$, then 2 feature vectors were created instead of one. One of them consists of first m^* rows and the other consists of the remaining rows padded with zero valued rows.

2) **Number of Samples** = No. of times any random selection was made from the repository and used for training

Normal number of samples simply means the number of audio files in the repository i.e. 390

Increased number of Samples means number of Audio files in the repository (390) * 20 (Arbitrarily chosen)

3) **Threshold** = A value below which all audio features are discarded. Only features above are taken

4) Training Done on 390 Audio Samples

5) Testing Done on 124 Audio Samples

6) The RNN model structure was similar for all the given models following a sequential model having two LSTM (Long Short Term Memory) layer to protect the model having a vanishing gradient problem, one Dropout layer, four Time Distributed Layer with Relu activation and one flatten layer. Finally, one dense layer was used with softmax activation.

6.2 Discussion on the Models Trained

6.2.1 Model 1- Feature vector of (5*20) Dimension, Threshold 0.005

First model was built just taking the sample size as the number of audio files in the system i.e. 390 files and giving a threshold of 0.005 for the envelope in the pre-processing step.

The feature selection was done using randomly selecting 1/10 second part of a randomly selected audio file and applying 20 mfcc filters over that. Thus the feature vector for a sample would be of shape $m \times 20$ where m varies sample to sample. For this case the value of m was set as 5. Thus if any sample has less than 5 number of rows 0 were padded to it and it has greater than 5 rows it was split into two parts.

Model: "sequential_5"		
Layer (type)	Output Shape	Param #
lstm_9 (LSTM)	(None, 5, 128)	76288
lstm_10 (LSTM)	(None, 5, 128)	131584
dropout_5 (Dropout)	(None, 5, 128)	0
time_distributed_17 (TimeDis	(None, 5, 64)	8256
time_distributed_18 (TimeDis	(None, 5, 32)	2080
time_distributed_19 (TimeDis	(None, 5, 16)	528
time_distributed_20 (TimeDis	(None, 5, 8)	136
flatten_5 (Flatten)	(None, 40)	0
dense_25 (Dense)	(None, 6)	246
Total params: 219,118		
Trainable params: 219,118		
Non-trainable params: 0		

Fig 4.1(a) RNN model description of layers for Model 1

```

Train on 569 samples, validate on 143 samples
Epoch 1/10
569/569 [=====] - 2s 3ms/step - loss: 1.7817 - acc: 0.1828 - val_loss: 1.7905 - val_acc: 0.1538

Epoch 00001: val_acc improved from -inf to 0.15385, saving model to model\rnn5_res1.model
Epoch 2/10
569/569 [=====] - 0s 682us/step - loss: 1.7369 - acc: 0.2004 - val_loss: 1.7869 - val_acc: 0.1608

Epoch 00002: val_acc improved from 0.15385 to 0.16084, saving model to model\rnn5_res1.model
Epoch 3/10
569/569 [=====] - 0s 661us/step - loss: 1.6749 - acc: 0.2882 - val_loss: 1.7323 - val_acc: 0.1678

Epoch 00003: val_acc improved from 0.16084 to 0.16783, saving model to model\rnn5_res1.model
Epoch 4/10
569/569 [=====] - 0s 666us/step - loss: 1.5836 - acc: 0.3322 - val_loss: 1.7194 - val_acc: 0.1678

Epoch 00004: val_acc did not improve from 0.16783
Epoch 5/10
569/569 [=====] - 0s 669us/step - loss: 1.4656 - acc: 0.3884 - val_loss: 1.6995 - val_acc: 0.2238

Epoch 00005: val_acc improved from 0.16783 to 0.22378, saving model to model\rnn5_res1.model
Epoch 6/10
569/569 [=====] - 0s 684us/step - loss: 1.3549 - acc: 0.4271 - val_loss: 1.6566 - val_acc: 0.3007

Epoch 00006: val_acc improved from 0.22378 to 0.30070, saving model to model\rnn5_res1.model
Epoch 7/10
569/569 [=====] - 0s 676us/step - loss: 1.2798 - acc: 0.4429 - val_loss: 1.6735 - val_acc: 0.2937

Epoch 00007: val_acc did not improve from 0.30070
Epoch 8/10
569/569 [=====] - 0s 854us/step - loss: 1.2067 - acc: 0.4675 - val_loss: 1.6909 - val_acc: 0.3357

Epoch 00008: val_acc improved from 0.30070 to 0.33566, saving model to model\rnn5_res1.model
Epoch 9/10
569/569 [=====] - 0s 669us/step - loss: 1.1401 - acc: 0.5026 - val_loss: 1.6503 - val_acc: 0.3566

Epoch 00009: val_acc improved from 0.33566 to 0.35664, saving model to model\rnn5_res1.model
Epoch 10/10
569/569 [=====] - 0s 675us/step - loss: 1.0730 - acc: 0.5448 - val_loss: 1.7869 - val_acc: 0.3916

Epoch 00010: val_acc improved from 0.35664 to 0.39161, saving model to model\rnn5_res1.model

```

Fig 4.1(b) Accuracy metrics for Model 1

The training accuracy for this model is (54-56) % while validation accuracy is 40%.

Using this model for on-spot testing didn't gave encouraging results as it failed quite a number of times.

Comment: A not so good model for prediction for stored file in the system but can't be used for our purpose. Gives a conclusion that either processed audio still contains noise or features are not properly selected.

6.2.2 Model 2- Feature Vector of (5*20) Dimension, Threshold= 0.009

This was also trained on the same number of samples i.e. 390 audio files but during pre-processing of the file the envelope threshold was chosen as 0.009.

From the selected file again only 1/10 second of the part was used for extracting the feature vector.

Features selection was same as model 1, i.e. 5rows* 20 columns vector for a sample, if size was greater sample was divided into two.

Model: "sequential_8"

Layer (type)	Output Shape	Param #
lstm_15 (LSTM)	(None, 5, 128)	76288
lstm_16 (LSTM)	(None, 5, 128)	131584
dropout_8 (Dropout)	(None, 5, 128)	0
time_distributed_29 (TimeDis	(None, 5, 64)	8256
time_distributed_30 (TimeDis	(None, 5, 32)	2080
time_distributed_31 (TimeDis	(None, 5, 16)	528
time_distributed_32 (TimeDis	(None, 5, 8)	136
flatten_8 (Flatten)	(None, 40)	0
dense_40 (Dense)	(None, 6)	246

Total params: 219,118
 Trainable params: 219,118
 Non-trainable params: 0

Fig 4.2(a) RNN model description of layers for Model 2

```

Train on 411 samples, validate on 103 samples
Epoch 1/10
411/411 [=====] - 2s 5ms/step - loss: 1.7883 - acc: 0.1606 - val_loss: 1.7789 - val_acc: 0.1748

Epoch 00001: val_acc improved from -inf to 0.17476, saving model to model\rnn5_res1.model
Epoch 2/10
411/411 [=====] - 0s 929us/step - loss: 1.7613 - acc: 0.2749 - val_loss: 1.7420 - val_acc: 0.3592

Epoch 00002: val_acc improved from 0.17476 to 0.35922, saving model to model\rnn5_res1.model
Epoch 3/10
411/411 [=====] - 0s 725us/step - loss: 1.7066 - acc: 0.3236 - val_loss: 1.6817 - val_acc: 0.3107

Epoch 00003: val_acc did not improve from 0.35922
Epoch 4/10
411/411 [=====] - 1s 1ms/step - loss: 1.6103 - acc: 0.3552 - val_loss: 1.6341 - val_acc: 0.3495

Epoch 00004: val_acc did not improve from 0.35922
Epoch 5/10
411/411 [=====] - 0s 783us/step - loss: 1.4903 - acc: 0.4209 - val_loss: 1.5986 - val_acc: 0.4078

Epoch 00005: val_acc improved from 0.35922 to 0.40777, saving model to model\rnn5_res1.model
Epoch 6/10
411/411 [=====] - 0s 973us/step - loss: 1.3819 - acc: 0.4866 - val_loss: 1.5428 - val_acc: 0.3883

Epoch 00006: val_acc did not improve from 0.40777
Epoch 7/10
411/411 [=====] - 0s 888us/step - loss: 1.2443 - acc: 0.4988 - val_loss: 1.5889 - val_acc: 0.4078

Epoch 00007: val_acc did not improve from 0.40777
Epoch 8/10
411/411 [=====] - 0s 704us/step - loss: 1.1484 - acc: 0.5645 - val_loss: 1.6200 - val_acc: 0.3981ETA: 0s - loss: 1.1139 - acc: 0.5938

Epoch 00008: val_acc did not improve from 0.40777
Epoch 9/10
411/411 [=====] - 0s 679us/step - loss: 1.0489 - acc: 0.5888 - val_loss: 1.7677 - val_acc: 0.4078

Epoch 00009: val_acc did not improve from 0.40777
Epoch 10/10
411/411 [=====] - 0s 669us/step - loss: 0.9460 - acc: 0.6350 - val_loss: 1.8836 - val_acc: 0.3981

Epoch 00010: val_acc did not improve from 0.40777

```

Fig 4.2(b) Accuracy metrics for Model 2

The training accuracy for this model was around 65% with a validation accuracy of 45%

Comment: Huge Threshold value for the envelope decreases the amount of information from the phones as this amount of threshold not only removes unvoiced data but may remove some

part of useful information. This amount of huge threshold gave the time series structure of both dayen and bayen phones almost similar, i.e. the phones could have been misunderstood as the same. From this it was concluded that something with the feature selection method or with the amount of data has to be done to improve the accuracy.

6.2.3 Model 3- Feature Vector of (5*20) Dimension, Sample Size is Multiplied by 20, Threshold = 0.005

Since earlier results failed, one reason could have been the number of samples. Thus from here the number of samples were increased by simply multiplying it by 20.

Again only 1/10 of a second of audio file was randomly selected to extract the feature from the audio files.

Keeping the feature selection method to be the same i.e. 5row*20 col feature vector.

Model: "sequential_7"		
Layer (type)	Output Shape	Param #
=====		
lstm_13 (LSTM)	(None, 5, 128)	76288
lstm_14 (LSTM)	(None, 5, 128)	131584
dropout_7 (Dropout)	(None, 5, 128)	0
time_distributed_25 (TimeDis	(None, 5, 64)	8256
time_distributed_26 (TimeDis	(None, 5, 32)	2080
time_distributed_27 (TimeDis	(None, 5, 16)	528
time_distributed_28 (TimeDis	(None, 5, 8)	136
flatten_7 (Flatten)	(None, 40)	0
dense_35 (Dense)	(None, 6)	246
=====		
Total params: 219,118		
Trainable params: 219,118		
Non-trainable params: 0		

Fig 4.3(a) RNN model description of layers for Model 3


```

Train on 11363 samples, validate on 2841 samples
Epoch 1/10
11363/11363 [=====] - 11s 992us/step - loss: 1.5558 - acc: 0.3326 - val_loss: 1.2910 - val_acc: 0.4565

Epoch 00001: val_acc improved from -inf to 0.45653, saving model to model\rnn5_res1.model
Epoch 2/10
11363/11363 [=====] - 9s 800us/step - loss: 1.2206 - acc: 0.4885 - val_loss: 1.1189 - val_acc: 0.5304] - ETA: 8s - loss: 1.3320 - acc: 0.4306 - ETA: 3s - loss: 1.2454 - acc: 0.4785 -
ETA: 1s - loss: 1.2310 - acc: 0.4855 - ETA: 1s - loss: 1.2292 - acc: 0.485910144/11363 [=====] - ETA: 0s - loss: 1.2279 - acc: 0.4850

Epoch 00002: val_acc improved from 0.45653 to 0.53045, saving model to model\rnn5_res1.model
Epoch 3/10
11363/11363 [=====] - 9s 796us/step - loss: 1.0640 - acc: 0.5667 - val_loss: 1.0179 - val_acc: 0.5713.] - ETA: 6s - loss: 1.1029 - acc: 0.5453

Epoch 00003: val_acc improved from 0.53045 to 0.57128, saving model to model\rnn5_res1.model
Epoch 4/10
11363/11363 [=====] - 9s 792us/step - loss: 0.9456 - acc: 0.6174 - val_loss: 0.9406 - val_acc: 0.6132] - ETA: 8s - loss: 0.8958 - acc: 0.6354 4224/11363
[=====] - ETA: 4s - loss: 0.9683 - acc: 0.6044

Epoch 00004: val_acc improved from 0.57128 to 0.61316, saving model to model\rnn5_res1.model
Epoch 5/10
11363/11363 [=====] - 8s 672us/step - loss: 0.8525 - acc: 0.6580 - val_loss: 0.8753 - val_acc: 0.65518/11363 [=====] - ETA: 4s - loss: 0.8519 - acc: 0.6618
6048/11363 [=====] - ETA: 3s - loss: 0.8551 - acc: 0.6569 7936/11363 [=====] - ETA: 2s - loss: 0.8512 - acc: 0.6583

Epoch 00005: val_acc improved from 0.61316 to 0.65505, saving model to model\rnn5_res1.model
Epoch 6/10
11363/11363 [=====] - 8s 666us/step - loss: 0.7659 - acc: 0.6923 - val_loss: 0.8040 - val_acc: 0.6906] - ETA: 4s - loss: 0.7807 - acc: 0.6849 8160/11363
[=====] - ETA: 1s - loss: 0.7762 - acc: 0.6868 - ETA: 1s - loss: 0.7696 - acc: 0.6892

Epoch 00006: val_acc improved from 0.65505 to 0.69060, saving model to model\rnn5_res1.model
Epoch 7/10
11363/11363 [=====] - 8s 666us/step - loss: 0.6815 - acc: 0.7279 - val_loss: 0.7716 - val_acc: 0.7075

Epoch 00007: val_acc improved from 0.69060 to 0.70750, saving model to model\rnn5_res1.model
Epoch 8/10
11363/11363 [=====] - 8s 672us/step - loss: 0.6186 - acc: 0.7589 - val_loss: 0.7609 - val_acc: 0.7346] - ETA: 5s - loss: 0.6147 - acc: 0.7547 4640/11363
[=====] - ETA: 4s - loss: 0.6164 - acc: 0.7560

Epoch 00008: val_acc improved from 0.70750 to 0.73460, saving model to model\rnn5_res1.model
Epoch 9/10
11363/11363 [=====] - 8s 664us/step - loss: 0.5801 - acc: 0.7760 - val_loss: 0.7151 - val_acc: 0.7483

Epoch 00009: val_acc improved from 0.73460 to 0.74833, saving model to model\rnn5_res1.model
Epoch 10/10
11363/11363 [=====] - 8s 697us/step - loss: 0.5196 - acc: 0.8030 - val_loss: 0.7582 - val_acc: 0.7483] - ETA: 6s - loss: 0.4747 - acc: 0.8344 - ETA: 6s - loss: 0.5272 - acc: 0.8045 -
ETA: 1s - loss: 0.5190 - acc: 0.8040

Epoch 00010: val_acc did not improve from 0.74833

```

Fig 4.3(b) Accuracy metrics for Model 3

The training and validation accuracy both were quite near to 80%.

Comment: A Good model for making predictions on the files stored in the repository but fails while making instant predictions. The increase in accuracy made sure that somewhere the amount of audio files in the repository were not sufficient to get the expected results. Thus the only aspect which can be played with is the feature selection i.e. changing the value of m^* may give better results.

6.2.4 Model 4-Feature Vector of (9*20) Dimension, Sample Size is Multiplied by 20, Threshold = 0.005

As the increased number of samples gave better results, for this approach also increased number of samples were used with the same envelope threshold of 0.005 but here the value of m^* for the feature vector was changed from 5 to 9. Thus the shape of the feature vector was 9*20. 1/10 second of audio was again chosen from the files to extract these features.

Model: "sequential_13"

Layer (type)	Output Shape	Param #
lstm_25 (LSTM)	(None, 9, 128)	76288
lstm_26 (LSTM)	(None, 9, 128)	131584
dropout_13 (Dropout)	(None, 9, 128)	0
time_distributed_49 (TimeDis	(None, 9, 64)	8256
time_distributed_50 (TimeDis	(None, 9, 32)	2080
time_distributed_51 (TimeDis	(None, 9, 16)	528
time_distributed_52 (TimeDis	(None, 9, 8)	136
flatten_13 (Flatten)	(None, 72)	0
dense_65 (Dense)	(None, 6)	438

Total params: 219,310
 Trainable params: 219,310
 Non-trainable params: 0

Fig 4.4(a) RNN model description of layers for Model 4

```

Epoch 00001: val_acc improved from -inf to 0.45992, saving model to model\rnn5_res1.model
Epoch 2/10
5685/5685 [=====] - 7s 1ms/step - loss: 1.2324 - acc: 0.4753 - val_loss: 1.1233 - val_acc: 0.5183] - ETA: 1s - loss: 1.2555 - acc: 0.4693

Epoch 00002: val_acc improved from 0.45992 to 0.51828, saving model to model\rnn5_res1.model
Epoch 3/10
5685/5685 [=====] - 7s 1ms/step - loss: 1.0850 - acc: 0.5374 - val_loss: 1.0567 - val_acc: 0.5577] - ETA: 4s - loss: 1.1351 - acc: 0.5064 - ETA: 0s

Epoch 00003: val_acc improved from 0.51828 to 0.55767, saving model to model\rnn5_res1.model
Epoch 4/10
5685/5685 [=====] - 8s 1ms/step - loss: 0.9711 - acc: 0.5945 - val_loss: 0.9673 - val_acc: 0.5999.] - ETA: 0s - loss: 0.9736 - acc: 0.5915

Epoch 00004: val_acc improved from 0.55767 to 0.59986, saving model to model\rnn5_res1.model
Epoch 5/10
5685/5685 [=====] - 8s 1ms/step - loss: 0.8691 - acc: 0.6480 - val_loss: 0.9087 - val_acc: 0.6294] - ETA: 2s - loss: 0.8807 - acc: 0.6410

Epoch 00005: val_acc improved from 0.59986 to 0.62940, saving model to model\rnn5_res1.model
Epoch 6/10
5685/5685 [=====] - ETA: 0s - loss: 0.7587 - acc: 0.6859 512/5685 [=] - ETA: 7s - loss: 0.7652 - acc: 0.67773328/5685
ETA: 2s - loss: 0.7558 - acc: 0.68034608/5685 [=====] - ETA: 1s - loss: 0.7618 - acc: 0.6821 - ETA: 0s - loss: 0.7591 - acc: 0.6855 - 7s 1ms/step -
val_loss: 0.8644 - val_acc: 0.6667

Epoch 00006: val_acc improved from 0.62940 to 0.66667, saving model to model\rnn5_res1.model
Epoch 7/10
5685/5685 [=====] - 8s 1ms/step - loss: 0.6887 - acc: 0.7223 - val_loss: 0.8242 - val_acc: 0.697612/5685 [=====] - ETA: 0s

Epoch 00007: val_acc improved from 0.66667 to 0.69761, saving model to model\rnn5_res1.model
Epoch 8/10
5685/5685 [=====] - 7s 1ms/step - loss: 0.6063 - acc: 0.7567 - val_loss: 0.7987 - val_acc: 0.7124] - ETA: 4s - loss: 0.6134 - acc: 0.75963392/5685
ETA: 2s - loss: 0.6105 - acc: 0.7559

Epoch 00008: val_acc improved from 0.69761 to 0.71238, saving model to model\rnn5_res1.model
Epoch 9/10
5685/5685 [=====] - 7s 1ms/step - loss: 0.5409 - acc: 0.7884 - val_loss: 0.7674 - val_acc: 0.7342 ETA: 4s - loss: 0.5555 - acc: 0.7917

Epoch 00009: val_acc improved from 0.71238 to 0.73418, saving model to model\rnn5_res1.model
Epoch 10/10
5685/5685 [=====] - 7s 1ms/step - loss: 0.5176 - acc: 0.8019 - val_loss: 0.7659 - val_acc: 0.7307] - ETA: 2s - loss: 0.5000 - acc: 0.8050

Epoch 00010: val_acc did not improve from 0.73418

```

Fig 4.4(b) Accuracy metrics for Model 4

Training accuracy for this approach was (81-85) % with a validation accuracy of 75% however shows some acceptable results for the instant testing.

Comment: A good model from accuracy point of view and also gives good result for on-spot prediction. Increasing the number of original sample instead of just multiplying it by 20 as multiplying just make the system to learn from the same samples only a number of times but

increasing the size of repository will give the system a better chance to learn some new aspects of these phones.

6.2.5 Model 5-Feature Vector of (172*20) Dimension, Sample Size is Multiplied by 20, Threshold = 0.005

Again taking the increased number of samples and same envelope threshold of 0.005, the model was trained but instead of taking 1/10 second part of the sample the audio as a whole was taken for extracting the features, thus the size of feature vector was huge compared to all above models. The feature vector here is 172rows * 20 columns.

Model: "sequential_12"		
Layer (type)	Output Shape	Param #
=====		
lstm_23 (LSTM)	(None, 172, 128)	76288
lstm_24 (LSTM)	(None, 172, 128)	131584
dropout_12 (Dropout)	(None, 172, 128)	0
time_distributed_45 (TimeDis	(None, 172, 64)	8256
time_distributed_46 (TimeDis	(None, 172, 32)	2080
time_distributed_47 (TimeDis	(None, 172, 16)	528
time_distributed_48 (TimeDis	(None, 172, 8)	136
flatten_12 (Flatten)	(None, 1376)	0
dense_60 (Dense)	(None, 6)	8262
=====		
Total params: 227,134		
Trainable params: 227,134		
Non-trainable params: 0		

Fig 4.5(a) RNN model description of layers for Model 5

```

(7910, 172, 20)
Train on 6328 samples, validate on 1582 samples
Epoch 1/10
6328/6328 [=====] - 113s 18ms/step - loss: 0.7087 - acc: 0.7252 - val_loss: 0.1994 - val_acc: 0.9393 ETA: 41s - loss: 0.9745 - acc: 0.61394416/6328 [=====] - .....] -
ETA: 32s - loss: 0.8918 - acc: 0.6490 - ETA: 20s - loss: 0.8169 - acc: 0.6792 - ETA: 18s - loss: 0.8001 - acc: 0.6864

Epoch 00001: val_acc improved from -inf to 0.93932, saving model to model\rnn5_res1.model
Epoch 2/10
6328/6328 [=====] - 108s 17ms/step - loss: 0.1158 - acc: 0.9644 - val_loss: 0.1154 - val_acc: 0.9697 - ETA: 35s - loss: 0.1453 - acc: 0.9532

Epoch 00002: val_acc improved from 0.93932 to 0.96966, saving model to model\rnn5_res1.model
Epoch 3/10
6328/6328 [=====] - 114s 18ms/step - loss: 0.0903 - acc: 0.9771 - val_loss: 0.0367 - val_acc: 0.9861 - ETA: 1:22 - loss: 0.0627 - acc: 0.98164736/6328 [=====] - .....] -
ETA: 25s - loss: 0.1012 - acc: 0.97405440/6328 [=====] - .....] - ETA: 15s - loss: 0.0983 - acc: 0.9746 - ETA: 8s - loss: 0.0946 - acc: 0.9756

Epoch 00003: val_acc improved from 0.96966 to 0.98609, saving model to model\rnn5_res1.model
Epoch 4/10
6328/6328 [=====] - 113s 18ms/step - loss: 0.0000 - acc: 0.9981 - val_loss: 8.9059e-04 - val_acc: 1.00006328 [=====>.....] - ETA: 1:28 - loss: 0.0203 - acc:
0.99501152/6328 [=====>.....] - ETA: 1:24 - loss: 0.0205 - acc: 0.9948 - ETA: 1:23 - loss: 0.0198 - acc: 0.99525088/6328 [=====>.....] - ETA: 7s - loss: 0.0085 - acc:
0.9980

Epoch 00004: val_acc improved from 0.98609 to 1.00000, saving model to model\rnn5_res1.model
Epoch 5/10
6328/6328 [=====] - 111s 17ms/step - loss: 6.7564e-04 - acc: 1.0000 - val_loss: 3.1389e-04 - val_acc: 1.0000121 - loss: 9.9939e-04 - acc: 1.00003456/6328
[=====] - .....] - ETA: 46s - loss: 7.7590e-04 - acc: 1.00004256/6328 [=====] - .....] - ETA: 33s - loss: 7.8359e-04 - acc: 1.0000

Epoch 00005: val_acc did not improve from 1.00000
Epoch 6/10
6328/6328 [=====] - 109s 17ms/step - loss: 3.1713e-04 - acc: 1.0000 - val_loss: 1.5977e-04 - val_acc: 1.0000696/6328 [=====>.....] - ETA: 1:13 - loss: 3.7365e-04 -
acc: 1.0000 - ETA: 34s - loss: 3.5575e-04 - acc: 1.00006112/6328 [=====>.....] - ETA: 3s - loss: 3.2250e-04 - acc: 1.0000

Epoch 00006: val_acc did not improve from 1.00000
Epoch 7/10
6328/6328 [=====] - 110s 17ms/step - loss: 1.7150e-04 - acc: 1.0000 - val_loss: 1.0229e-04 - val_acc: 1.00007A: 1:06 - loss: 2.1121e-04 - acc: 1.00002880/6328
[=====>.....] - ETA: 56s - loss: 1.9586e-04 - acc: 1.00004960/6328 [=====>.....] - ETA: 22s - loss: 1.8199e-04 - acc: 1.00005792/6328 [=====>.....] -
ETA: 8s - loss: 1.7390e-04 - acc: 1.0000

Epoch 00007: val_acc did not improve from 1.00000

```

Fig 4.5(b) Accuracy metrics for Model 5

The training and validation accuracy showed a result of 100% which seems to be perfect but the on-spot testing fails to a huge extent. Thus this amount of huge accuracy can be explained with the view of overfitting as in this model we are feeding the whole audio file at a time and doing this process for an increased number of sample thus somehow the same features are fed a number of times to the sample which makes the system learn these values and give accuracy according to that but when new samples which the system has not seen earlier the system tends to fail. Thus this approach can't be taken.

7. FEW CLOSE OBSERVATIONS

While working on this project, many insights about the phones were gathered and some problems regarding the collected data was faced, which are mentioned as follows:

i) Since the data was manually collected and from different persons the length of audio files greatly varied maximum being of (1-2) second of length but the overall range of the files were from (1-8) seconds and all were sampled at different frequencies and being in different format, thus it was quite difficult to make them a homogenous kind of files which affected the model building as selection a uniform vector from all the files was not possible. Even zero padding or splitting technique was used but it gives some irrelevant information to the model and affects the results.

ii) The structure of the phone ‘āge’ was a kind of concentrated wave and other being nothing but some low amplitude parts which can be misjudged with the small amount of unvoiced data present in other phones. Thus the prediction of the word aage was the main exception which was giving hindrance in giving good predictions.

iii) The phonic structure of the word ‘bāen’ and ‘dāen’ are similar, thus, many a times these two words were interchangeably predicted.

iv) The voices ‘dāen’ and ‘bāen’ are quite long in duration thus their impact on the overall training made some bias towards these words and for all the phones these words have some part of probability in the result which somehow decreases the original phonic probability.

v) The computational power and resource requirement to achieve greater results is also needed.

8. FUTURE SCOPE

Speech Recognition is the immediate future of human interaction with computer systems. With this technology growing rapidly, speech engines are becoming more accurate and are well on their way to replace traditional ways of interaction.

- i) With increase in the number of samples and different phones this can help to recognize a huge amount of phones of the language.
- ii) This project can be used for any other language just by changing the repository and training the model with those phones.
- iii) The predictions can further be fed to many home appliances which performs basic operations and automate them just by passing human voices.
- iv) With the achievement of huge resources and computational power the model can cover more and more phones.

9. CONCLUSION

Speech Recognition is a challenging Computer recognizing problem that requires robust Neural Networks that are capable of extracting and processing every minute detail in order to give the best prediction. Researches and studies are going on to establish systems or algorithms enabling better, faster and more efficient recognition.

We trained a deep Recurrent Neural Network to recognize a given phone given through microphone. The models parameters were tweaked changing the feature vectors and envelope threshold, thus changing these parameters we trained 5 models giving different accuracies.

Thus we can conclude that the threshold should not be huge as it might remove useful information and the model 4 was showing better results for further approach.

REFERENCES

- [1] Python Speech Features Documentation visited on 12/10/2019 (<https://python-speech-features.readthedocs.io/en/latest/>)
- [2] Introduction to Convolutional Neural Networks last visited on 13/11/2019 (https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf)
- [3] Deep Learning Concepts using Python(<https://www.youtube.com>).
- [4] <http://datagenetics.com/blog/november32012/index.html> visited on 15/11/2019
- [5] Keras Documentation for the Neural Network and deep learning concepts visited from 13/10/2019 to 02/12/2019 (<https://keras.io/>)
- [6] <https://sonix.ai/history-of-speech-recognition> visited on 24/05/2020
- [7] History of voice recognition system last visited on 12/02/2020
<https://www.condecosoftware.com/blog/a-history-of-voice-recognition-technology/>
- [8] Survey on Hidden Markov Model and works in regional languages visited on 25/03/2020
(https://www.researchgate.net/publication/280628018_Speech_Recognition_in_Indian_Languages-A_Survey_Hidden_Markov_model_HMM_Gaussian_mixture_model_GMM_Artificial_neural_network_ANN)
- [9] Recurrent Neural Network by Mahendran Venkatachalam last visited on 24/05/2020
(<https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce>)
- [10] Fast Fourier Transform visited 25/06/20202 ([https://towardsdatascience.com/fast-fourier-transform-937926e591cb#:~:text=As%20the%20name%20implies%2C%20the,\)%20to%20O\(NlogN\)%20.](https://towardsdatascience.com/fast-fourier-transform-937926e591cb#:~:text=As%20the%20name%20implies%2C%20the,)%20to%20O(NlogN)%20.))
- [11] Mohanty, S., Swain, B. K., “Markov Model Based Oriya Isolated Speech Recognizer- An Emerging Solution for Visually Impaired Students in School and Public Examination,” In: Special Issue of IJCCT Vol. 2 Issue 2, 3, 4; International Conference On Communication Technology-2010.(https://www.researchgate.net/figure/Details-of-Oriya-isolated-words_tbl1_266268918)

- [12] Hegde, S., Achary, K.K., Shetty, S., "Isolated Word Recognition for Kannada Language Using Support Vector Machine," In: Wireless Networks and Computational Intelligence Communications in Computer and Information Science, Volume 292, 2012, pp 262- 269.
- [13] Review on the progress of NLP in India last visited on 01/04/2020 (<http://www.academia.edu/download/36415761/7I23-IJAET0723718-v7-iss5-1420-1425.pdf>)