

Human Voice Recognition in the Regional Language

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

ABSTRACT

This project 'Human Voice Recognition in the Regional Language' is a speech recognition system to recognize a phone from a set of phones in a regional Language (Hindi), 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 (which are आगे(āge), पीछे(peeche), बाँ(āen), दाँ (dāen), चलो (chalo), रुको (ruko)) and then use this recognition system to operate an Arduino based vehicle through on-spot human voice.

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 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

1. INTRODUCTION

2. A BRIEF SURVEY

3. PREPROCESSING WORK

4. PROPOSED APPROACH

5. RESULT AND CORRESPONDING EVALUATION

6. FEW CLOSE OBSERVATIONS

7. FUTURE SCOPE

8. CONCLUSION

REFERENCES

INTRODUCTION

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.

We can use speech recognition system to perform an action based on the instructions defined by the human voice. For that we need to train the speech recognition system by storing speech patterns and vocabulary of their language into the system & based on these patterns we have to define some certain actions.

Now a days many complex recognition system has been built in English language. Be it Google (OK Google), Alexa etc. they are able to recognise complex sentences and give results according to that. But any complex system in Indian Regional Language is not yet developed.

The proposed approach is making a speech recognition system which can predict human voice from specific domain of phones (that are आगे(āge), पीछे(peeche), बाँएँ(bāen), दाएं (dāen), चलो (chalo), रूको (ruko)). The expectation from the system is to make a promising model which can predict voices from human which is expected to be fed to an Arduino based toy vehicles which can be operated from the human voice.

A BRIEF SURVEY

Survey on Tamil & Telugu Automatic Speech Recognition System

- a) Krishnaveni present a Continuous Speech Recognition (CSR) system for Tamil language using Hidden Markov Model (HMM) approach. Used Most powerful and widely used MFCC feature extraction 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.
- b) 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.

Survey on Assamese & Bengali Automatic Speech Recognition System

- a) 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.
- b) S. Mandal, in their paper introduce the SPHINX3-based Bengali Automatic Speech Recognition system Shruti-II and an E-mail application based on it. This ASR system converts standard Bengali continuous speech to Bengali Unicode. This paper also demonstrates an application based on Shruti-II which enables visually challenged people to send E-mail by using this system.

Survey on Kannada ASR System

- a) 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.
- b) 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.

Survey on Gujarati ASR System

- a) 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.
- b) 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.

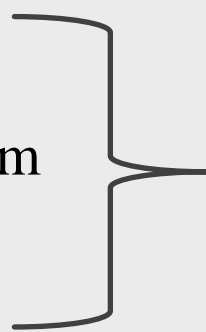
PREPROCESSING WORK

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 were in different formats (such as .mp3, .opus, .m4a etc.). All these files were converted to a uniform format (.wav) for making the training module. We plotted 3 different graphs for each phone, which are:

1. Time series graph

2. Fast Fourier transform

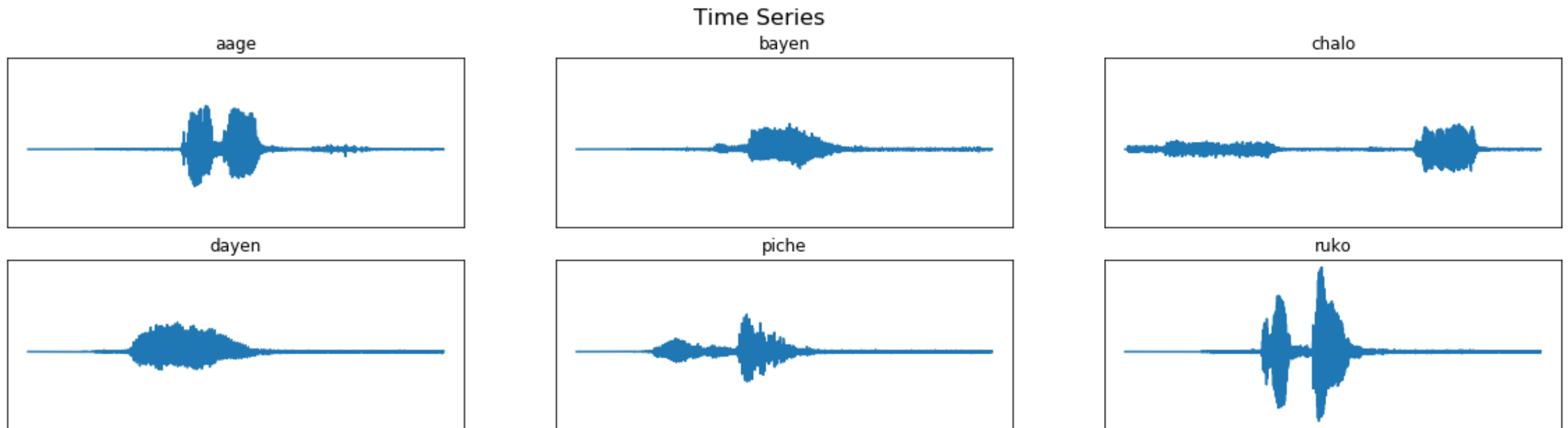
3. MFCC graph



After taking the insights of these graphs 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.

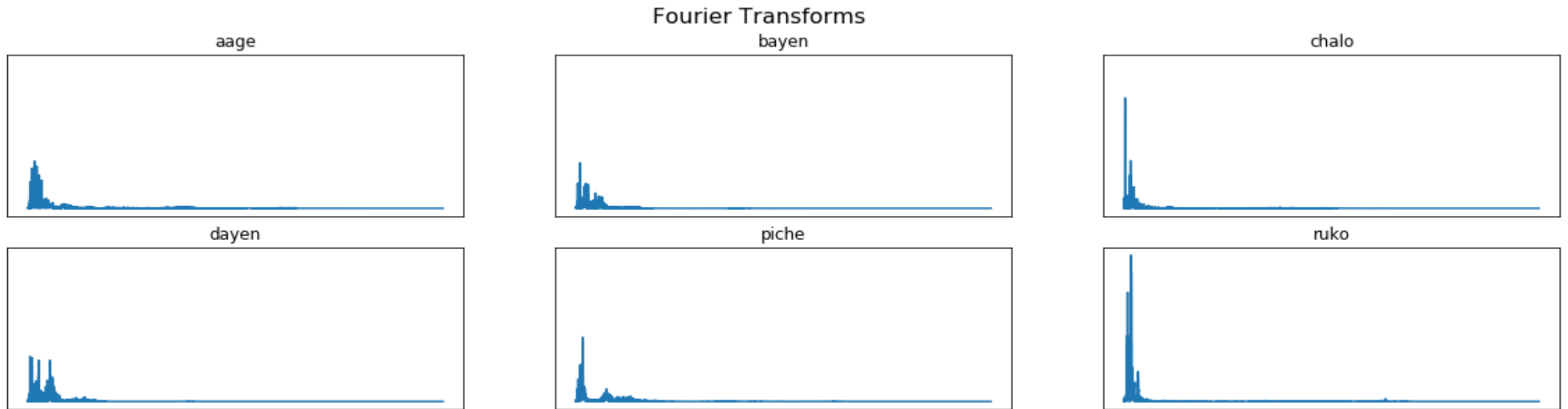
PREPROCESSING WORK-CONTD.

1. Time series graph: Time series plot is a 2D plot of the Amplitude of a signal with respect to time. 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.



PREPROCESSING WORK-CONTD.

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.



PREPROCESSING WORK-CONTD.

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.

MFCCs

aage



bayen



chalo



dayen



piche



ruko



PROPOSED APPROACH

Speech recognition: A technique by which specialized software and systems are created to identify, distinguish and authenticate the voice of an individual speaker.

The world has gone far ahead in Speech Recognition with Development of Siri, Alexa etc.

Many works are done or many are under process in Indian regional languages also but the complexity is not that huge compared to English 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) and will identify 6 phones(that are आगे(āge), पीछे(peeche), बाएँ(bāen), दाएँ (dāen), चलो (chalo), रूको (ruko)).

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

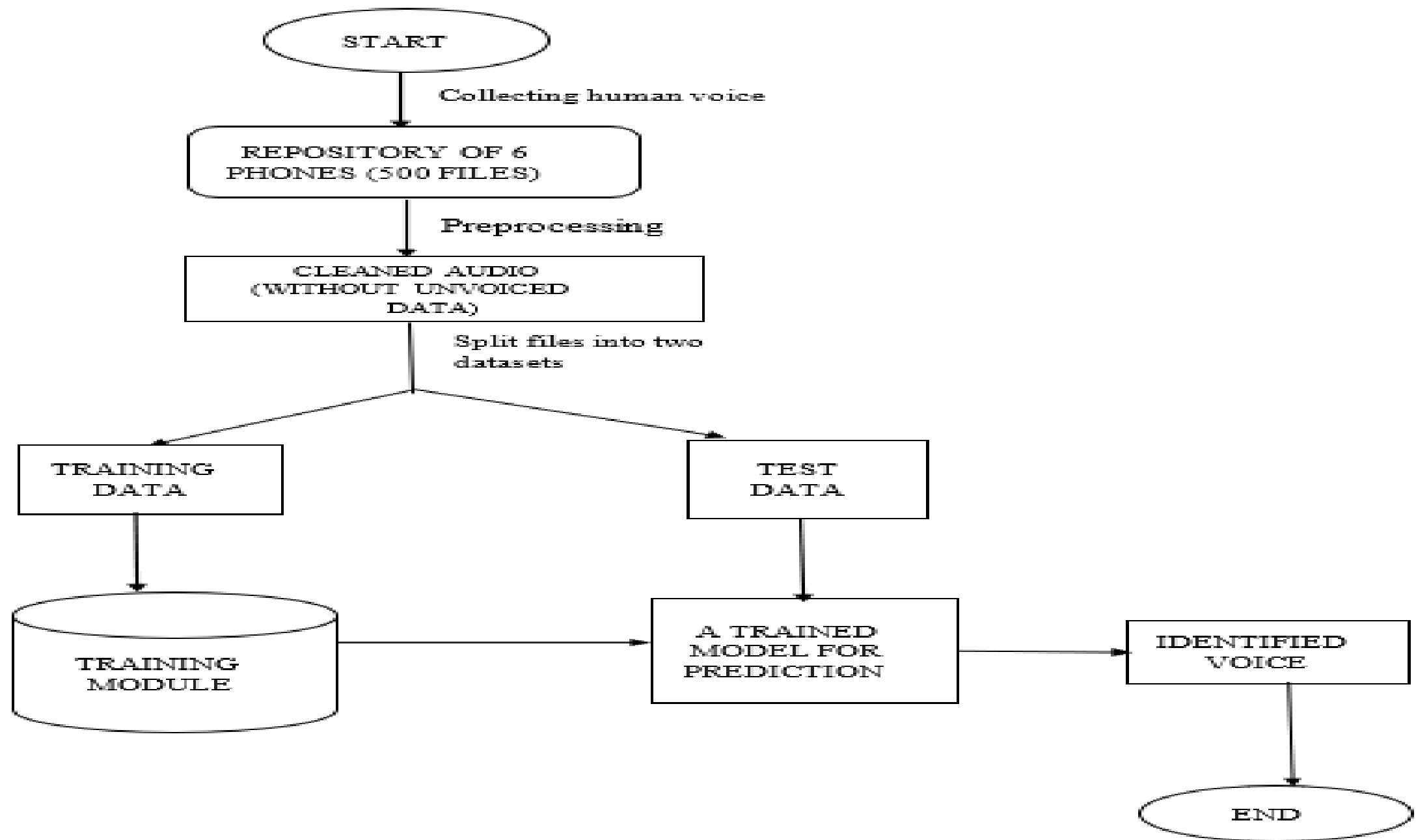


Fig.3(a) Approach for Making a Speech Recognition System

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.

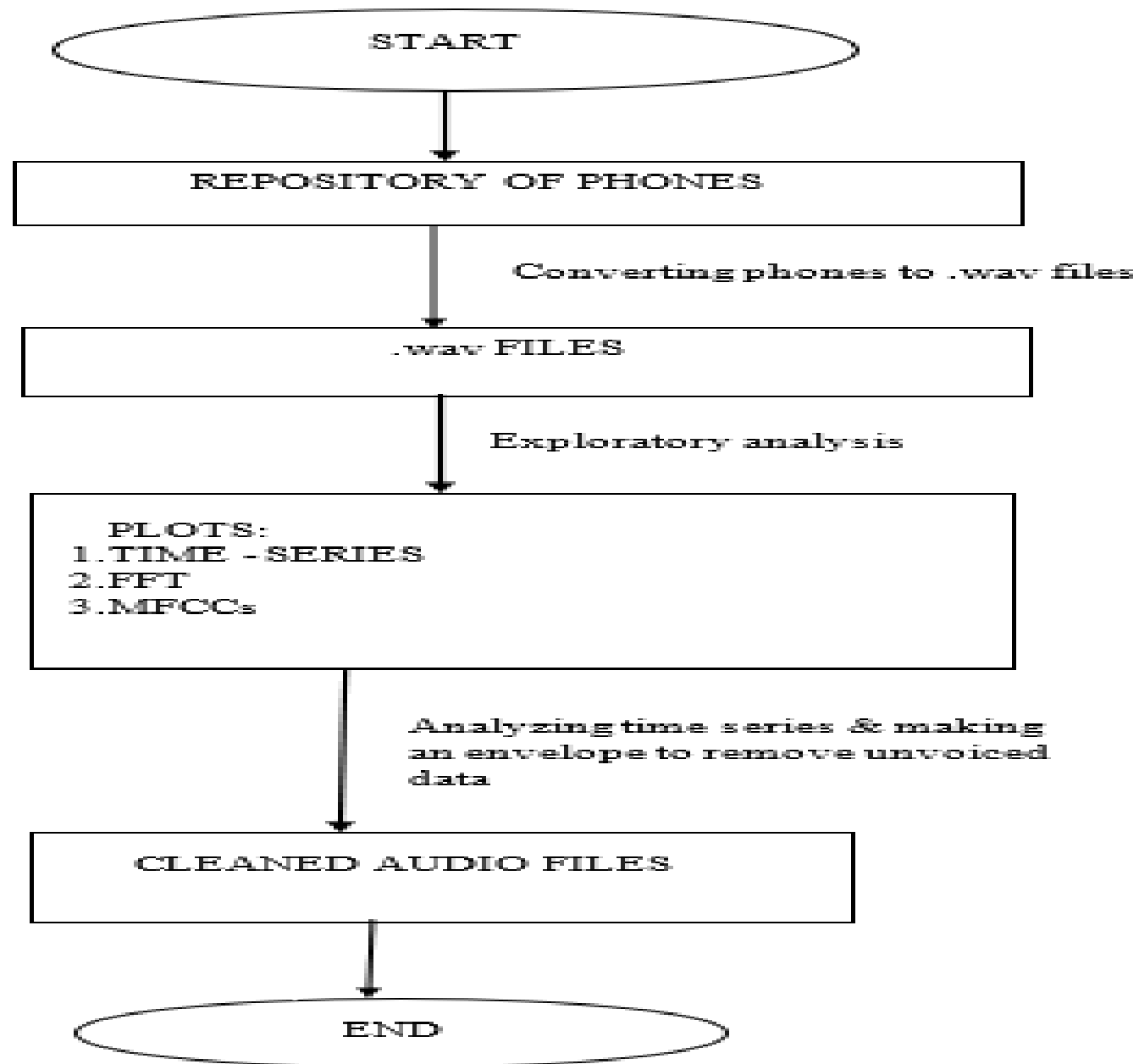


Fig 3(b). preprocessing Steps

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

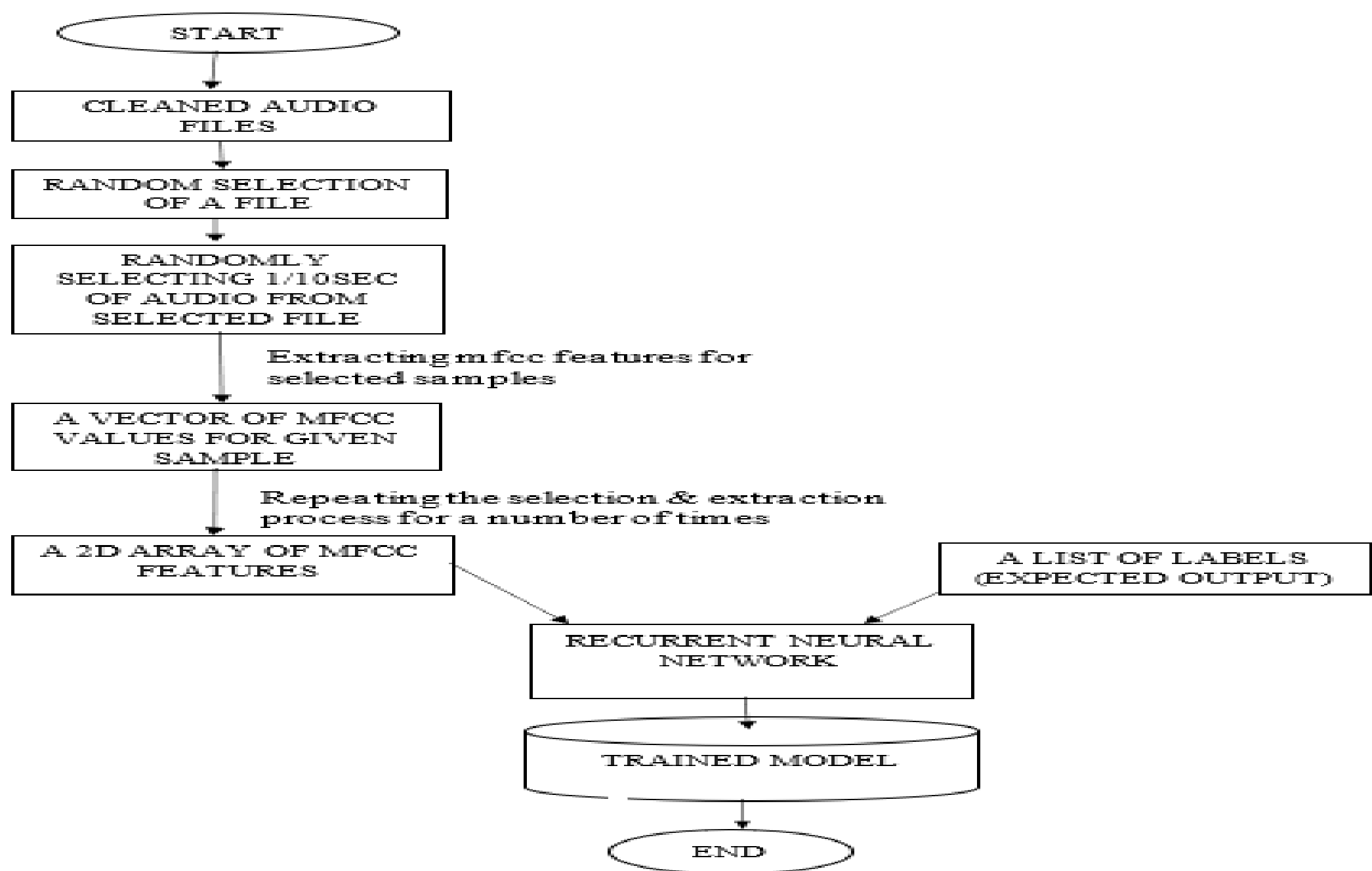


Fig 3(c). Training Module

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

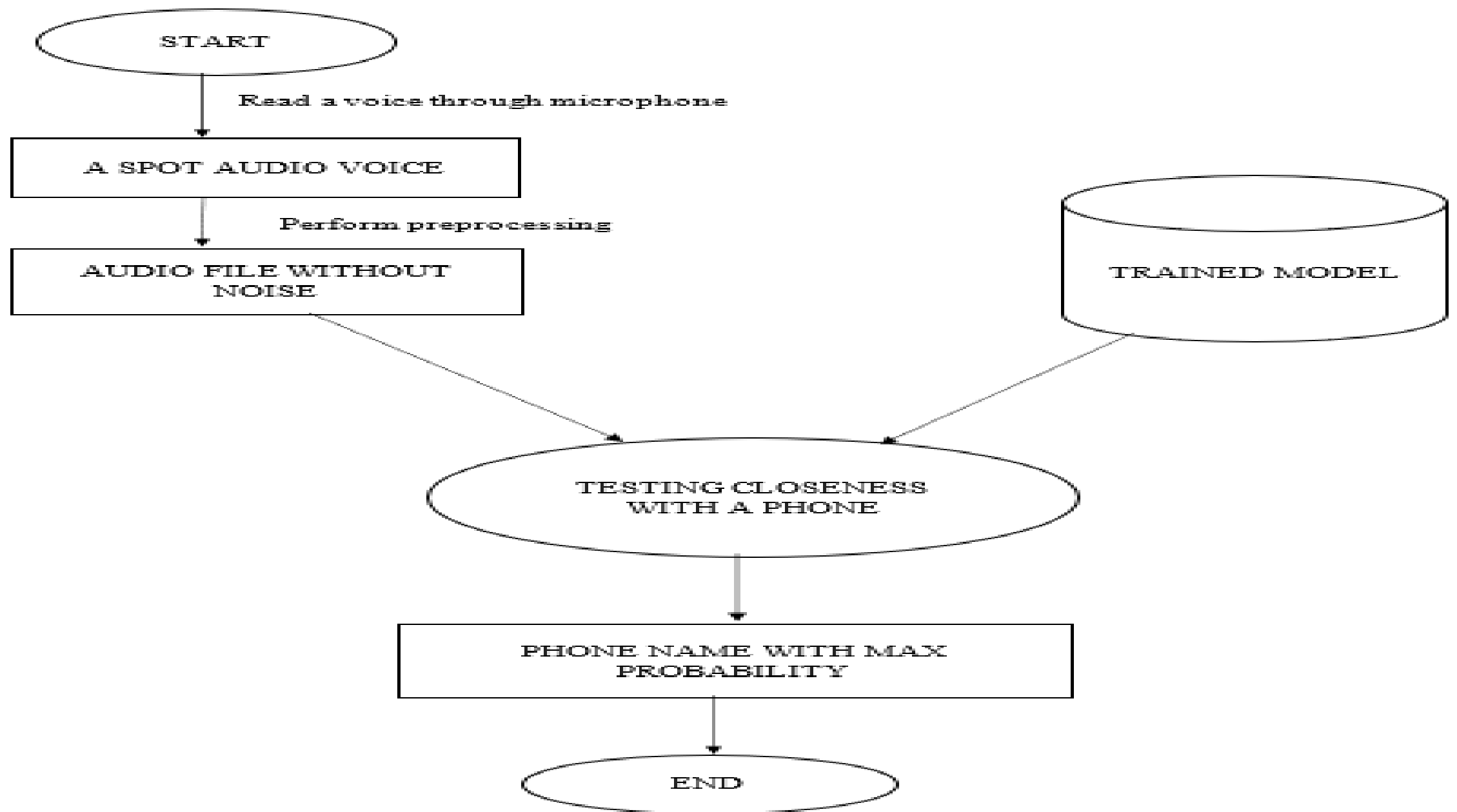


Fig (d). On-spot Testing

RESULT AND CORRESPONDING EVALUATION

Common Terms Used

1) **Feature Selection:** Feature shape is $m \times \text{number of mfcc filters}$.

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 $(390) \times 20$ (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

Model 1

Dimension of Feature vector = (5×20)

Threshold = 0.005

Sample size = no. of audio files i.e. 390

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

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

Layer Structure of RNN 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
```

Accuracy per Epoch for model 1

Model 2

Dimension of Feature vector = (5×20)

Threshold = 0.009

Sample size = no. of audio files i.e. 390

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

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		

Layer Structure of RNN 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

```

Accuracy per Epoch for model 2

Model 3

Dimension of Feature vector = (5×20)

Threshold = 0.005

Sample size = $(20 \times \text{no. of audio files})$

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

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		

Layer Structure of RNN 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

Accuracy per Epoch for model 3

Model 4

Dimension of Feature vector = (9×20)

Threshold = 0.005

Sample size = $(20 \times \text{no. of audio files})$

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

```
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
```

Layer Structure of RNN 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: 0.5006 - acc: 0.8050

Epoch 00010: val_acc did not improve from 0.73418

```

Accuracy per Epoch for model 4

Model 5

Dimension of Feature vector = (172×20)

Threshold = 0.005

Sample size = $(20 \times \text{no. of audio files})$

The training and validation accuracy showed a result of 100% which seems to be perfect but the on-spot testing fails to a huge extend (due to overfitting).

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		

Layer Structure of RNN 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: 0.1150 - 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: 0.0880 - 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: 0.99525888/6328 [=====>....] - 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: 1.00001:21 - 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: 1.0000ETA: 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

```

Accuracy per Epoch for model 5

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) Size or Duration of the audio files were not uniform
- ii) The structure of the phone 'āge' has huge unvoiced data, thus mismatching occurs.
- 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.

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.

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 25/06/2020 ([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] 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.

[12] 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>)

THANK YOU!