**B.E. Project**

**ON**

# Text-Independent Speaker Recognition Using Deep Learning

**Submitted by**

**Raunaq Bhalla (2016UIC3022)**
**Pranjal Naman (2016UIC3018)**
**Satyarth Vats (2016UIC3009)**
**Monarch Batra (2016UIC3034)**

In partial fulfilment of B.E. (Instrumentation and Control Engineering) degree of University of Delhi

**Under the guidance of**

**Prof. Smriti Srivastava**

**DIVISION OF INSTRUMENTATION AND CONTROL ENGINEERING**
**NETAJI SUBHAS INSTITUTE OF TECHNOLOGY**
**UNIVERSITY OF DELHI, DELHI**

# SELF DECLARATION

The project entitled “ Text-Independent Speaker Recognition Using Deep Learning ” is a bonafide work carried out by **Pranjal Naman, Satyarth Vats, Monarch Batra and Raunaq Bhalla** in Department of Instrumentation and Control, Netaji Subhas Institute of Technology, Delhi under the supervision and guidance of **Prof. Smriti Srivastava** in partial fulfilment of the requirement for the Degree of Bachelor of Engineering in Instrumentation and Control Engineering, University of Delhi for the year 2019-2020.

The content of this report is to the best of our knowledge and hasn’t been used for any other academic activity.

Date: 27 July 2020

# ACKNOWLEDGEMENT

We would like to take this opportunity to acknowledge the support of all those without whom the project wouldn't have been possible.

We sincerely thank our mentor, **Prof. Smriti Srivastava**, for her guidance, criticism and encouragement which led to the completion of the project. The regular brainstorming sessions with her gave us a deep insight into the topic and evolved our ideas. We thank her for giving us this opportunity and supporting us throughout.

We wish to express heartfelt gratitude to our college, **Netaji Subhas Institute of Technology** for giving us this opportunity for research and development.

Lastly, we would also like to thank our parents who supported us through our thick and thin. Their trust in our capabilities helped us in giving our best.

| | |
|---|---|
| **Pranjal Naman** | **Monarch Batra** |
| **2016UIC3018** | **2016UIC3034** |
| **Satyarth Vats** | **Raunaq Bhalla** |
| **2016UIC3009** | **2016UIC3022** |

**Netaji Subhas Institute of Technology**

Azad Hind Fauj Marg, Sector - 3, Dwarka, New Delhi-110078
Website: www.nsut.ac.in

# Thesis Certificate ( Supervisor )

This is to certify that the project entitled **"Text-Independent Speaker Recognition Using Deep Learning"** is a bonafide work done by Mr. Pranjal Naman, Mr. Satyarth Vats, Mr. Monarch Batra and Mr. Raunaq Bhalla, students of eight semester, B.E. Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Delhi.

This project work has been prepared as a partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Instrumentation and Control Engineering, University of Delhi, in the academic year 2019-2020.

This work has not been presented for any other academic purpose before. I wish them luck for all their future endeavours.

Date: 27 July 2020

**Prof. Smriti Srivastava**

Professor, Deptt. of Instrumentation and Control Engineering

Netaji Subhas Institute of Technology

**Netaji Subhas Institute of Technology**

Azad Hind Fauj Marg, Sector - 3, Dwarka, New Delhi-110078
Website: www.nsut.ac.in

# Thesis Certificate ( HOD )

This is to certify that the project entitled **"Text-Independent Speaker Recognition Using Deep Learning"** is a bonafide work done by Mr. Pranjal Naman, Mr. Satyarth Vats, Mr. Monarch Batra and Mr. Raunaq Bhalla, students of eight semester, B.E. Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Delhi.

This project work has been prepared as a partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Instrumentation and Control Engineering, University of Delhi, in the academic year 2019-2020.

This work has not been presented for any other academic purpose before. I wish them luck for all their future endeavours.

Date: 27 July 2020

**Dr. Prerna Gaur**

Professor and Head of Division,

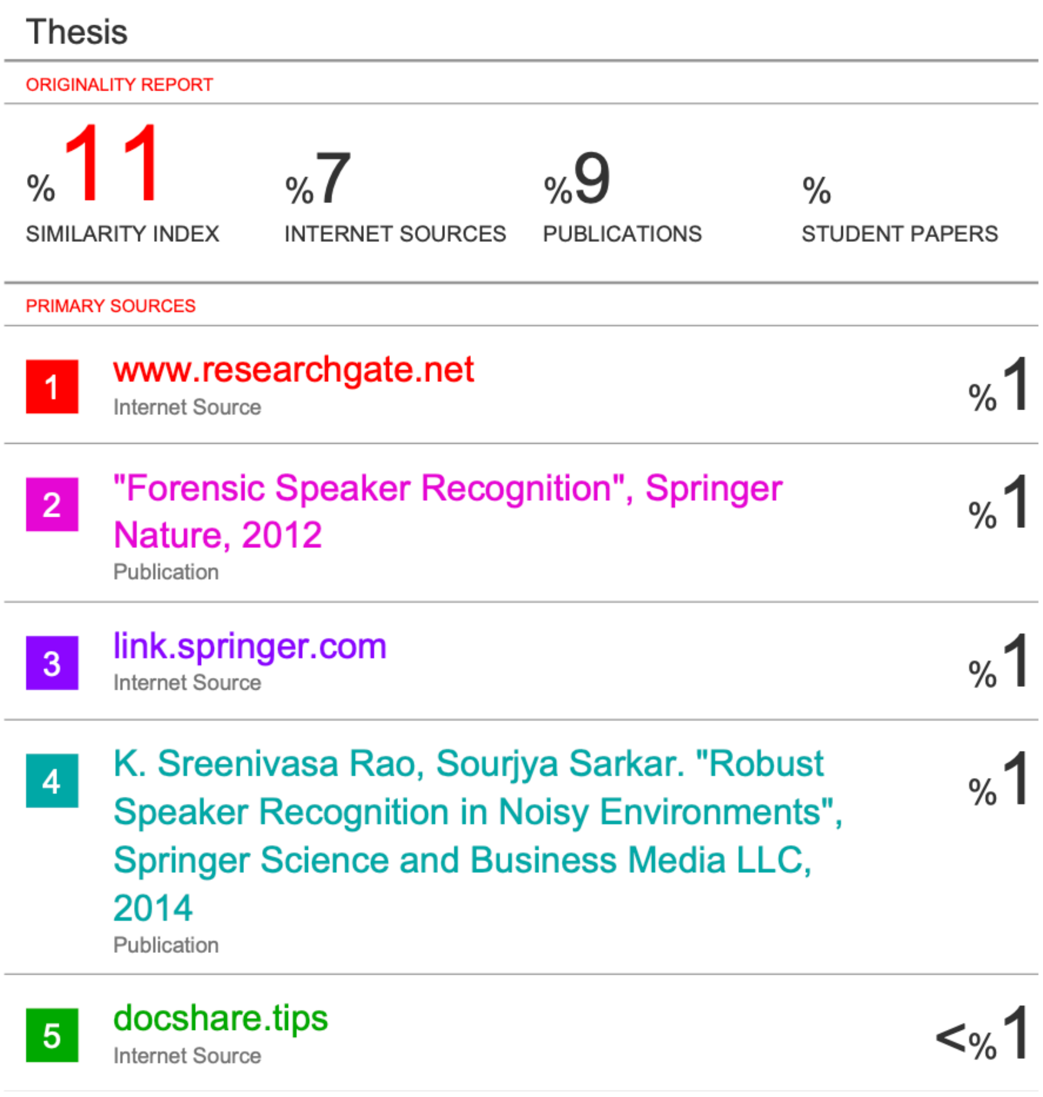
Deptt. of Instrumentation and Control Engineering

Netaji Subhas Institute of Technology

# Similarity Report

The entire research thesis was compiled and processed through “Turnitin” plagiarism checker. Turnitin is an American commercial, Internet-based plagiarism detection service which is a subsidiary of Advance. The site checks submitted documents against its database and the content of other websites with the aim of identifying plagiarism.

**The final plagiarism score received was 11%.**



Thesis

ORIGINALITY REPORT

| %11 | %7 | %9 | % |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.researchgate.net<br>Internet Source | %1 |
| 2 | "Forensic Speaker Recognition", Springer Nature, 2012<br>Publication | %1 |
| 3 | link.springer.com<br>Internet Source | %1 |
| 4 | K. Sreenivasa Rao, Sourjya Sarkar. "Robust Speaker Recognition in Noisy Environments", Springer Science and Business Media LLC, 2014<br>Publication | %1 |
| 5 | docshare.tips<br>Internet Source | <%1 |

# Abstract

**Speech** is a vocal form of communication in Human Beings. It combines the phonetics of consonants and vowels to form sound of words. All this constitutes the linguistic information of speech. Apart from this linguistic information, speech also incorporates various other features such as emotion, gender and identity of the individual. Decoding these features into numeric features allows us to differentiate between individuals and their unique speech content. This can be done with the help of speech recognition.

**Speech recognition systems** are at the crossroads of Digital Signal Processing and Machine Learning. These systems find applications in multiple sectors such as audio-prompts in car systems, biometric systems, military applications and robotics. These can be classified into text-dependent and text-independent speech recognition systems. In a **text-dependent system**, the system requires training ie. the speaker has to read a predefined text or a fixed set of vocabulary. The system analyses this training and tunes its recognition parameters. In case of a **text-independent system**, there are no constraints on the words which the speakers are allowed to use. The entire system can be categorised into speech identification and speech verification. Speaker identification involves the identification of input speech using trained parameters. On the other hand, speaker verification aims to verify if the input speech actually mapped to a speaker in the database.

Through this project, we have ventured into automatic speech recognition systems using **Deep Learning**. This can be evaluated as a typical supervised learning problem. The entire process can be divided into three parts -

1. Extracting features from voice samples
2. Building a multi-class classification model using the extracted features and their respective labels
3. Testing this model on our test samples

We started working on the **Voxforge Open Speech** dataset and the **TIMIT Acoustic-Phonetic Continuous Speech Corpus**. Methods such as Gaussian Mixture Models (GMM), Hidden Markov Models (HMM) and Deep Learning methods like Convolutional Neural Networks (CNN) were studied and applied on these datasets. All of these make use of **Mel Frequency Cepstral Coefficients (MFCC)**. The extracted features were divided into training and testing. These MFCCs were extracted from the datasets and then fed to the CNN.

This paper presents speaker recognition as a typical supervised learning problem with the audio as features and the speaker ID as labels. The focus is on using deep learning methods to solve this supervised learning problem and to optimise the **hyper-parameters** of this deep learning model to produce good results. The number of classes being considered is fairly large and thus the model can be assumed to perform well in case where there are a large number of speakers. The approach suggests extraction of audio features in the form of MFCCs and uses the power of convolutional neural networks to generate a well performing speaker recognition models.

The CNN concerned with this project has **four convolutional blocks** followed by a Flatten layer and a Dense layer. The optimiser used is Adam with **learning rate set at 0.0002 and momentum set at 0.5**. The maximum training accuracy that was achieved was 97.80%. However, the actual performance of the model can only be tracked by evaluating the model on unseen samples and seeing how well it generalises. The model presented in this paper was then tested on over 13000 previously unseen samples from the 1227 speakers and an **accuracy of 96.95%** was achieved.

# List of Tables

# List of Figures

# Table of Contents

# 1. INTRODUCTION

Speech recognition systems are at the crossroads of Digital Signal Processing and Machine Learning. These systems find applications in multiple sectors such as audio-prompts in car systems, biometric systems, military applications and robotics. These can be classified into text-dependent and text-independent speech recognition systems. In a text-dependent system, the system requires training ie. the speaker has to read a predefined text or a fixed set of vocabulary. The system analyses this training and tunes its recognition parameters. In case of a text-independent system, there are no constraints on the words which the speakers are allowed to use.

The entire system can be categorised into speech identification and speech verification. Speaker identification involves the identification of input speech using trained parameters. On the other hand, speaker verification aims to verify if the input speech actually mapped to a speaker in the database.

## 1.1 OBJECTIVE

The objective of this project is to venture into automatic speech recognition systems using Deep Learning. This is a typical supervised learning problem. The entire problem can be divided into extracting features from voice samples, building a multi-class classification model using the extracted features and their respective labels and testing this model on our test samples.

The focus is on using deep learning methods to solve this supervised learning problem and to optimise the hyper-parameters of this deep learning model to produce good results.

## 1.2 MOTIVATION

Speech Recognition systems find applications in multiple sectors such as audio-prompts in car systems, biometric systems, military applications and robotics. As a result, this supervised learning problem holds a lot of significance across multiple industries. The attempt is to add to the accuracy of this highly spread out and used system.

## 1.3 ORGANISATION OF CHAPTERS

In the upcoming chapters, we will provide more details to provide an insight into the work done. Chapter 2 talks about the literature survey conducted before implementing each functionality. Chapter 3 discusses related works. In chapter 4, we explain the proposed usage of various models and methods. In chapter 5, we talk about the implementation details of the project. Chapter 5 covers the analysis, observations and the results. Finally, in chapter 6 we present the the conclusion of our work.

# 2. LITERATURE SURVEY

This chapter talks in detail about the theoretical and practical concepts explored before starting with the project.

## 2.1 SPEECH

Speech is a vocal form of communication in Human Beings. It combines the phonetics of consonants and vowels to form sound of words. All this constitutes the linguistic information of speech. Apart from this linguistic information, speech also incorporates various other features such as emotion, gender and identity of the individual. Decoding these features into numeric features allows us to differentiate between individuals and their unique speech content. This can be done with the help of speech recognition.

## 2.2 BIOMETRIC VERIFICATION SYSTEMS

Since times memorial, the question of identification of individuals has posed a problem to many. Such unique identifications formed the basis for trust based systems such as banking. It was thus highly imperative that a verification system be reliable so as to avoid cases of fraud. There are records of the 2nd century, Chinese emperor Liu Xie using fingerprints to authenticate confidential seals. Much later, the use of palm recognition of a British officer posted in Bengal is also an example of an approach to human identification. While many such approaches were found, token-based identification systems and more recently Biometric verification systems have prevailed.

The traditional token/knowledge based systems used things like personal identification numbers, which could be alphanumeric or hash based. However, it is common knowledge that while these work for most cases of personnel identification, any individual with access to another's token could impersonate, and fool the system.

This situation begged for a more unique id system. Biometric Verification systems provide the answer.

Biometrics are physical or behavioural characteristics associated with every human being. These could range from things like fingerprints and retina, to DNA and voice patterns. Conversion of these traits to a digital form allows for a robust identity verification system. Application of Machine Learning Algorithms to these digital entities has been tried and tested to produce satisfactory results and as such they can be used in autonomous systems  for identification. However for physiological trait to be accepted as a means for identification it should follow some criteria, as listed below:-

- Universality- The traits should be found in all individuals.
- Uniqueness- The traits of one entity must be distinguishable from traits of every other entity.
- Recordable- Instrumentation to record the trait must exist.
- Performance- Should have a fair balance between speed and accuracy.
- Permanence- The traits should not vary over time. If they do their changes must be recorded.

Based on the above criteria, some of the biometric traits identified to build a verification system are:

- Fingerprints- They refer to the friction ridges of the human finger. They are fairly detailed and obviously, universal. Fingerprint recognition find extensive use in forensic analysis.
- Facial recognition- It is preferred over other methods due to its non-invasive nature, even though it has far less reliability. It is finding growing use in smartphone technology, be it for verification or augmented reality.
- DNA- It is the most extensive form of verification, boasting of near perfect accuracy. DNA analysis can not be used for day to day systems, bothe because of the highly invasive nature of DNA extraction as well as little need for that high of an accuracy.

- Retinal Scans- They seek to record the unique pattern formed by the blood vessel in the human eye. They have a high accuracy as well and thus used for high confidentiality environments.
- Voice recognition- It uses the difference in frequency patterns in humans as a way for identification. Apart from security, they can also be used in smartphone AI assistants.

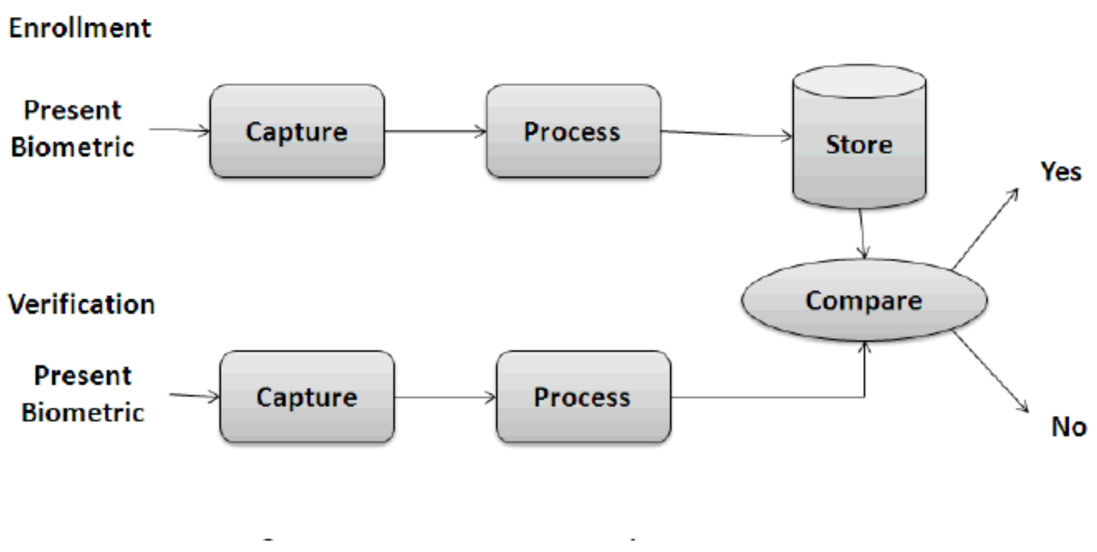


*FIGURE 2.1 : BIOMETRIC VERIFICATION SYSTEM*

## 2.3 SPEAKER RECOGNITION

Speaker recognition is the automatic recognition of the speaker based on an input voice sample. It is essentially a pattern recognition model, in which a speaker's input voice pattern is compared to a store of voice acoustics to identify/verify the speaker. It has found use in several industries as below:-

- Use as a voice assistant in smartphones to increase quality of life. Voice recognition and speech recognition have allowed for automation of trivial tasks, for example use of Siri/Cortana in phone unlock (an example of speaker verification), or to make a call(an example of speech recognition).

- Use as biometric authentication to access restricted areas, or as an attendance management tool. However, these methods should be supplemented with another form of authentication due to the possibility of remote access.
- Use in forensic analysis, to identify from crime scene audio clips the possible identity of perpetrators.

## 2.4 SPEAKER VERIFICATION

It is the part of Speaker Recognition that uses 1:1 mapping to find if the speaker is who they claim. Intuitively it is used mostly in cases of security and authentication. A typical system would involve the person entering a token based identification, upon which they will be prompted to speak either a fixed phrase(Text-Dependent) or any long phrase(Text-Independent). This voice will be recorded, converted to machine readable form, have its features extracted using HMM/GMM or simple frequency analysis, and compared to voice sample patterns present in its stored database. If the person is who he claims, the features would match exactly and they will be allowed access.

## 2.5 SPEAKER IDENTIFICATION

It is the part of speaker recognition that uses a 1:N mapping to find who a speaker is. It is used in cases of forensic analysis where a voice sample containing several speakers, a suspect's, might be needed to be identified. A typical process would involve initially forming an extensive database of voice samples and using them to train the models. Next as and when, a voice sample is obtained and converted to a digital form, its spectral pattern is matched against those present in the stored database(of labelled samples), and the match is outputted.

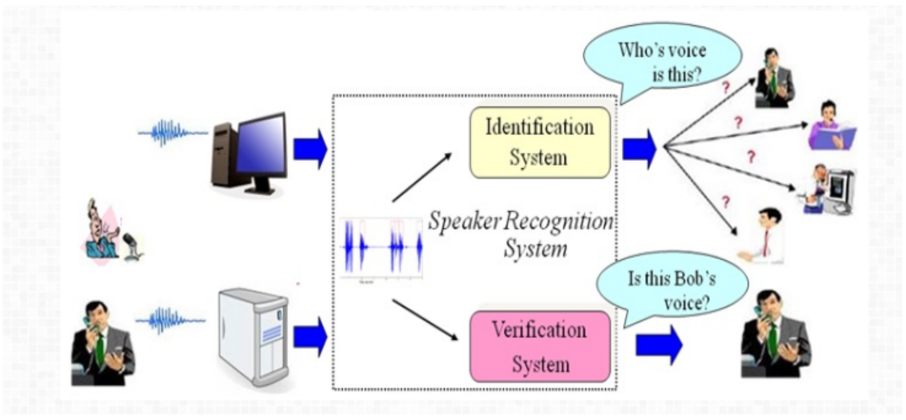


*FIGURE 2.2 : SPEAKER IDENTIFICATION SYSTEM*

## 2.6 TEXT DEPENDENT SPEAKER RECOGNITION

In the context of speaker recognition we talked about the two phases of the process being, enrollment and identification. A speaker recognition system is said to be Text Dependent if the recording of voice samples is based on a fixed prompt. The speaker has to cooperate with the model and have his voice recorded saying a phrase. This recorded phrase in the speaker's voice is then converted to a digital form, it's features are extracted and used to train the model. The preeminent form of feature extraction is done through Hidden Markov Models(HMM).

Compared to the Text-Independent model(as we'll see soon), Text Dependents system allows for a more robust system, usually with very high accuracy on speaker identification. However, this accuracy comes at the cost of such a system being more prone to fraud. For example, a fraud may record the victim uttering the initial phrase and use it to bypass the system. This is usually countered by using a 2 factor authentication system, either in the form of a voice verification and PIN verification combination, or simply using long random phrases initially.

## 2.7 TEXT INDEPEDENT SPEAKER RECOGNITION

Text Independent Speaker Recognition Systems are much loose in the requirement of voice recording. The speaker is not restricted to a fixed phrase during recording. As such it allows for systems to be made with non-cooperative individuals. The feature extraction in such systems is usually done through a combination of HMMs and Gaussian Mixture Models(GMM).

They perform relatively poorly(compared to TDSR), since the phrase during verification and testing may be different. The model would thus require a large training sample to extract fully the spectral features from the speaker's acoustic pattern. This can be done by recording long phrases during testing. Another problem in such systems is the requirement of forming different models for each different language model a speaker can use. Research is being conducted on unifying such models, but no solution has been accepted as yet by the academia. Despite all problems, Text-independent models are much more flexible as recognition systems. It would be easier for organisations to construct a broader database of speakers since cooperation is not required.

It is widely accepted to use text-dependent systems for speaker verification while text-independent for speaker identification.

## 2.8 APPLICATION OF SPEAKER RECOGNITION IN BIOMETRIC VERIFICATION SYSTEMS

Speaker Recognition refers to the identification of an individual based on their voice patterns. Often confused with speech recognition, which is the conversion of spoken words into machine readable language (and is not biometric), it can be used for both verification and identification purposes. Here speaker verification is a 1:1 process, while identification being a 1:N.

An individual’s voice is dependent on the physical shape of the vocal tract, passage of air through which produces the acoustic sounds. Another form of voice recognition would take into account the behavioural component of this vocal system, in the sense of analysing the jaw and lips movement to identify speech.

Unlike other biometric verification systems, Speaker recognition allows for remote authentication. Its ability to dynamically collect voice samples over time due to easy integration of voice sampler into many devices, assists in modification of models as an individual's voice changes with age. The process of recognition would consist of 2 phases, enrolment and verification/identification. In the enrolment phase the individual's voice is recorded, converted to digital form and stored in a database. In the verification/identification phase, input voice samples are compared to labelled voice samples in the database to find the correct match and identify the individual.

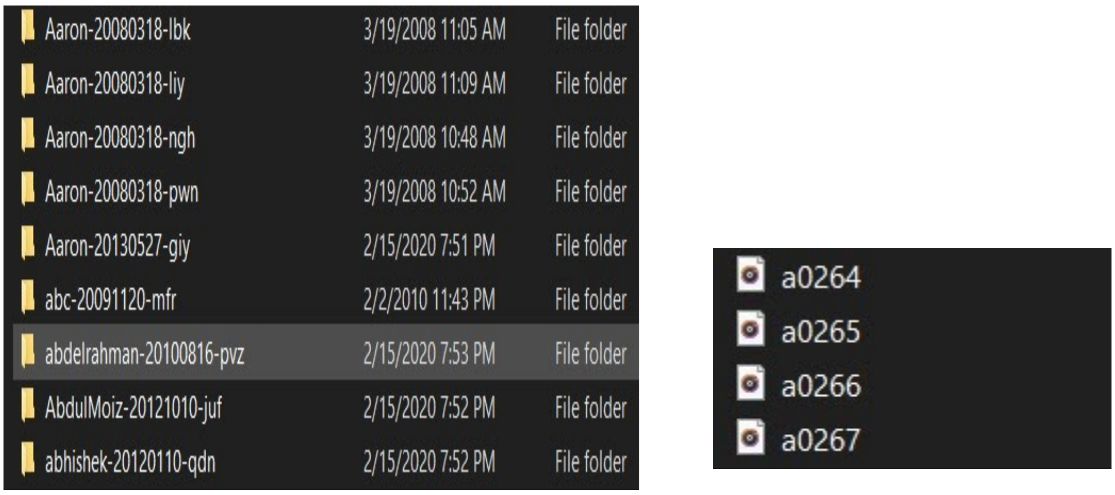
## 2.9 DATASETS

This project involved the usage of deep learning principles on the Voxforge Open Speech dataset and the TIMIT Continuous Speech Dataset. Methods such as Gaussian Mixture Models (GMM), Hidden Markov Models (HMM) and Deep Learning methods like Convolutional Neural Networks (CNN) were studied and applied on these datasets.

### *2.9.1 Voxforge Open Speech Dataset*

VoxForge provides an open speech dataset that collects transcribed speech for use in open source and free speech recognition engines (on Mac, Windows and Linux). They make available all submitted audio files in GPL format and then compile them into acoustic models for use in free speech recognition engines.

The dataset employed in this research paper is publicly available as Voxforge Speech Corpus. The dataset contains speech samples from more than 1300 speakers. However, the dataset contains numerous samples from anonymous sources and including these in the project would not be productive. Each speaker has about 50 or more unique utterances each of about 3 seconds in length. After cleaning and ignoring the anonymous voice samples, a total of 1227 speakers' data was used for the purposes of this project. A total of about 50 thousand voice samples from 1227 speakers was used for training and the model was tested on about 13 thousand unseen samples. The setup used for the purposes of this research paper is rather modest. A system with an i5 processor and 16 gigabytes of RAM was used. Sublime Text was used as the text editor and a metric tracking tool provided by Weights &amp; Biases was used to record the training and testing processes.



```
Aaron-20080318-lbk          3/19/2008 11:05 AM    File folder
Aaron-20080318-liy          3/19/2008 11:09 AM    File folder
Aaron-20080318-ngh          3/19/2008 10:48 AM    File folder
Aaron-20080318-pwn          3/19/2008 10:52 AM    File folder
Aaron-20130527-giy          2/15/2020 7:51 PM     File folder
abc-20091120-mfr            2/2/2010 11:43 PM     File folder
abdelrahman-20100816-pvz    2/15/2020 7:53 PM     File folder
AbdulMoiz-20121010-juf      2/15/2020 7:52 PM     File folder
abhishek-20120110-qdn       2/15/2020 7:52 PM     File folder
```

```
a0264
a0265
a0266
a0267
```

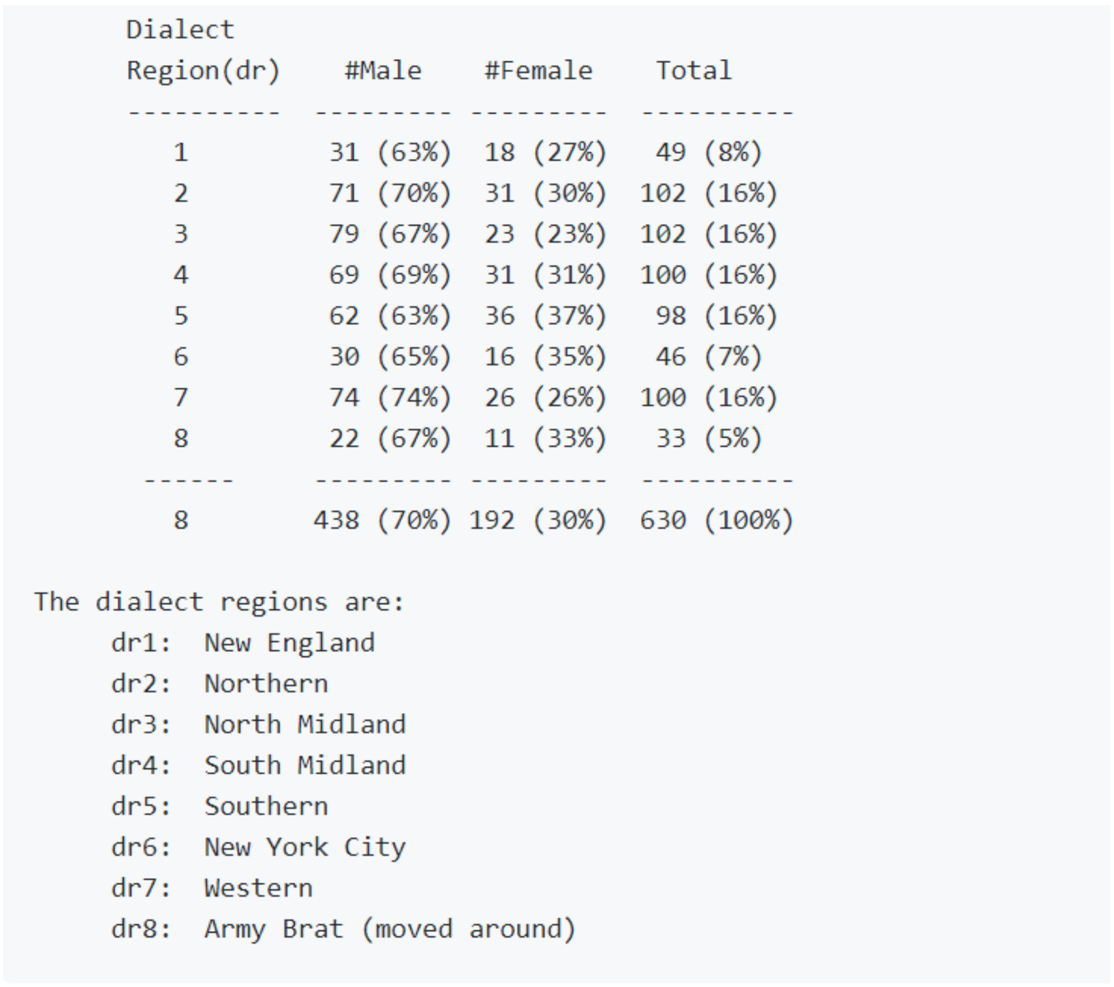
*FIGURE 2.3 : VOXFORGE DATASET*

### *2.9.2 TIMIT Acoustic-Phonetic Continuous Speech Corpus*

The TIMIT corpus of read speech provides speech data to gain acoustic-phonetic knowledge and to develop and evaluate speaker identification models. Text Corpus was created by the joint efforts of Texas Instruments (TI)(responsible for recording the speakers audio clips), Massachusetts Institute of Technology (MIT)(Responsible for transcribing) and Stanford Research Institute (SRI)(Responsible for maintenance).

#### *2.9.2.1 Corpus Speaker Distribution*

TIMIT has a total count of 6300 sentences, there were 630 speakers who spoke 10 sentences each. The speakers belonged to 8 different dialect regions of The United States, each consisting of both male and female samples. The percentages are mentioned in the brackets. The geographical region of the United States where the speaker has spent his childhood years is his dialect region. The geographical regions corresponding to dialect regions in United States (Ohio State University Linguistics Department, Language Files, 1982), with the exception of dialect region 8 where speakers move a lot during their childhood and the Western region(dr7) where a lot of overlaps could be observed.



```
      Dialect
      Region(dr)    #Male    #Female    Total
      ----------  --------- ---------  ----------
         1         31 (63%)  18 (27%)   49 (8%)
         2         71 (70%)  31 (30%)  102 (16%)
         3         79 (67%)  23 (23%)  102 (16%)
         4         69 (69%)  31 (31%)  100 (16%)
         5         62 (63%)  36 (37%)   98 (16%)
         6         30 (65%)  16 (35%)   46 (7%)
         7         74 (74%)  26 (26%)  100 (16%)
         8         22 (67%)  11 (33%)   33 (5%)
       ------     --------- ---------  ----------
         8        438 (70%) 192 (30%)  630 (100%)

The dialect regions are:
     dr1:  New England
     dr2:  Northern
     dr3:  North Midland
     dr4:  South Midland
     dr5:  Southern
     dr6:  New York City
     dr7:  Western
     dr8:  Army Brat (moved around)
```

*FIGURE 2.4 : TIMIT DATASET DISTRIBUTION BASED ON SPEAKERS*

### *2.9.2.2 Corpus Text Material*

The text content in the TIMIT prompts contains 2 dialect “shibboleth” sentences made at SRI, TI selected 1890 phonetically-diverse sentences, and MIT designed a total of 450 phonetically-compact sentences. The goal of dialect sentences (the SA sentences) were to expose speakers dialect variants and all 630 speakers read them. The phonetically-compact sentences were created to provide coverage of pairs of phones, with extra occurrences phonetic contexts could be either of particular interest or difficult. Every speaker read 5 of the SX sentences and 7 different speakers spoke each text. The phonetically-diverse SI sentences were taken from sources such as Playwrights Dialog (Hultzen, et al., 1964) and Brown Corpus (Kuchera and Francis, 1967)- with an aim to add diversity in phonetic contexts and sentence types. The variety of allophonic contexts found in texts was maximised due to the selection criteria used.

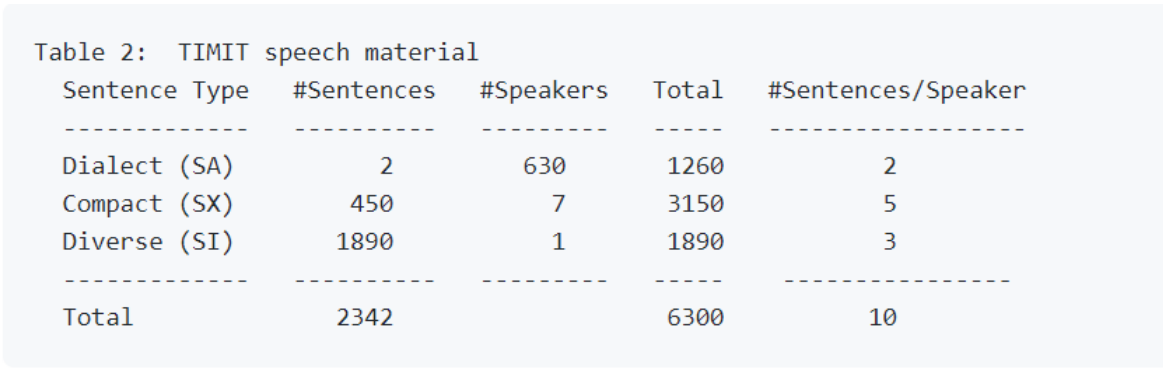


Table 2: TIMIT speech material

| Sentence Type | #Sentences | #Speakers | Total | #Sentences/Speaker |
|---|---|---|---|---|
| Dialect (SA) | 2 | 630 | 1260 | 2 |
| Compact (SX) | 450 | 7 | 3150 | 5 |
| Diverse (SI) | 1890 | 1 | 1890 | 3 |
| Total | 2342 | | 6300 | 10 |

*FIGURE 2.5 : TIMIT DATASET DISTRIBUTION BASED ON SPEECH MATERIAL*

### *2.9.2.3 Training and Test Subdivision*

The criteria used to subdivide the speakers and tests into suggested training and test sets are:

1. The data set has been divided into a 80-20 set, with 80% being used for training and 20% for testing purposes.
2. Speakers can't be a part of both testing and training portions.
3. Care should be taken to include both male and female speakers for each dialect.
4. Care should also be taken to include, in the test set, each of the samples several times with differing phrases.

### *2.9.2.4 Practical Test Set*

The test dataset is composed of 24 speakers, three from each region(with two male and 1 female). This practical test set is represented in the table below. Speakers utter 8 distinct sentences, bringing the total number of utterances to 192.

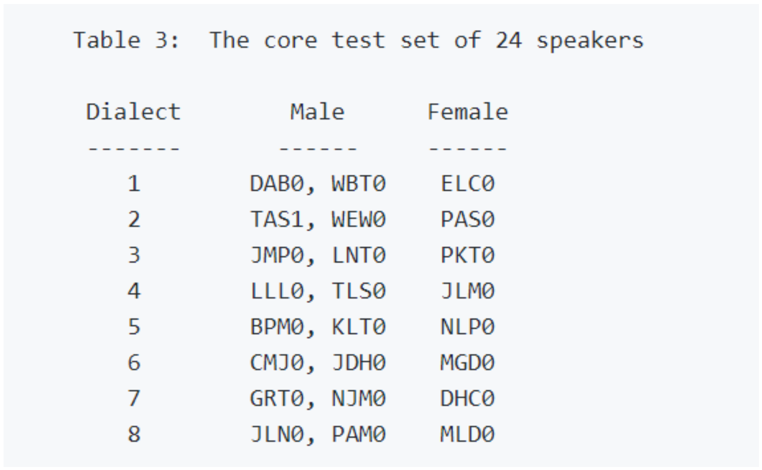


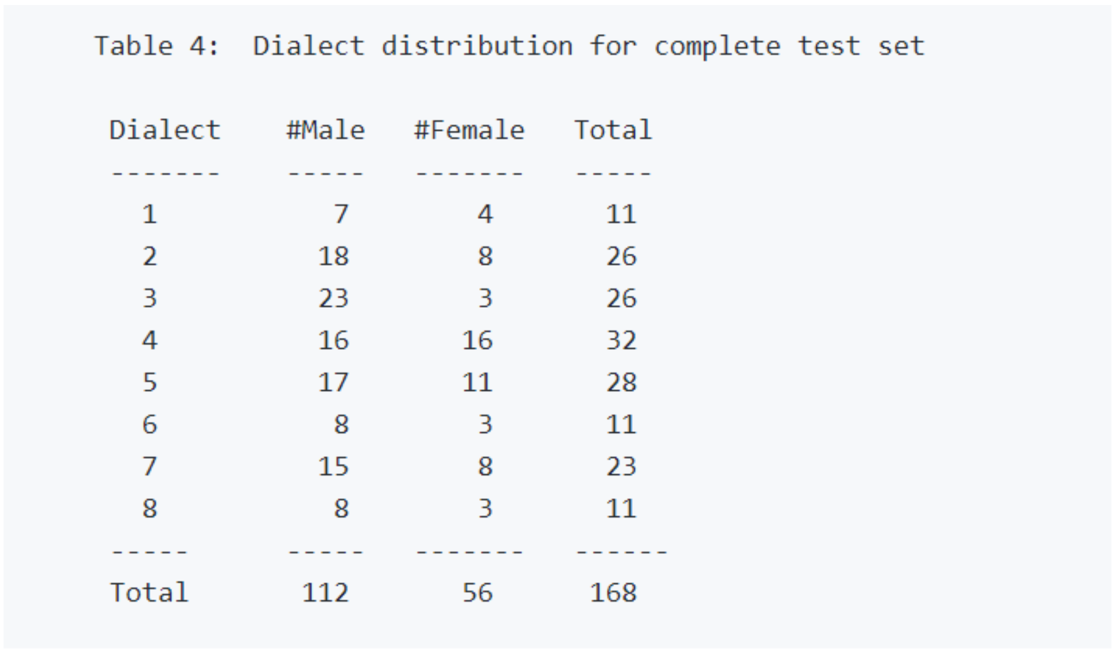
Table 3: The core test set of 24 speakers

| Dialect | Male | Female |
|---|---|---|
| 1 | DAB0, WBT0 | ELC0 |
| 2 | TAS1, WEW0 | PAS0 |
| 3 | JMP0, LNT0 | PKT0 |
| 4 | LLL0, TLS0 | JLM0 |
| 5 | BPM0, KLT0 | NLP0 |
| 6 | CMJ0, JDH0 | MGD0 |
| 7 | GRT0, NJM0 | DHC0 |
| 8 | JLN0, PAM0 | MLD0 |

*FIGURE 2.6 : CORE TEST SET OF 24 SPEAKERS*

### *2.9.2.5 Extensive Test Set*

This set consists of the SX sentences left out in the initial test set.Thus, a clear boundary exists between the training and test sets. This set is then responsible for about 27% of all utterances.

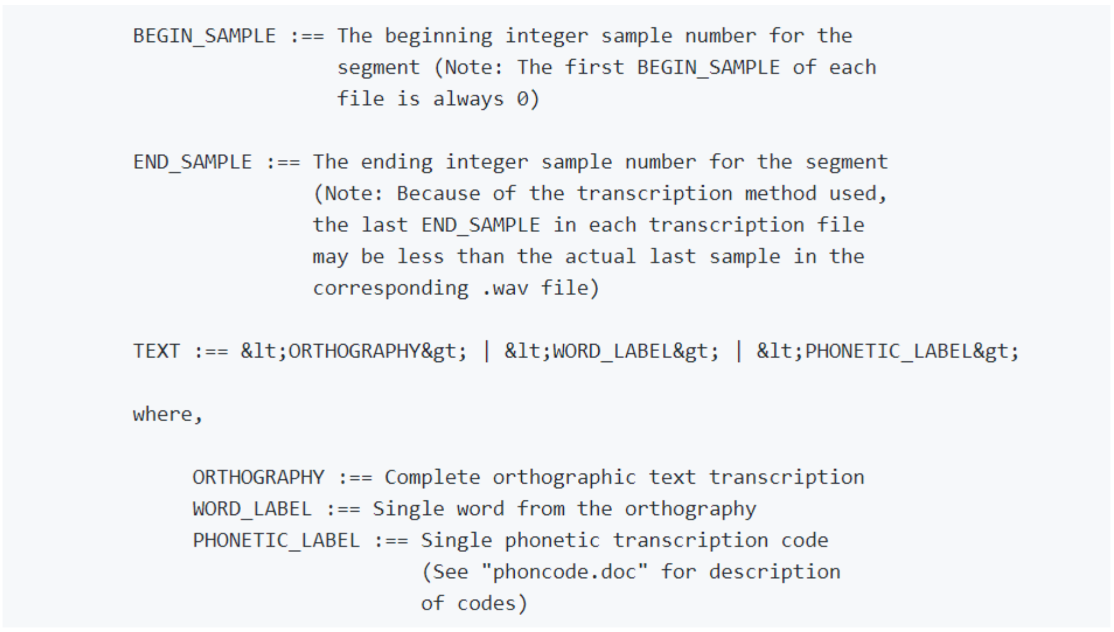


```
Table 4:  Dialect distribution for complete test set

 Dialect    #Male   #Female   Total
 -------    -----   -------   -----
   1           7        4       11
   2          18        8       26
   3          23        3       26
   4          16       16       32
   5          17       11       28
   6           8        3       11
   7          15        8       23
   8           8        3       11
 -----      -----   -------   ------
 Total       112       56      168
```

*FIGURE 2.7 : COMPLETE TEST SET*

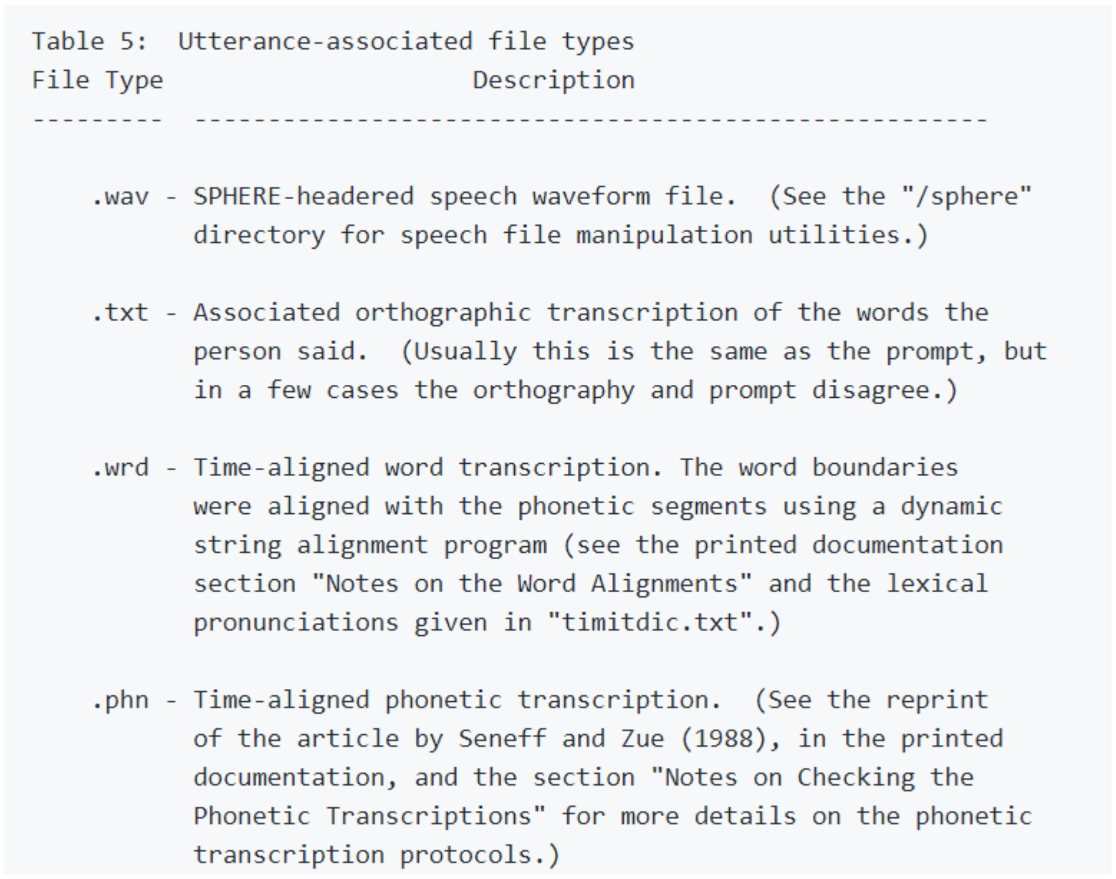
### *2.9.2.6 File Types*

Each utterance in TIMIT corpus is associated with several files. While obviously the audio files are represented in the form of (.wav) files, a set of metadata files exist for the processing of this clip and present as simple text(.tx), or word(.wrd) file.



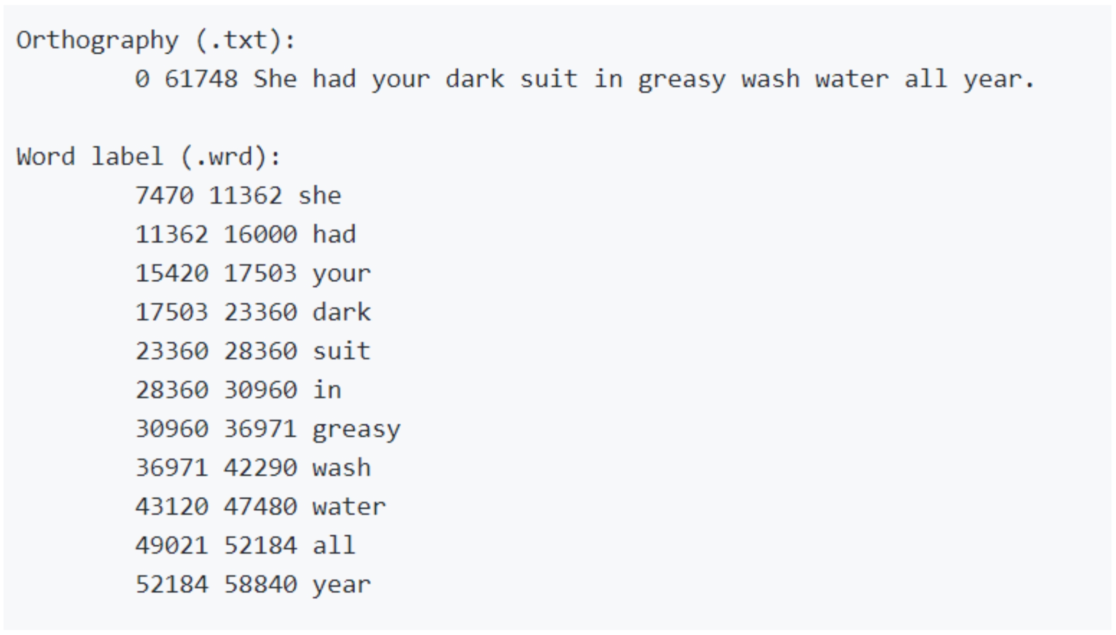
```
BEGIN_SAMPLE :== The beginning integer sample number for the
                 segment (Note: The first BEGIN_SAMPLE of each
                 file is always 0)

END_SAMPLE :== The ending integer sample number for the segment
               (Note: Because of the transcription method used,
               the last END_SAMPLE in each transcription file
               may be less than the actual last sample in the
               corresponding .wav file)

TEXT :== &lt;ORTHOGRAPHY&gt; | &lt;WORD_LABEL&gt; | &lt;PHONETIC_LABEL&gt;

where,

     ORTHOGRAPHY :== Complete orthographic text transcription
     WORD_LABEL :== Single word from the orthography
     PHONETIC_LABEL :== Single phonetic transcription code
                        (See "phoncode.doc" for description
                        of codes)
```

*FIGURE 2.8 : SAMPLE FILE TYPE*



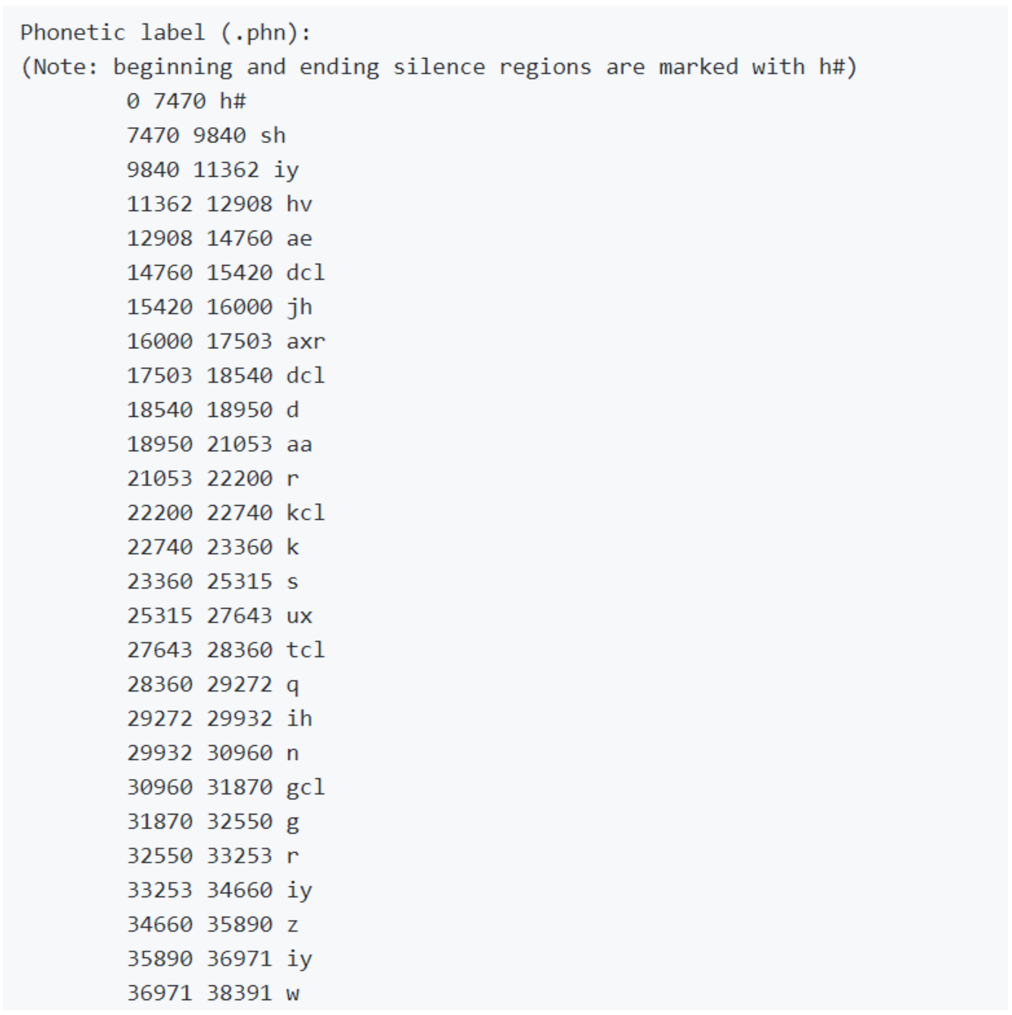
```
Table 5:  Utterance-associated file types
File Type                    Description
---------  ------------------------------------------------------

    .wav - SPHERE-headered speech waveform file.  (See the "/sphere"
           directory for speech file manipulation utilities.)

    .txt - Associated orthographic transcription of the words the
           person said.  (Usually this is the same as the prompt, but
           in a few cases the orthography and prompt disagree.)

    .wrd - Time-aligned word transcription. The word boundaries
           were aligned with the phonetic segments using a dynamic
           string alignment program (see the printed documentation
           section "Notes on the Word Alignments" and the lexical
           pronunciations given in "timitdic.txt".)

    .phn - Time-aligned phonetic transcription.  (See the reprint
           of the article by Seneff and Zue (1988), in the printed
           documentation, and the section "Notes on Checking the
           Phonetic Transcriptions" for more details on the phonetic
           transcription protocols.)
```

*FIGURE 2.9 : UTTERANCE ASSOCIATED FILE TYPES*



```
Orthography (.txt):
        0 61748 She had your dark suit in greasy wash water all year.

Word label (.wrd):
        7470 11362 she
        11362 16000 had
        15420 17503 your
        17503 23360 dark
        23360 28360 suit
        28360 30960 in
        30960 36971 greasy
        36971 42290 wash
        43120 47480 water
        49021 52184 all
        52184 58840 year
```

*FIGURE 2.10 : LABELS IN FILE TYPES*



```
Phonetic label (.phn):
(Note: beginning and ending silence regions are marked with h#)
        0 7470 h#
        7470 9840 sh
        9840 11362 iy
        11362 12908 hv
        12908 14760 ae
        14760 15420 dcl
        15420 16000 jh
        16000 17503 axr
        17503 18540 dcl
        18540 18950 d
        18950 21053 aa
        21053 22200 r
        22200 22740 kcl
        22740 23360 k
        23360 25315 s
        25315 27643 ux
        27643 28360 tcl
        28360 29272 q
        29272 29932 ih
        29932 30960 n
        30960 31870 gcl
        31870 32550 g
        32550 33253 r
        33253 34660 iy
        34660 35890 z
        35890 36971 iy
        36971 38391 w
```

*FIGURE 2.11 : PHONETIC LEVELS*

## 2.10 Convolutional Neural Networks

A Convolutional Neural Network is a type of Neural Network that works by comparing different image matrices after assigning learnable weights to the components of the matrix. One of the most useful features of these types of Neural Networks is their ability to auto extract features from a dataset when provided with enough computation power and over several thousand epochs. Another important feature is their requirement for the least amount of data pre-processing in its class of classification algorithms.

A CNN is modelled after the part of the brain responsible for processing visual information gathered from the eyes. These are built upon the ideas of sparse connectivity and weights sharing. Sparse (local) connectivity enforces units to connect only with a subset of units from the previous layer, called a receptive field, as in the primary visual cortex.

Such units are then replicated across the entire input to form a feature map – a set of units with identical weights and biases, which act like filters for detecting the same feature in different locations, while traversing the input.

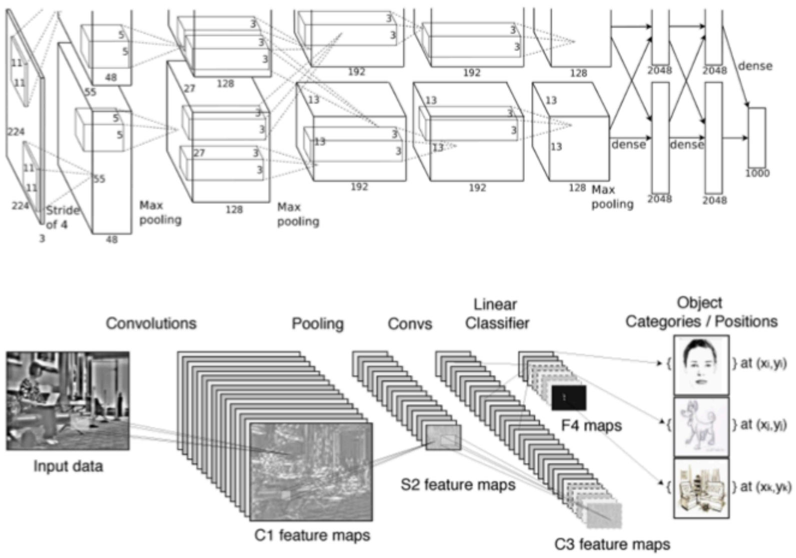


*FIGURE 2.12 : CONVOLUTIONAL NEURAL NETWORK*

A typical CNN Architecture would consist of:

- An input layer, allows for the CNN to take as input a raw image which would require flattening of the image in a feedforward neural network.

- The Convolution layers, A kernel is applied to the input image and a convolution opera-tion takes place between these two matrices. This is done by shifting the kernel over the image in predetermined steps called strides. The resultant image could then be reduced or increased in dimensionality, by using another operation called padding. Conventionally, the convolutional layers are designed in a way that the initial ones are responsible for extraction of low-level features and the complexity of the features extracted increases with every layer. This would give the network a complete understanding of the input image.

- The Pooling layers are responsible for reducing dimensionality of the results from convolution operation, to decrease the computational power required to process the data. It also acts to reduce noise, to make dominant features much more visible.

- The classification layer or the Fully Connected Layer, it learns the nonlinear combinations of features extracted from the combination of convolution and pooling layer.

- The output obtained from this network can be flattened and fed to a feed forward neural network, with back-propagation to get a more accurate classification model.

## 2.11 GAUSSIAN MIXTURE MODELS (GMM)

Machine learning as a field generally deals with only two classes of problems – Supervised learning and Unsupervised learning. Supervised learning is the process of training a machine learning model in a supervised manner which means that the model has to be trained with a mapping of an input to the output. While supervised learning deals with labelled input-output pairs, the unsupervised learning algorithms do not have the same luxury. Unsupervised learning algorithms have to deal with unlabelled data. The way unsupervised learning algorithms deal with unlabelled data is clustering. Clustering algorithms are a part of unsupervised learning algorithms that deal with grouping together data points in the feature space in clusters where points in each cluster are more similar to each other than to the points in other clusters. An example of clustering algorithm is the K means clustering algorithm and this can be understood using the following images –

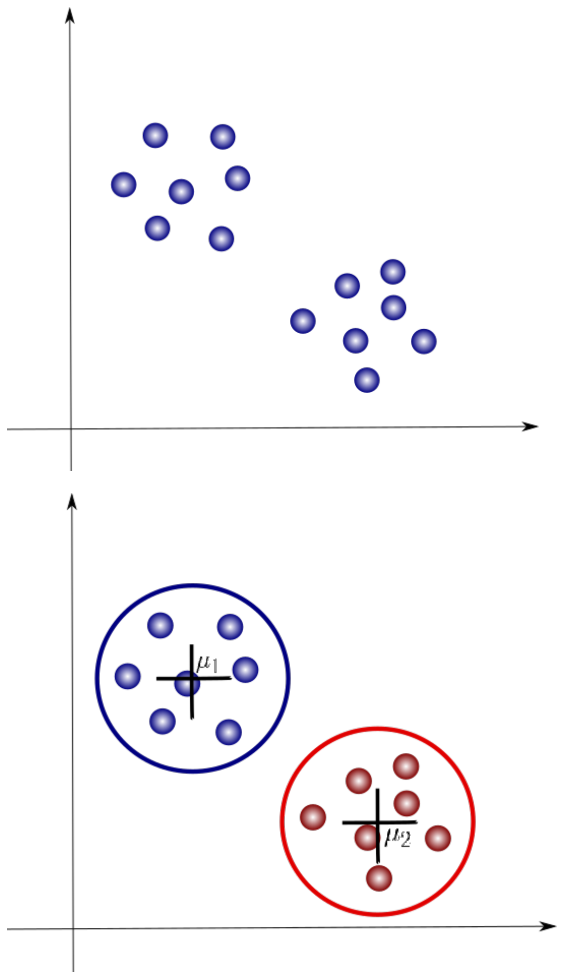


*FIGURE 2.13 : K-MEANS CLUSTERING ALGORITHM*

The K means clustering algorithm divides the feature space into two clusters – the red and the blue one. U1 and u2 here are centroids of each of these clusters and theses are found by the k means algorithm. The problem with this algorithm is that it is a hard-clustering method meaning it associates each point to only one cluster and does not provide the probability of a point belonging to other clusters. The algorithms that provide uncertainty or probability that a point belongs to a cluster are called soft clustering algorithms. This is exactly what Gaussian Mixture Models are supposed to do.

Many datasets are modelled by the Gaussian distribution in real life and therefore, it is only fair to assume that the clusters formed during the clustering algorithm also come from different Gaussian distributions. A dataset can be therefore, in theory, be tried to modelled as a mixture of Gaussian distributions. The gaussian distribution in one dimension is given by –

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and $\sigma^2$ is the variance.

For multivariate Gaussian distribution, the probability function can be given as –

$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}(x-\mu)\right]$$

An assumption is made that there are K clusters and since there are K clusters, the probability density is defined as a linear function of densities of all of these k distributions [insert formula here].

The expectation - minimisation algorithm now works towards finding the right model parameters and the cluster of speaker models can be estimated by working this algorithm.

$$f(x) = \max(0, x)$$

## 2.12 HIDDEN MARKOV MODELS (HMM)

Another way to model sequential data is provided by the Hidden Markov Model. The process works on the assumption that the underlying data to be a markov process. These Markov Models have found increasing use in machine learning, especially in the reinforced learning domain.

A markov process refers to the predictions of probability made on a sequence of random variables. It holds the assumption that future probability be independent of past states and dependent only on current state. The Hidden Markov model differs from this process in the sense that while in the simple model we had to compute probability of observable events, in HMM the events themselves are said to be hidden i.e. unobservable.

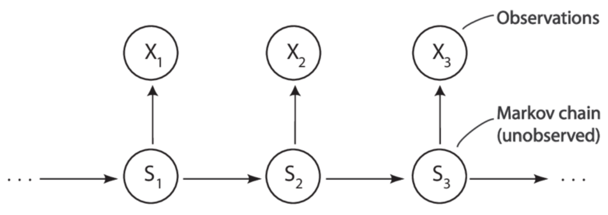


*FIGURE 2.14 : THE HIDDEN MARKOV MODEL ARCHITECTURE*

One of the thought process when evaluating probability of a random variable from an unobservable sequence using HMM is its characterisation using the three problems given in the tutorial of Jack Ferguson in 1960s, listed as below:

- Problem 1, determination of likelihood. Given an HMM distribution and an observable sequence, finding the likelihood of the observable sequence occurring in the distribution.

- Problem 2, deals with decoding. This relates to the determination of the best hidden sequence when the observation sequence and the HMM distribution is known.
- Problem 3, deals with learning associated with HMMs. It requires the learning of HMM parameters, when the observation sequence is known along with the states in the HMM.

## 2.13 MEL FREQUENCY CEPSTRAL COEFFICIANTS (MFCC)

The primary goal in a speech based application is to derive features from the raw audio data in a manner that can be utilized by the system to process it. There are components of the audio signal that are important such as the linguistic content and there is information that is of no use such as background noise etc. The sounds and the voices generated by humans are filtered by the shape of the vocal tracts. The shape of the vocal tract approximately represents the phoneme being produced by the person. A phoneme is a distinct unit of sound that distinguishes one word from another. A need for a mathematical representation of the vocal tract arose for speech based applications.

Prior to MFCCs, features like Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) was used. The concept of Mel Frequency Cepstral Coefficients was introduced by Davis and Mermelstein in 1980s and have been at the front of the line ever since. MFCCs have been widely used in speech and speaker recognition applications as a method for feature extraction succeeded by supervised or unsupervised learning algorithms like Guassian Mixture Models or Hidden Markov Models etc.

Calculation of MFCCs from raw audio data is a digital signal processing technique comprising steps mentioned in brief below. A detailed explanation of the steps has been mentioned later in this thesis.

- The signals are framed in short frames
- Windowing (Hamming Window)
- Discrete Fourier Spectrum
- Conversion of frequencies to Mel spectrum
- Discrete Cosine transform

# 3. RELATED WORKS

Speech being the most prominent mode of communication between humans, researchers have spent countless hours on the subject of speech and speaker recognition. Early papers on the topic focussed on the various potential applications of speech/speaker recognition. Among them, the use of speaker recognition for biometric authentication was agreed to possess the most utility.

The problem of speaker recognition has thus led to development of various models.

- One of the very first solutions to the problem of speaker recognition was given by Reynold D. in 1995. His research[8] demonstrated the use of statistical speaker recognition models for automatic recognition. He presented the first use of a GMM-UBM(Universal Background Model) as a method for feature extraction from acoustic patterns.

- Other papers [3,4,9], used his research on GMMs as a basis for upgrading their own models. Sinith's paper[3], utilized GMM to model speech features extracted from voice samples using Mel-Frequency Spectral Coefficients. They then used the Maximum Likelihood Detection Algorithm as a decision making process. They claimed a recognition rate of 98.6% for a 60 second duration audio clip, with 16 Gaussians.

- Paper[4], used a similar process, having used MFCC for feature extraction and subsequent modelling using GMM. However for the decision making process they used Expectation Maximization Algorithms to achieve an eventual recognition accuracy of 87.5%.

- In Reynold's other paper[9], the distribution of feature vectors extracted from a person's speech is modelled by a Gaussian Mixture Density. This approach is essentially an unsupervised, clustering algorithm that first generates GM models for different speakers and then uses pattern matching of a test sample with all these models to predict the identity of speaker.

- Paper [13] explores the use of continuous density HMM and fuzzy model with the advantages of both Mamdani and Takagi-Sugeno methods. The HMM is utilized for extraction of shape-based batch feature vectors that are fitted with GFM to identify the speaker.

- Current research is being conducted predominantly using deterministic techniques like Dynamic Time Warping(DTW) and Vector Quantization. However the use of the aforementioned GMM and Hidden Markov Model are also gaining popularity.

# 4. METHODOLOGY

## 4.1 LIBRARIES USED

- Os – The OS module is used for interacting with the operating system. It is one of the basic utilities module that is used to rearrange and reorganise the dataset in a manner that can be useful to the rest of the code. os.listdir() is used to list out all the files in the dataset and remove the unnamed and the anonymous speakers from the dataset.

- Numpy – The Python library numpy is used for manipulation of arrays and tensors to make them acceptable for our convolutional neural network.

- Sklearn – sklearn or Scikit-Learn is a Python library mainly concerned with loading different machine learning models. sklearn.preprocessing.train_test_split() is used to split the dataset into train and test datasets for training and testing phases respectively.

- TensorFlow – TensorFlow is an open source Python library for numerical computation and differentiable analysis. It is essentially a math library and is used for machine learning algorithms primarily building and using neural networks. TensorFlow is used to train and run deep neural networks for large scale machine learning problems. Tensorflow applications can run on almost any convenient target be it a local machine, a cluster on the cloud, CPUs or GPUs.

- Matplotlib – matplotlib is a Python library that is mainly concerned with visualisation in Python. Matplotlib is plotting library that can be used to plot graphs and charts. matplotlib.pyplot is used to understand trends, patterns and to make correlations. This library provides instruments to draw insights from raw data or results obtained.

- Librosa – Librosa is a signal processing Python library primarily used for music and speech processing. librosa.load() is used to load the audio signal so that feature extraction can be performed. librosa.feature.mfcc is used to extract and generate MFCC features to be later used for training.

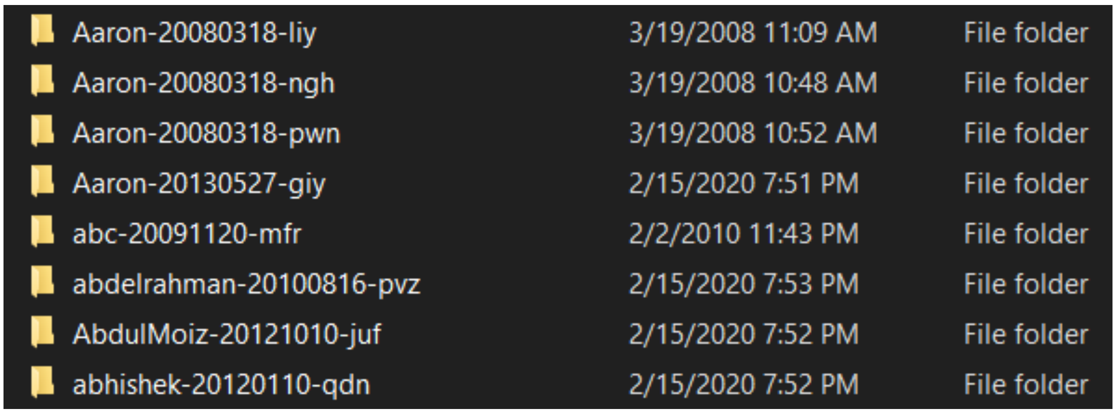
- Pickle – Pickle is used to save and load machine learning models so that the model doesn't have to be run every time to load the model.

- Wandb – Wandb or Weights and Biases is an experiment tracking and model optimisation tool. Weights & Biases provides an interactive dashboard that can be used for logging and plotting training and testing metrics. It can be directly imported and initialised in the code itself and can be used for tracking of loss or accuracy, hyper-parameter search and model optimisation. The process to set up wandb and its functionality is described in the forthcoming sections.

## 4.2 LOADING DATA

The first step in any machine learning problem is to load the data that is to be worked upon. As we know from our discussion above, there are two datasets that have been used for the purposes of this thesis – the VoxForge Corpus and the TIMIT database. Each of these databases have been loaded into separate files but the core code remains the same. The process and model have been optimized and tested for the VoxForge dataset and the same model has also been used on the TIMIT database to see how well the model transfers onto different datasets. In other words, the VoxForge dataset is the basis dataset for this research.

As we know from the discussion in the Dataset section, the total size of the first database is north of 7 gigabytes. The database contains more 6000 folders containing utterances from different speakers. However, as is evident from the file structure shown below in Fig.4.1, the dataset contains random number of folders for each of these users.



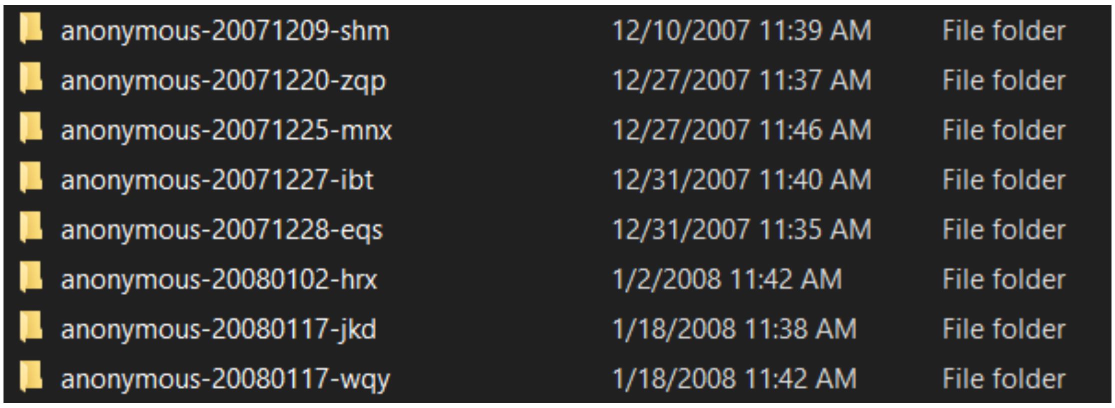
| | | |
|---|---|---|
| Aaron-20080318-liy | 3/19/2008 11:09 AM | File folder |
| Aaron-20080318-ngh | 3/19/2008 10:48 AM | File folder |
| Aaron-20080318-pwn | 3/19/2008 10:52 AM | File folder |
| Aaron-20130527-giy | 2/15/2020 7:51 PM | File folder |
| abc-20091120-mfr | 2/2/2010 11:43 PM | File folder |
| abdelrahman-20100816-pvz | 2/15/2020 7:53 PM | File folder |
| AbdulMoiz-20121010-juf | 2/15/2020 7:52 PM | File folder |
| abhishek-20120110-qdn | 2/15/2020 7:52 PM | File folder |

*FIGURE 4.1 : SNAPSHOT OF DATASET FOLDERS*

This problem has to be tackled since audio from the same speaker must be clubbed together regardless of the fact that it is on different folders. This is taken care by *os.listdir* and a little data manipulation involving extraction of audio from all of these folders and placing them on same folders if they are from the same user, otherwise a new folder from the new user is created.

Another problem that arises while loading data from VoxForge corpus is the presence of anonymous speakers as is evident from Fig 2. VoxForge dataset contains a large number of anonymous speakers which have to be removed and not be loaded into our usable dataset since these anonymous users serve no purpose as speaker recognition cannot be applied to these.

Therefore, these anonymous speakers have to be removed. This is taken care by a simple *if-else* construct while loading the dataset into a usable format.



```
anonymous-20071209-shm      12/10/2007 11:39 AM    File folder
anonymous-20071220-zqp      12/27/2007 11:37 AM    File folder
anonymous-20071225-mnx      12/27/2007 11:46 AM    File folder
anonymous-20071227-ibt      12/31/2007 11:40 AM    File folder
anonymous-20071228-eqs      12/31/2007 11:35 AM    File folder
anonymous-20080102-hrx      1/2/2008 11:42 AM      File folder
anonymous-20080117-jkd      1/18/2008 11:38 AM     File folder
anonymous-20080117-wqy      1/18/2008 11:42 AM     File folder
```

*FIGURE 4.2 : ANONYMOUS SPEAKERS IN THE DATASET*

## 4.3 FEATURE EXTRACTION

The first step after loading the data is to convert the actual dataset that can be used by our machine learning model and this is where the processing step comes into play. There are multiple steps involved in the processing of data. We will discuss these steps in detail here. When dealing with audio signal, the data cannot be directly input to the machine learning model as it will not make sense to the model. Therefore, the audio signal has to be converted into numerical data that makes sense to the machine learning model.

As discussed in the aforementioned section, the next obvious step would be to convert audio signal into numerical data. This process is referred to as feature extraction. This includes extracting features from the audio signals and compiling them into a tensor or an array corresponding to that audio signal. When we talk about sound, the basic features that come to mind would be the pitch, loudness, shrillness etc. And to think about it, these actually are the features that help humans differentiate voices of different people. Therefore, the feature extraction step would include extracting these features from the audio data.

In the previous section, the concept of Mel Frequency Cepstral Coefficients (MFCCs for short) was introduced and it has been made clear that these are the features that encapsulate all the features mentioned above and therefore, it makes sense that MFCC features be extracted for each of the audio signals. We have discussed in brief how the MFCC features are to be calculated. We discuss the same process in detail here.

### *4.3.1 Framing and Windowing*

Since the audio signal varies with time , it needs to be examined over a sufficiently short time period. In other words, each of the original audio signals in the dataset are about 3-4 seconds long and since the speech signal is time varying, it cannot be analysed as a whole. Therefore, the signal needs to be divided into shorter segments or frames. The signal can be considered stationary if the frames are sufficiently small. The signal is divided into 25 ms segments with an overlap of about 15 ms. An overlap needs to be maintained to preserve the characteristics of the speech signal. On each of the frames, a window is applied to taper the signal towards the frame boundaries. A Hamming window is used to enhance the harmonics, smooth the edges and to reduce the edge effects.

### *4.3.2 Fourier Spectrum*

Each windowed frame needs to be converted from a time to frequency spectrum using the Discrete Fourier Transform. Fast Fourier Transform algorithm is used to calculate the Discrete Fourier Transform.

$$X(k) = \sum_{n=0}^{n=N-1} x(n)e^{-\frac{j2\pi nk}{N}} \qquad 0 \le k \le N-1$$

where $N$ is the number of points used to compute the DFT.

## *4.3.3 Mel Spectrum*

A set of triangular band-pass filters are generated known as the Mel-filter bank. The Mel spectrum can be computed by passing the Fourier transformed signal through the Mel-filter bank. The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. The formula for converting into from frequency to Mel scale is given below:

$$M(f) = 1125\ln(1 + \frac{f}{700})$$

where *f* denotes the physical frequency and *M(f)* refers to the perceived frequency.

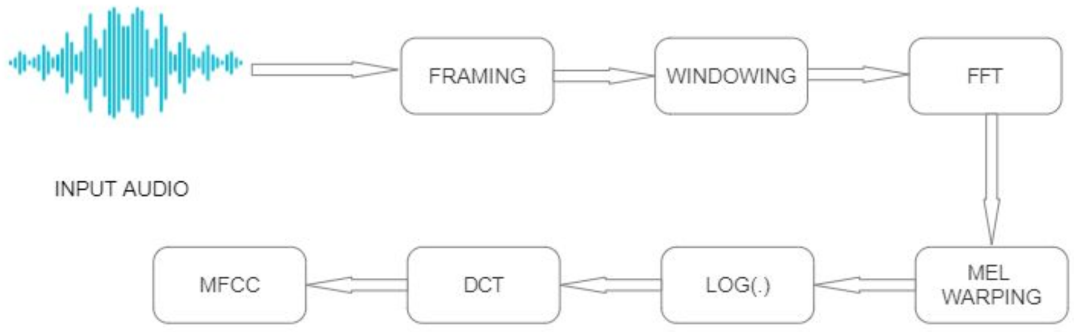


*FIGURE 4.3 : MFCC CALCULATION ALGORITHM*

## *4.3.4 DCT*

Discrete Cosine Transform or DCT is applied to transformed Mel frequency coefficients to produce a set of cepstral coefficients. Traditionally, MFCC calculation involves calculating 8-13 MFCC features. However, in our case 20 MFCC features per frame are calculated.

### *4.3.5 Padding*

Since each of the audio signals is not necessarily of the same length, the number of frames for each audio signals might not be the same. To make each feature vector of similar size, padding is required. Since the number of frames per audio signals might be different, the frames are padded to 196 i.e. each audio has a maximum of 196 frames and each of these frames has 20 MFCC coefficients and therefore, each feature vector is of size *20 X 196.* Since the convolutional layer requires a three dimensional input, the feature vector dimensions needs to be expanded using *numpy.expand_dims().*

## 4.4 MODEL CREATION

Before moving ahead with the actual model architecture, we need to understand the underlying components of the layers involved. The two activation functions that are employed in the internal layers are discussed here .

### *4.4.1 ReLu*

Rectified Learning unit is one of the most commonly used activation functions in deep learning models. The function gives a value of 0 for negative input values and changes linearly for positive input values. ReLu can be defined by the function as:

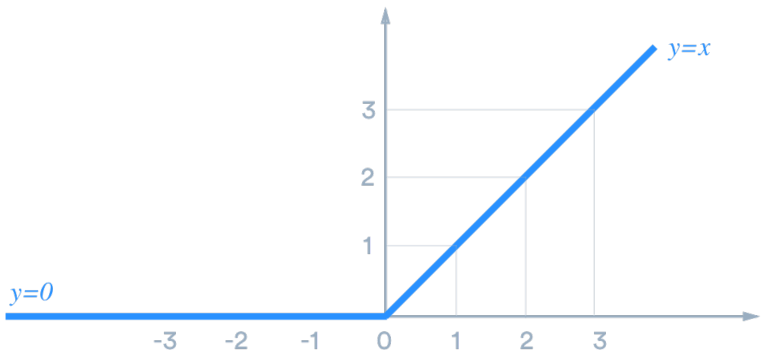


*FIGURE 4.4 : RELU ACTIVATION FUNCTION*

### *4.4.2 SoftMax*

Softmax activation is typically applied to the output layer of the neural network and is given by the formula:

$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Where N is the number of classes, z is the input vector and *σ(z)* is the output class probability.

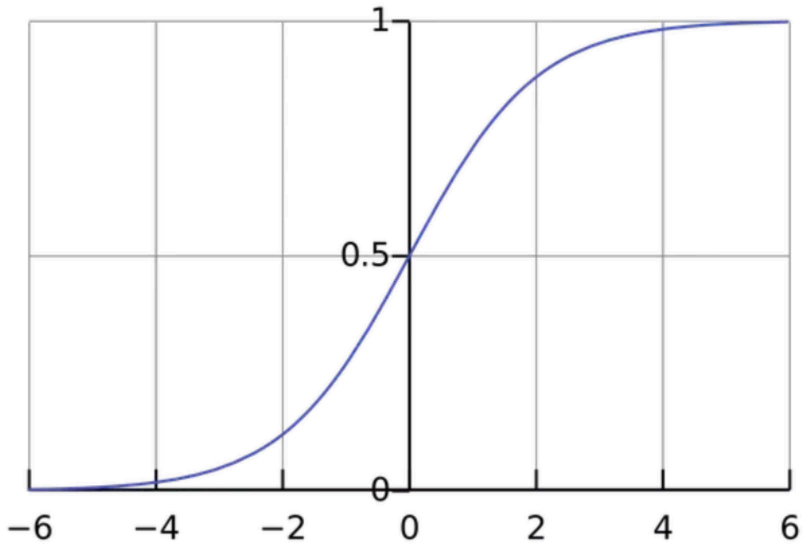


*FIGURE 4.5 : SOFTMAX ACTIVATION FUNCTION*

### 4.4.3 Adam Optimiser

The optimisation algorithm used in training here is Adam. Adam optimiser is discussed in brief here. Adam stands for adaptive moment estimation. It is an extension of the stochastic gradient descent. Adam utilises the advantages of both AdaGrad and RMSProp algorithms. Instead of adapting the parameter learning rates according to the average first moment of the means like RMSProp, it adapts to the average of the second moments of the gradients (the uncentered variance).

It is observed that Adam performs slightly better than the AdaGrad and RMSProp algorithms.

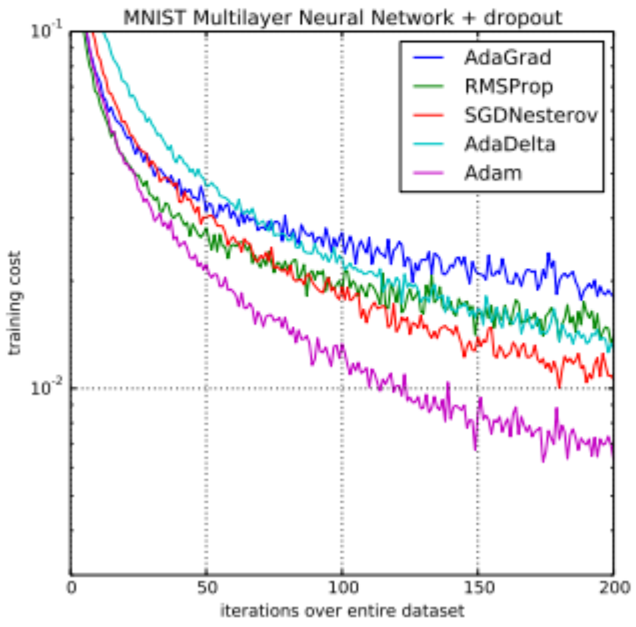
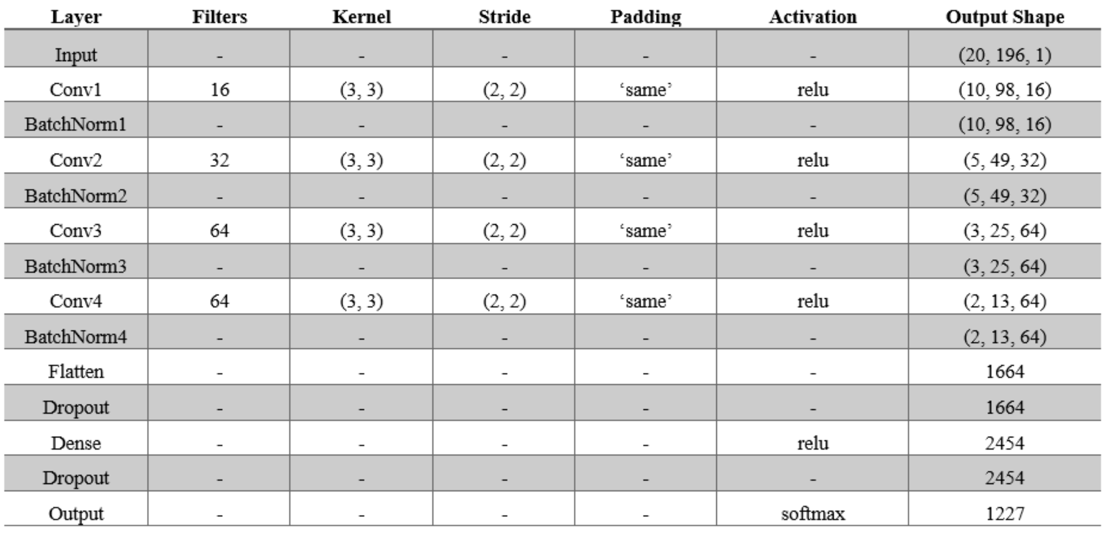


*FIGURE 4.6 : COMPARISON OF ADAM OPTIMISER WITH OTHER ALGORITHMS*

### *4.4.4 Model*

In line with the general architecture of a convolutional neural network discussed in the previous section, the CNN model that we employ consists of stacked convolutional and dense layers to produce the final result. However, drawing conclusions from [8] we decided to replace max-pooling layers with strided convolutional layers. This was done because an increase in model performance, if not very significant, was observed on re placing max-pooling layers with strided convolutional layers [11]. The activations and shapes of these layers have been mentioned in Table 1. Each convolutional block also contains a batch normalisation layers to normalise and scale all inputs to the next layer [10]. There are four convolutional blocks followed by a *Flatten* layer and a *Dense* layer. The optimiser used is Adam with learning rate set at 0.0002 and momentum set at 0.5. The whole architecture can be found in the table.



| Layer | Filters | Kernel | Stride | Padding | Activation | Output Shape |
|---|---|---|---|---|---|---|
| Input | - | - | - | - | - | (20, 196, 1) |
| Conv1 | 16 | (3, 3) | (2, 2) | 'same' | relu | (10, 98, 16) |
| BatchNorm1 | - | - | - | - | - | (10, 98, 16) |
| Conv2 | 32 | (3, 3) | (2, 2) | 'same' | relu | (5, 49, 32) |
| BatchNorm2 | - | - | - | - | - | (5, 49, 32) |
| Conv3 | 64 | (3, 3) | (2, 2) | 'same' | relu | (3, 25, 64) |
| BatchNorm3 | - | - | - | - | - | (3, 25, 64) |
| Conv4 | 64 | (3, 3) | (2, 2) | 'same' | relu | (2, 13, 64) |
| BatchNorm4 | - | - | - | - | - | (2, 13, 64) |
| Flatten | - | - | - | - | - | 1664 |
| Dropout | - | - | - | - | - | 1664 |
| Dense | - | - | - | - | relu | 2454 |
| Dropout | - | - | - | - | - | 2454 |
| Output | - | - | - | - | softmax | 1227 |

*TABLE 4.1 : CNN MODEL ARCHITECTURE*

## 4.5 SETTING UP WEIGHTS AND BIASIS

Weights and Biases is a metric tracking and model evaluation tool that we have employed here. Weights and Biases is primarily used for continuous and real-time tracking of performance metrics such as the accuracy and the categorical cross entropy loss. It is also used to draw correlations between the different aspects of training process and their effect on the model performance. However, before the tool gives us results regarding the training process, Weights & Biases needs to be set up properly first.

The *wandb* library helps with setting up the Weights & Biases environment. *wandb.init()* is used to initialise a new project. The config file has to be shared with the project to convey the parameters of the project. The config is built in the form of a JSON object and updated in the project. Once the config is set up, the project is ready for metric recording and it does so automatically once the training process starts.

## 4.6 TRAINING AND TESTING

By now, the data has been loaded and processed. The features have been extracted and training and testing data-frames have been created. The convolutional neural network model has been set up and initialised and is ready for training. The total dataset is split into 80-20 training and testing portions. *model.fit()* starts the training process. Since the machine on which training is being done doesn't have a GPU, the training process is long and arduous. The model takes about two and a half hours to train initially. To more clearly present the training process, different resource utilisation graphs have been added below.

It should also be kept in mind that several runs were performed by changing the hyper-parameters of the convolutional neural network and updating the configs of the W&B project. The results and metrics for all of these runs were successfully recorded and compiled and will be discussed in detail in the Results section.

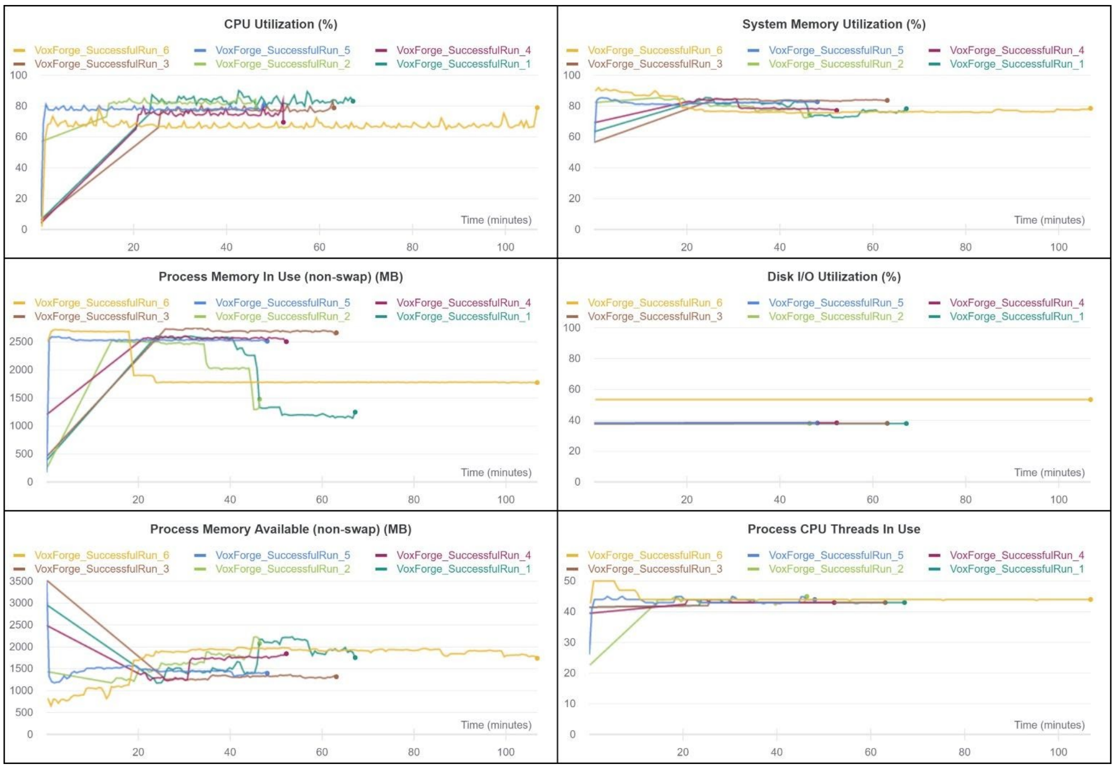


*FIGURE 4.7 : DIFFERENT METRICS COMPARISON FOR 6 CYCLES*

# 5. RESULTS

## 5.1 VOXFORGE

The training accuracies and categorical cross entropy losses have been recorded and plotted as can be seen in Figure 7 and Figure 8 respectively. Model can be seen converging for all six successful runs and the losses and accuracies for the same have been recorded and tabulated in Table 2. The maximum training accuracy that has been achieved is 97.80%.

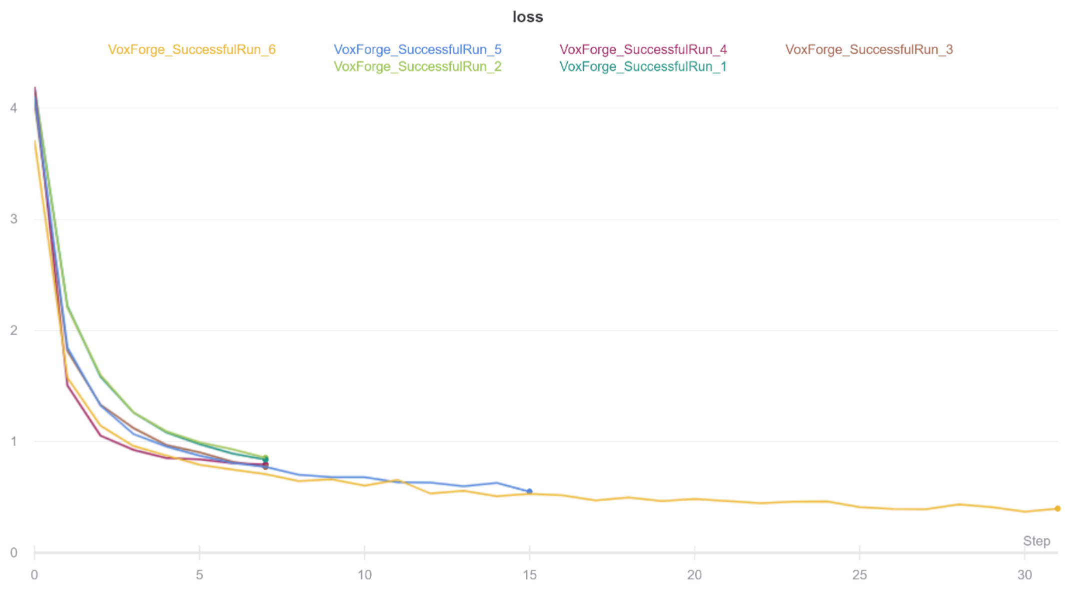


*FIGURE 5.1 : COMPARISON OF "LOSS" METRIC FOR 6 RUNS (VOXFORGE)*

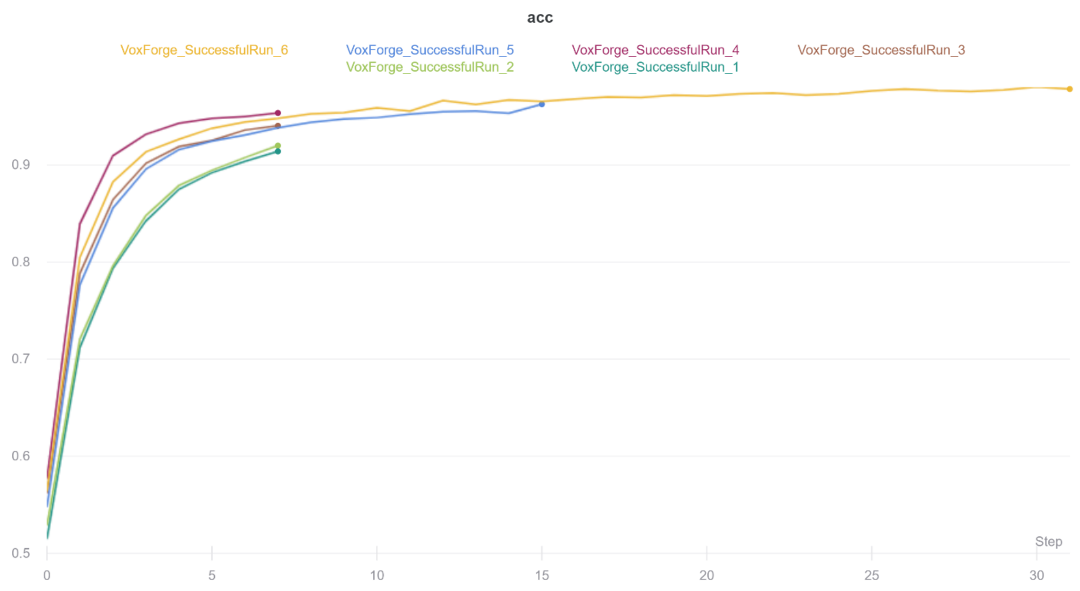


*FIGURE 5.2 : COMPARISON OF "ACCURACY" METRIC FOR 6 RUNS (VOXFORGE)*

However, the actual performance of the model can only be tracked by evaluating the model on unseen samples and seeing how well it generalises. The model is evaluated using the formula:

$$\% \textit{ of correct recognitions} = \frac{\# \textit{ of correctly identified samples}}{\textit{total } \# \textit{ of samples}} \times 100\%$$

The research paper cited as no. [12], using Deep Belief Networks to model the statistical features of sound waves could achieve an accuracy of 95%.

Another paper with the same objective cited as no. [7] in references, used spectrogram images of audio clips are used as an input and applied CNN on it to get an average accuracy of 95.83%.

The model presented in this paper is tested on over 13000 previously unseen samples from the 1227 speakers and an accuracy of 96.95% was achieved.

In a machine learning model such as a neural network, optimisation can be achieved either through changing the bias/weights attached to each node, or by varying the parameters that control the learning process. The former is achieved by the model itself in the training phase, the latter requires domain knowledge to find the accuracy optima. For a neural network these parameters consist of number of epochs and batch size. The activation function type could also be varied.

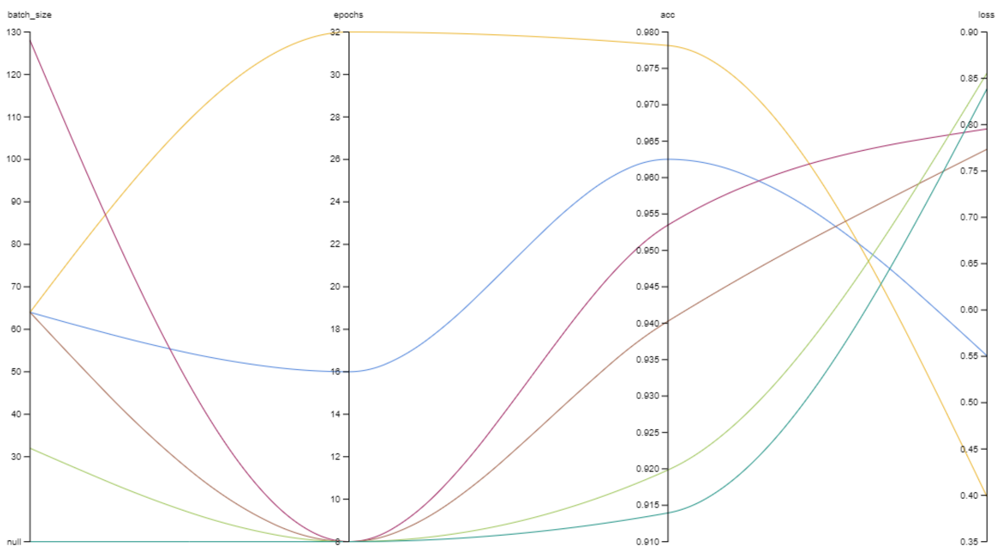


*FIGURE 5.3 : COMPARISON OF "BATCH SIZE" , "EPOCHS" , "ACCURACY" AND "LOSS" METRICS FOR ALL 6 RUNS*

The end goal of the process being to reduce variance and bias. For most intents and purposes, hyper-parameter tuning can be done through trial and error by a domain expert, however, for much larger data processing tasks, algorithms to find the optimal hyper-parameters can be used. These algorithms would act as loops on the model itself.

A hit and trial method has been tried for hyper-parameter tuning and the hyper-parameters have been randomly updated and the results for different runs have been compiled and mentioned in Table 5.1.

| Runs | Batch Size | Epochs | Number of Classes | Accuracy | Loss |
|---|---|---|---|---|---|
| VoxForge_SuccessfulRun_1 | 32 | 8 | 1227 | 91.39% | 0.8388 |
| VoxForge_SuccessfulRun_2 | 32 | 8 | 1227 | 91.98% | 0.8557 |
| VoxForge_SuccessfulRun_3 | 64 | 8 | 1227 | 94.03% | 0.7735 |
| VoxForge_SuccessfulRun_4 | 128 | 8 | 1227 | 95.35% | 0.7954 |
| VoxForge_SuccessfulRun_5 | 64 | 16 | 1227 | 96.25% | 0.5505 |
| VoxForge_SuccessfulRun_6 | 64 | 32 | 1227 | 97.80% | 0.4355 |

*TABLE 5.1 : COMPARISON OF DIFFERENT METRICS FOR ALL 6 RUNS*

## 5.2 TIMIT

The TIMIT database comprises audio samples from 630 speakers and the models is trained over 5040 samples and tested over 1260 samples. The training and testing split has already been done in the dataset itself. Each speaker has exactly 8 training samples and 2 unseen test samples for each speaker.

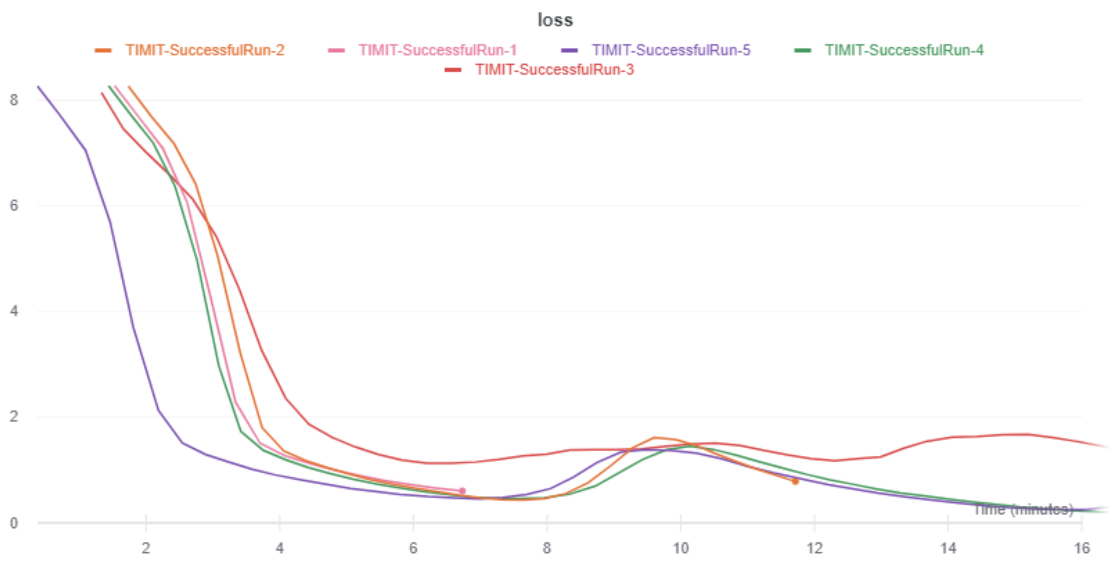


*FIGURE 5.4 : COMPARISON OF " LOSS" METRIC (TIMIT)*

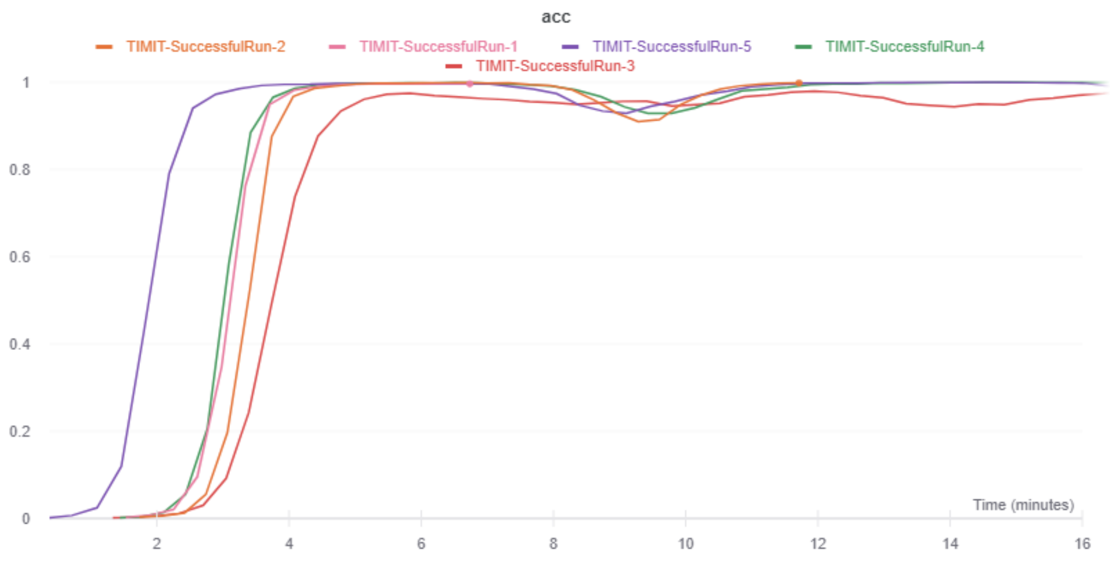


*FIGURE 5.5 : COMPARISON OF "ACCURACY" METRIC (TIMIT)*

Various successful runs have been performed for different hyperparameters. A total of 5 successful runs have been performed. The accuracy and categorical cross entropy loss while training the convolutional neural network have been logged and plotted and can be seen in Figure 10 and 11. However, the actual accuracy of the model can be predicted only by testing in on the unseen test samples. The test accuracy is again calculated according to the formula:

$$\%\ of\ correct\ recognitions = \frac{\#\ of\ correctly\ identified\ samples}{total\ \#\ of\ samples} \times 100\%$$

The test accuracy achieved on the unseen test samples in TIMIT database is 98.40% which is slightly better than the one achieved for the VoxForge dataset. However, this can be expected and explained since the total number of speakers in VoxForge database (1227) is almost the double of the number of speakers in the TIMIT database (630). Also, the total number of test samples in VoxForge dataset is about 13000 while the number of unseen test samples in the TIMIT database is only 1260.

# 6. CONCLUSION

The experiment proposed in this thesis, however successful, does make a few assumptions that need to be addressed in future applications. One of these assumptions is that the audio files received are, for the most part, free of noise. However, this won't be the case in an industrial application and therefore, the need for a noise removal algorithm is highly advisable for future applications.

Another assumption made while performing the experiments mentioned in this thesis is the availability of huge amounts of data. It is to be noted that multiple training segments are available for each of the speakers in the experiment proposed. However, this might not always be the case since the system might need to perform on very few (sometimes, even single) samples. Therefore, further applications should consider one-shot or few-shot algorithms to counter this problem. One of the approaches might be to use Siamese neural networks for few-shot speaker recognition.

A 96.95% accuracy in the VoxForge dataset and a 98.4% accuracy for the TIMIT database still means that a significant amount of false results shall be obtained for the model to be deployed in industrial usage. So, what can be done is a hyperparameter space search to receive optimal hyperparameters that, in turn, provide us with the best possible model. However, this approach does require a substantial amount of computing power and even then, we might have to wait for a long time (hours or even days) to get the optimal hyperparameter set.

However, accuracies of 96.95% on VoxForge and 98.4% on TIMIT database not only warrant research into the field of speaker recognition but also suggest the definite possibility that speaker recognition, in its improved version, can be used as a standalone biometric verification system.The experiment proposed in this thesis, however successful, does make a few assumptions that need to be addressed in future applications.

# 7. REFERENCES

[1] Furui, Sadaoki. (1996). An Overview of Speaker Recognition Technology. 43. 10.1007/978-1-4613-1367-0_2.

[2] M S, Sinith & Salim, Anoop & Sankar, K. & Nar yanan, K. & Soman, Vishnu. (2010). A novel method for Text-Independent speaker identification using MFCC and GMM. 292 - 296. 10.1109/ICALIP.2010.5684389.

[3] M S, Sinith & Salim, Anoop & Sankar, K. & Narayanan, K. & Soman, Vishnu. (2010). A novel method for Text-Independent speaker identification using MFCC and GMM. 292 - 296. 10.1109/ICALIP.2010.5684389.

[4] Mahboob, Tahira & Khanam, Memoona & Khiyal, Malik & Bibi, Ruqia. (2015). Speaker Identification Using GMM with MFCC. International Journal of Computer Science Issues. 12. 126-135.

[5] Santosh, K.Gaikwad & Bharti, W.Gawali & Yannawar, Pravin. (2010). A Review on Speech Recognition Technique. International Journal of Computer Applications. 10. 10.5120/1462-1976.

[6] Hasan, Md & Jamil, Mustafa & Rabbani, Golam & Rahman, Md. Saifur. (2004). Speaker Identification Using Mel Frequency Cepstral Coefficients. Proceedings of the 3rd International Conference on Electrical and Computer Engineering (ICECE 2004).

[7] S. Bunrit, T. Inkian, N. Kerdprasop & K. Kerdprasop (2019). Text-Independent Speaker Identification Using Deep Learning Model of Convolution Neural Network. International Journal of Machine Learning and Computing, Vol. 9, No. 2, April 2019.

[8] Douglas A. Reynolds & Richard C. Rose (1995). Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models. IEEE Transactions on Speech and Audio Processing 3(1). 72 – 83.

[9] Douglas A. Reynolds (1995). Speaker Identification and Verification Using Gaussian Mixture Speaker Models. Speech Communication, vol. 17, no. 1-2, pp. 91–108, 1995.

[10] Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[11] Springenberg, Jost & Dosovitskiy, Alexey & Brox, Thomas & Riedmiller, Martin. (2014). Striving for Simplicity: The All Convolutional Net.

[12] A. Banerjee, A. Dubey, A. Menon, S. Nanda & G.C. Nandi.Speaker Recognition Using Deep Belief Networks to CCIS proceedings.

[13] S. Bhardwaj, S. Srivastava, M.Hanmandlu, J.R.P.Gupta. GFM Based Methods for Text Independent Speaker Identification. IEEE Transactions on Systems, Man, and Cybernetics, Part B,vol. 43,no.3 , pp.1047-1058, 2013 DOI: 10.1109/ TSMCB.2012.2223461.

## 8. Contact Details of Authors

**Pranjal Naman**

Contact Number : 9871402280

Email ID : naman4697@gmail.com

**Raunaq Bhalla**

Contact Number : 9999904175

Email ID : raunaq.bhalla98@gmail.com

**Satyarth Vats**

Contact Number : 9654015236

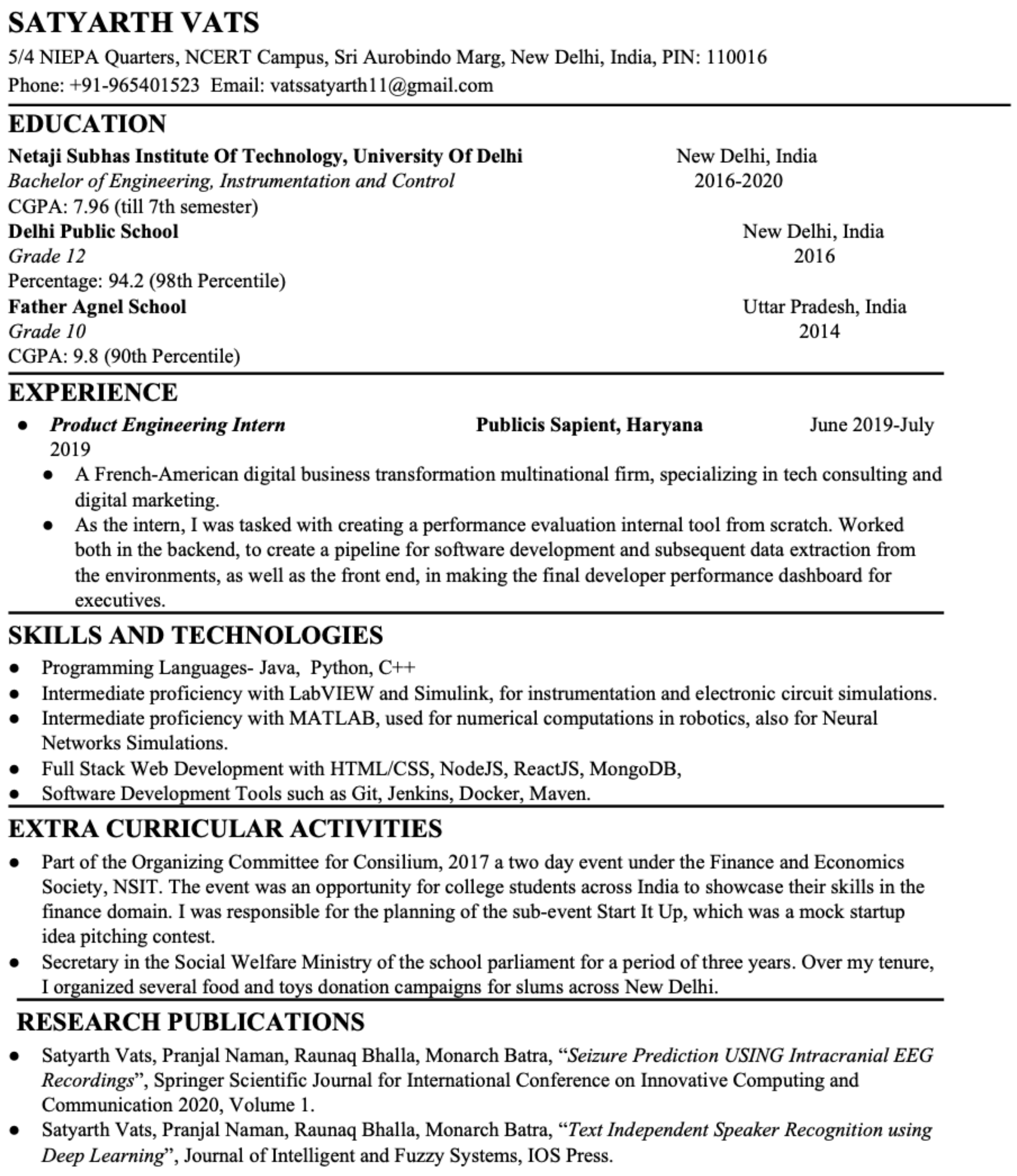
Email ID : vatssatyarth11@gmail.com

**Monarch Batra**

Contact number : 7042634345

Email ID : batramonarch13@gmail.com

## 9. Brief curriculum vitae



**SATYARTH VATS**

5/4 NIEPA Quarters, NCERT Campus, Sri Aurobindo Marg, New Delhi, India, PIN: 110016
Phone: +91-965401523 Email: vatssatyarth11@gmail.com

**EDUCATION**

**Netaji Subhas Institute Of Technology, University Of Delhi** New Delhi, India
*Bachelor of Engineering, Instrumentation and Control* 2016-2020
CGPA: 7.96 (till 7th semester)
**Delhi Public School** New Delhi, India
*Grade 12* 2016
Percentage: 94.2 (98th Percentile)
**Father Agnel School** Uttar Pradesh, India
*Grade 10* 2014
CGPA: 9.8 (90th Percentile)

**EXPERIENCE**

- ***Product Engineering Intern*** **Publicis Sapient, Haryana** June 2019-July 2019
  - A French-American digital business transformation multinational firm, specializing in tech consulting and digital marketing.
  - As the intern, I was tasked with creating a performance evaluation internal tool from scratch. Worked both in the backend, to create a pipeline for software development and subsequent data extraction from the environments, as well as the front end, in making the final developer performance dashboard for executives.

**SKILLS AND TECHNOLOGIES**

- Programming Languages- Java, Python, C++
- Intermediate proficiency with LabVIEW and Simulink, for instrumentation and electronic circuit simulations.
- Intermediate proficiency with MATLAB, used for numerical computations in robotics, also for Neural Networks Simulations.
- Full Stack Web Development with HTML/CSS, NodeJS, ReactJS, MongoDB,
- Software Development Tools such as Git, Jenkins, Docker, Maven.

**EXTRA CURRICULAR ACTIVITIES**

- Part of the Organizing Committee for Consilium, 2017 a two day event under the Finance and Economics Society, NSIT. The event was an opportunity for college students across India to showcase their skills in the finance domain. I was responsible for the planning of the sub-event Start It Up, which was a mock startup idea pitching contest.
- Secretary in the Social Welfare Ministry of the school parliament for a period of three years. Over my tenure, I organized several food and toys donation campaigns for slums across New Delhi.

**RESEARCH PUBLICATIONS**

- Satyarth Vats, Pranjal Naman, Raunaq Bhalla, Monarch Batra, "*Seizure Prediction USING Intracranial EEG Recordings*", Springer Scientific Journal for International Conference on Innovative Computing and Communication 2020, Volume 1.
- Satyarth Vats, Pranjal Naman, Raunaq Bhalla, Monarch Batra, "*Text Independent Speaker Recognition using Deep Learning*", Journal of Intelligent and Fuzzy Systems, IOS Press.

# RAUNAQ BHALLA

B-172,Belvedere Towers, DLF Phase-2, Gurugram, Haryana 122002, India
Phone: +91-9999904175 Email**:** raunaq.bhalla98@gmail.com

## EXPERIENCE

Treasury Intern **Hindustan Media Group, Haryana** June 2018 - July 2018
*One of the leading print media companies in India, with various online ventures and approximately $360 million in revenue.*

- As part of the research team within the treasury department, evaluated the interdependence of gold prices and national budget on the National Stock Exchange using past trends and came up with insights regarding the same.
- Discussed the need to steer the investment strategy due to the early signs of a Bearish market as indicated by the extreme volatility in gold prices at that time. Recommendations were directly absorbed and the modified strategy increased profits in the stock market by approximately $200,000 in the subsequent weeks.

Supply Chain Management Intern **Design Elements, New Delhi** November 2017 - December 2017
*A Twelve year old leading garment manufacturer for Fabindia, a major Indian ethnic garment retail chain, with over 120 employees and $700,000 in turnover.*

- Enhanced efficiency of the manufacturing plant through layout changes. Reduced cycle time for producing finished garments from raw materials by approximately 10%.
- Recommendations drove manufacturing cost down from $3.50 to $2.75 per garment.

## EDUCATION

**Netaji Subhas Institute of Technology, University of Delhi** New Delhi, India
*Bachelor of Engineering in Instrumentation and Control Engineering* 2016 - 2020
CGPA: 7.7/10

**London School of Economics and Political Science** London, England
Strategic Management Summer Program June 2017 - July 2017
Grade : A; Top 15% in a class of 238

**The Indian School** New Delhi,India
Grade 12 2016
Percentage : 95.2%; Top 2% in the country with over 1.04 million test takers

**The Indian School** New Delhi,India
Grade 10 2014
CGPA : 10/10; Top 11% in the country with over 1.32 million test takers

## ACHIEVEMENTS

- Published a research paper, "Seizure Prediction using Intracranial EEG Recordings" in the Springer Scientific Journal,ICICC 2020, Volume 1. Analysed Preictal segment of EEG data of five canines and two humans, using Fast Fourier transform and Logistic Regression model, to predict the next seizure cycle
- Co-authored a research paper, "Text Independent Speaker Recognition using Deep Learning"; under review at Journal of Intelligent and Fuzzy Systems, IOS Press.
- Secured a position in the top 5 percentile amongst 1.2 million students appearing for the JEE Mains examination. (engineering entrance examination)
- Awarded Certificate of Excellence in Academics for performance in grade 12 for scoring in the top 2 percentile of students across the country.

## EXTRA CURRICULAR ACTIVITES

- Assistant Placement Coordinator at the NSIT Placement Cell - Provided logistics support and coordinated with 50+ recruiting firms to aid in internship and job opportunities for the 1500+ strong class. Recorded the highest number of internship offers for pre-final year students in the past five years.
- Junior Coordinator at Crosslinks, NSIT - the school's official media management society. Led multiple teams of 50 students each in domains like PR, Logistics etc to organise numerous official college events as well as cultural festivals, that saw participation from over 100+ colleges and 8,000+ spectators.
- Head Boy, The Indian School - Led school's student council and acted as an intermediary between students and administration.
- Sports Vice Captain, The Indian School - Led the entire sports department of the school and organised various inter-school sports events. Captained the school's soccer and athletics team.
- Secured 2nd position in inter-college and 4th position in Delhi state powerlifting competitions with 20+ competitors in each.

## COMPUTER LANGUAGES AND TOOLS USED

- R - Used packages such as dplyr, tidyr, cluster and ggplot2 for data manipulation and visualisation.
- Python - Used libraries such as Sklearn, Pandas and Keras to run predictive analysis of EEG data.
- MATLAB - Simulated engineering models of Induction Motors, PID Controllers etc.
- LabView - Simulated Digital Signal Processing components such as filters and Fourier Transform.

# MONARCH BATRA

## ACADEMIC QUALIFICATIONS

| Examination | Year | Board/University | %/CGPA |
|---|---|---|---|
| **Bachelor of Engineering** (Instrumentation and Control) | 2016-2020 | Netaji Subhash Institute of Technology, Dwarka, New Delhi | 8.8 CGPA (till 7th semester) |
| **Class XII** | 2016 | Bal Bhavan International School, New Delhi (C.B.S.E) | 95.8% |
| **Class X** | 2014 | Bal Bhavan International School, New Delhi (C.B.S.E) | 10 CGPA |

## PROFESSIONAL COURSES

- Attended Arduino Workshop held at NSIT, New Delhi.
- Attended MATLAB and Lab View Workshop organised by National Instruments at NSIT, New Delhi.
- Data Structures and Algorithm in C++ from CODING BLOCKS, New Delhi.

## INTERNSHIPS

- **Manufacturing Intern | Reliance Industries Ltd | Jamnagar, India** **May'19 - Jul'19**
    - Observed and studied the different types of Barriers and Isolators in the Central Engineering section of the DTA refinery in the Jamnagar Manufacturing division.
    - Visited Plant Interface Blocks and Plant Control Block, and got an insight of Distributed Control System and Emergency Shutdown System.

## COURSES Of BACHELORS

- Electrical and Electronics Engineering
- Electronics
- Control System
- Industrial Electronics
- Microprocessor and Microcontroller
- Transducer and Measurement
- Electronic Instrumentation

## ACADEMIC DISTINCTIONS

- Top 5% Merit Scholarship awarded by college for the year 2016-2017.
- Secured position in top 5 percentage of students appearing in JEE Mains examination.
- Awarded Certificate of Merit by C.B.S.E for securing 10 CGPA in Class 10.

## EXTRA CURRICULAR ACTIVITIES

- Captained the school's volleyball team for 4 years and won various events and championships.
- Secured 1st position in Zonal level Science Exhibition Competition in 2015.

batramonarch13@gmail.com Netaji Subhas Institute of Technology, New Delhi +(91)-7042634345

# PRANJAL NAMAN

## ACADEMIC QUALIFICATIONS

| Examination | Year | Board/University | %/CGPA |
|---|---|---|---|
| **Bachelor of Engineering** (Instrumentation and Control) | 2016-2020 | Netaji Subhash Institute of Technology, Dwarka, New Delhi | 9.04 CGPA (till 7th semester) |
| **Class XII** | 2016 | Delhi Public School, R.K. Puram (C.B.S.E) | 94.8% |

## PROFESSIONAL COURSES

- Programming Languages- Java, Python, C++
- Intermediate proficiency with LabVIEW and Simulink, for instrumentation and electronic circuit simulations.
- Intermediate proficiency with MATLAB, used for numerical computations in robotics, also for Neural Networks Simulations.
- Full Stack Web Development with HTML/CSS, NodeJS, ReactJS, MongoDB,

## INTERNSHIPS

- **Project Intern | Nucleus Software** **May'19 - Jul'19**
  - As the intern, I was tasked with creating a performance evaluation internal tool from scratch. Worked both in the backend, to create a pipeline for software development and subsequent data extraction from the environments, as well as the front end, in making the final developer performance dashboard for executives.

## COURSES Of BACHELORS

- Electrical and Electronics Engineering
- Electronics
- Control System
- Industrial Electronics
- Microprocessor and Microcontroller
- Transducer and Measurement
- Electronic Instrumentation

## RESEARCH PUBLICATIONS

- Pranjal Naman, Satyarth Vats, Raunaq Bhalla, Monarch Batra, "Seizure Prediction USING Intracranial EEG Recordings", Springer Scientific Journal for International Conference on Innovative Computing and Communication 2020, Volume 1.
- Pranjal Naman, Satyarth Vats, Raunaq Bhalla, Monarch Batra, "Text Independent Speaker Recognition using Deep Learning", Journal of Intelligent and Fuzzy Systems, IOS Press.

## EXTRA CURRICULAR ACTIVITIES

- Assistant Placement Coordinator at the NSIT Placement Cell - Provided logistics support and coordinated with 50+ recruiting firms to aid in internship and job opportunities for the 1500+ strong class. Recorded the highest number of internships offers for pre-final year students in the past five years.
- Part of the Organizing Committee for Consilium, 2017 a two-day event under the Finance and Economics Society, NSIT. The event was an opportunity for college students across India to showcase their skills in the finance domain. I was responsible for the planning of the sub-event Start It Up, which was a mock startup idea pitching contest.

naman4697@gmail.com Netaji Subhas Institute of Technology, New Delhi +(91)- 9871402280

# APPENDIX

```
import argparse
import os
from pathlib import Path
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
import matplotlib
import matplotlib.pyplot as plt
import librosa
import librosa.display
from sklearn.model_selection import train_test_split
import wandb
from wandb.keras import WandbCallback
from sklearn.metrics import confusion_matrix
from ann_visualizer.visualize import ann_viz
import json

#--------------------------------------------------------------------------#
#GLOBAL VARIABLES FOR THE PROJECT
#--------------------------------------------------------------------------#

voxforge_rootpath = 'C:/Users/Pranjal/Downloads/voxforge/voxforge/'
listofnames = os.listdir(voxforge_rootpath)
list_ds = []
speakerlist = []
labelencoder = preprocessing.LabelEncoder()
```

```
X = []
y = []
X1 = []
y1 = []
X_train = []
y_train = []
X_test = []
y_test = []
input_shape = (20, 196, 1)

num_conv_layers = 0
num_fc_layers = 0

#----------------------------------------------------------------------------#
#INITIALIZING NECESSARY GLOBAL VARIABLES
#----------------------------------------------------------------------------#
#----------------------------------------------------------------------------#
#NECESSARY FUNTIONS
#----------------------------------------------------------------------------#

def extract_speaker(file_path):
    sc = file_path.split('/')[-3]
    return sc.split('-')[0]

def loadConvMFCC(file_path, max_pad_len=196):
    wave, sample_rate = librosa.load(file_path, mono=True, sr=None)
    mfcc = librosa.feature.mfcc(wave, sample_rate)
    mfcc = mfcc[:, :max_pad_len]
    pad_width = max_pad_len - mfcc.shape[1]
    mfcc = np.pad(mfcc, pad_width=((0, 0), (0, pad_width)), mode='constant')
    return mfcc



def extract_mfcc(file_path):
    file_name = file_path
    mfcc = loadConvMFCC(file_name)
    mfcc = np.expand_dims(mfcc, 2)
    return mfcc

def convToJson(unique_speakers, labelencoder):
    speakerDict = {}
    for i in range(unique_speakers):
        speakerDict[i] = labelencoder.inverse_transform([i])[0]
        speakerDict[speakerDict[i]] = i
    print(speakerDict)
    out = open("C:/Users/Pranjal/Desktop/speakerJSON.json", "w")
    json.dump(speakerDict, out)
    out.close()

#--------------------------------------------------------------------------#
#SOME BASIC OPERATIONS
#--------------------------------------------------------------------------#

def preprocessing(encodedspeakerlist):
    for name in listofnames:
        if 'anonymous' not in name:
            voxforge_userpath = voxforge_rootpath + name + '/wav/'
            try:
```

```
            for file in os.listdir(voxforge_userpath):
                list_ds.append(voxforge_userpath + file)
        except:
            continue



 for file in list_ds:
      speakerlist.append(extract_speaker(file))

   encodedspeakerlist = labelencoder.fit_transform(speakerlist)
   print(encodedspeakerlist)

   for i in range(len(list_ds)):
      try:
         print(str(i) + '/ ' + str(len(list_ds)))
         feature = extract_mfcc(list_ds[i])
         print(feature.shape)
         X.append(feature)
         y.append(encodedspeakerlist[i])
      except:
         continue



   return len(set(encodedspeakerlist))
#---------------------------------------------------------------------------#
#MODEL CREATION FUNCTION
#---------------------------------------------------------------------------#

def create_model(unique_speakers):
   dropout_rate = .25
   regularazation = 0.001
   audio_input = keras.layers.Input(shape=input_shape)
   conv1 = keras.layers.Conv2D(16, kernel_size=(3, 3), padding='same',
                      activation='relu', input_shape=input_shape)(audio_input)
   maxpool1 = keras.layers.MaxPooling2D(pool_size=(2, 2), strides=2)(conv1)
   batch1 = keras.layers.BatchNormalization()(maxpool1)
   conv2 = keras.layers.Conv2D(32, kernel_size=(3, 3), padding='same',
                      activation='relu', input_shape=input_shape)(batch1)
   maxpool2 = keras.layers.MaxPooling2D(pool_size=(2, 2), strides=2)(conv2)
   batch2 = keras.layers.BatchNormalization()(maxpool2)
   conv3 = keras.layers.Conv2D(64, kernel_size=(3, 3), padding='same',
            activation='relu')(batch2)
   maxpool3 = keras.layers.MaxPooling2D(pool_size=(2, 2), strides=2)(conv3)
   batch3 = keras.layers.BatchNormalization()(maxpool3)
   flt = keras.layers.Flatten()(batch3)
   drp1 = keras.layers.Dropout(dropout_rate)(flt)

 dense1 = keras.layers.Dense(unique_speakers * 2, activation='relu',
            kernel_regularizer=keras.regularizers.l2(regularazation))(drp1)

   drp2 = keras.layers.Dropout(dropout_rate)(dense1)
   output = keras.layers.Dense(unique_speakers, activation='softmax',

name='speaker')(drp2)
   model = keras.Model(inputs=audio_input, outputs=output)
   model.compile(loss=keras.losses.sparse_categorical_crossentropy,
             optimizer=keras.optimizers.Adam(),
             metrics=['acc'])
```

```
    return model

#--------------------------------------------------------------------------------#
#EXTRACTING PARAMETERS OF INTEREST

def log_model_params(model, wandb_config, unique_speakers):
    temp1 = 0
    temp2 = 0
    for l in model.layers:
        layer_type = l.get_config()["name"].split("_")[0]
        if layer_type == 'conv2d':
            temp1 += 1
        elif layer_type == 'dense':
            temp2 += 1
    num_conv_layers = temp1
    num_fc_layers = temp2

    wandb_config.update({
            "epochs" : 32,
            "n_conv_layers" : num_conv_layers,
            "n_fc_layers" : num_fc_layers,
            "num_classes" : unique_speakers,
            "dropout" : 0.25,
            "optimizer" : "adam",
            "batch_size" : 64
            })

#--------------------------------------------------------------------------------#
#RUNNING THE EXPERIMENT
```

```
#---------------------------------------------------------------------------#

def run_exp(model):
    #model = create_model()
    model.summary()
    callbacks = [WandbCallback()]
    model.fit(X_train, y_train, epochs=32, batch_size = 64, validation_split=0.2,

callbacks=callbacks)
    model.save('C:/Users/Pranjal/Desktop/voxforge1.h5')
    model.evaluate(X_test, y_test)
    y_pred1 = model.predict(X_test)
    y_pred = []
    for i in y_pred1:
        temp = np.where(i == np.amax(i))
        y_pred.append(temp[0][0])

    conf = confusion_matrix(y_true = y_test, y_pred = y_pred)
    print(conf)

#---------------------------------------------------------------------------#
#MAIN FUNCTION

def main():
    wandb.init(project="voxforge")
    encodedspeakerlist = []
    unique_speakers = preprocessing(encodedspeakerlist)
    convToJson(unique_speakers, labelencoder)
    X = np.array(X)
    y = np.array(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
    model = create_model(unique_speakers)
    log_model_params(model, wandb.config, unique_speakers)
    run_exp(model)
#---------------------------------------------------------------------------#
main()
```