A.I. VOICE ASSISTANT - THESIS

A THESIS REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

BACHELOR OF TECHNOLOGY
IN
DIVISION OF COMPUTER ENGINEERING

Submitted by:

(Roll No. 1582/CO/2017)
(Roll No. 1592/CO/2017)
(Roll No. 1598/CO/2017)
(Roll No. 1629/CO/2017)

Under the supervision of
Associate Professor Dr. Veenu

DIVISION OF COMPUTER ENGINEERING
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI

DECEMBER, 2020

Department of Computer Engineering
University of Delhi
Delhi-110007, India

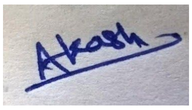
CERTIFICATE OF ORIGINALITY

We, Abhishek Singh (2017UCO1582), Akash Singh (2017UCO1592), Rahul Meena (2017UCO1598) , Ankit Arya (2017UCO1629) student(s) of B. Tech. Department of Computer Engineering, hereby declare that the Project-Thesis titled “A.I Voice Assistant” which is submitted by us to the Department of Computer Engineering, Netaji Subhas Institute of Technology, Delhi (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from source without proper citation. The manuscript has been subjected to plagiarism check by Turnitin software. This work has not previously formed the basis for the award of any Degree.

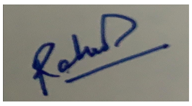
Place: Delhi (Name and signature of student(s))

Date:

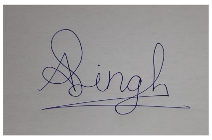
Akash Singh



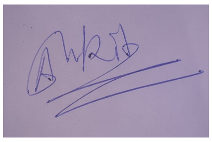
Rahul Meena



Abhishek Singh



Ankit Arya



Department of Computer Engineering
Netaji Subhas University of Technology
Dwarka, Delhi-110078, India

CERTIFICATE OF DECLARATION

This is to certify that the Project-Thesis titled " A.I Voice Assistant" which is being submitted by Abhishek Singh (2017UCO1582), Akash Singh (2017UCO1592), Rahul Meena (2017UCO1598) , Ankit Arya (2017UCO1629) to the Department of Computer Engineering, Netaji Subhas Institute of Technology (university of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the thesis work carried out by the students under my supervision and guidance. The content of this thesis, in full or in parts, have not been submitted for any other degree or diploma.

Place: Delhi (Associate Professor Dr. Veenu)

Date: SUPERVISOR

ACKNOWLEDGEMENT

We would like to begin our acknowledgement by expressing our heartiest thanks to our mentor, our guide and our goto person for all our queries regarding this project Dr. Venu. Without her insightful inputs and close monitoring this project could not have reached a stage where we could present it to our readers.Also we would acknowledge Dr. Anand Gupta, the project coordinator for the Department of Computer Sciences and Engineering for smoothly conducting the whole process and for providing us with guidelines throughout the course of the project enabling us to adhere to the norms. We would thank Dr. Bijendra Kumar, Head of Department of Computer Sciences and Engineering for providing us students an opportunity to work on this project and providing freedom to choose the domain of work. Lastly we would also like to thank all our colleagues and professors who have helped us at several instances during the course of the project. This whole project has been a great learning experience for all of us, from choosing our topic to handling internal disagreements and teamwork, we have come a long way. We hope that all our takeaways from this project shall assist us in our future endeavours too.

ABSTRACT

This thesis describes the tasks completed by us for project (I) as specified in our curriculum for the seventh semester of our B.E. course. The domain chosen by us for our project is Machine Learning and Artificial Intelligence with our topic being an 'AI Voice Assistant'.

Artificial intelligence technologies are starting to be actively utilized in human life, to carry out various activities. Devices are getting smarter in their ways to interact with both a person and among themselves. New capacities cause creation of varied systems for integration of smart things into Social Networks of the web of Things. One of the relevant trends in AI is that the technology of recognizing what a person is saying and responding accordingly and that has given birth to voice assistants. New insights within this topic can cause new means of natural human-machine interaction, during which the machine would find out how to understand a human's language, and also adjust and interact in it.

One of such tools is voice assistant, which may be integrated into many other intelligent systems. Today voice assistants are capable of responding to our voice based commands and perform tasks such as searching on the web, replying to texts, and activities to the extent of having a normal meaningful conversation with us.

In our paper, we will be explaining the principles of the functioning of an A.I. Voice Assistant and describe its main shortcomings and limitations as well. We will also be covering the future aspects, applications of our system and most importantly define how

it is different from other voice assistants that are already available in the form of Siri, Alexa etc. We will be describing The method of making an AI Voice Assistant without using cloud services and thus no user data being collected is described, which allows us to significantly expand the applicability of such devices within the future.

In this Project we have worked to make a voice assistant that uses Machine Learning for its working. Our voice Assistant can be divided into 4 different components

1. Wake word detection
2. Automatic speech Recognition
3. Natural Language Understanding
4. Speech Synthesis

Index

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

1.1 Motivation

Artificial intelligence has made great advancement in the past few years, they are now being used in every field to tackle every kind of problem. One of these fields is voice assistant in which humans ask a machine to perform certain tasks for them and the machine does accordingly. In this field machine learning has given voice assistants human-like intelligence and they are still improving.Voice assistants are more convenient to use and they are faster as we can speak faster than type. But while using these Voice Assistants privacy is of great concern to the user, they don't know what is being done with their data. They don't know what information the voice assistant has stored and how much it knows about the user as voice assistants now a days can buy groceries, plan your evening or book your favorite movie so if there is any data is stolen or is used for wrong purpose it can harm the user financially as well as mentaily, moreover linux based operating system doesn't have a native voice assistant in them.

1.2 Key Challenges

Training a wakeword model which is resource efficient and still accurate was no trivial task and since our dataset didn't contain voices with Indian accent it was tough to make it work, so to handle this and increase accuracy somewhat we have used specagument technique.

Combining all of the Machine Learning models together and then using them synchronously presented itself as a difficult task. Took us a long time integrating python codes with html and javascript codes.

For our NLU model we didn’t find any relevant dataset as we didn’t want our dataset to be too big otherwise our model will also be big and it won’t be resource efficient. so we had to make a small dataset by using web scraping and some we wrote it ourself.

Even when using google collab for training our model on lj speech dataset took a long time and we used to run into bugs all of the time because of that we had to train from the start and we spent quite a long time doing that.

1.3 Problem addressed in the thesis

There aren’t many Voice Assistants available that are privacy friendly and easy to use moreover Linux based systems don't have an integrated voice assistant like Windows has Cortana and MacOS has Siri. The Voice Assistant that we have developed will run on the user’s own machine, no data is sent to the cloud for processing or storing. the user's data is safe with them so they don't have to worry about privacy. Everything is stored and processed on the user’s computer itself some of the functionality would work even if there is no internet and it can be used on Linux, Windows, MacOs.

1.5 Dataset used

- To train our wakeword model we used Google speech commands dataset this dataset contains 1,10,000 one-second long utterances of 35 short words one of which is “happy” which we chose as our wakeword.

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052

Table 1.1 Google speech commands dataset

- For our NLU model we got some data by scraping websites for history and some we wrote ourselves to train our model.

- To train our TextToSpeech model we used Lj Dataset which consists of 13,100 short audio clips. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours.

Statistics

Total Clips	13,100
Total Words	225,715
Total Characters	1,308,678
Total Duration	23:55:17
Mean Clip Duration	6.57 sec
Min Clip Duration	1.11 sec
Max Clip Duration	10.10 sec
Mean Words per Clip	17.23
Distinct Words	13,821

Fig 1.1 Information regarding audio clips contained in Lj Dataset

1.6 Project Overview

For our AI Voice Assistants we have identified 4 main modules:

1. **Wake word detection**
In this module we discuss what is a wakeword, how we used it in our project. What steps we took while developing it and how well it performs.

2. **Automatic speech Recognition**
This module talks about how we converted our voice into text format, so that it can be further processed.

3. **Natural Language Understanding**
In this module, we give information and detailed steps of how our voice assistant tries to make sense of what the user has said and do the task told by the user.

4.Speech Synthesis
This is the last module describing our Voice Assistant, this module is about how the voice assistant will generate sound waves and talk back to the user.

We shall dedicate one chapter for further detailed discussion of each module. Discussing about how they are preprocessed what training method did we used and what results we obtained

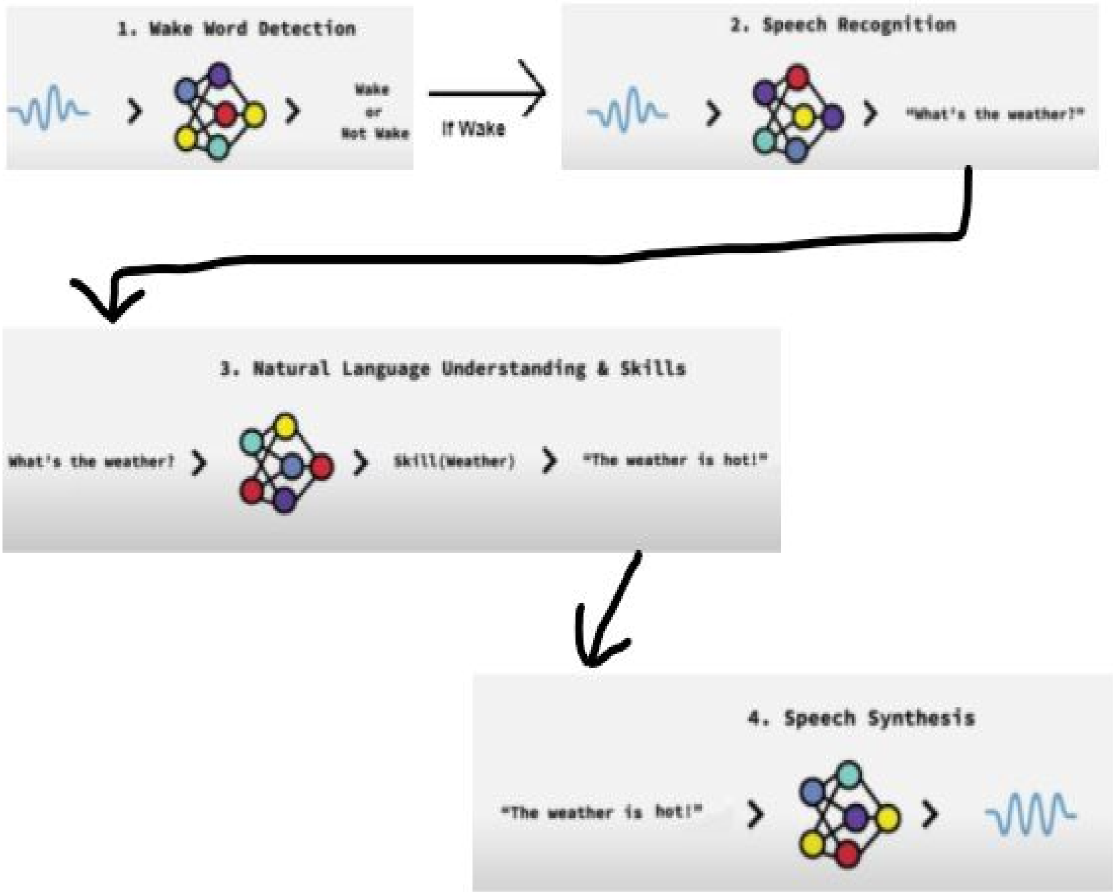


Fig 1.2 Block Diagram showing how the components in our modules will work

All of this is followed by a chapter showing its implementation and final working followed by a chapter which would include the conclusion about the entire project, future scope, applications and advantages and disadvantages of our application.

CHAPTER 2 WAKEWORD

2.1 Introduction

A wake word is a word or phrase that when spoken is meant to activate a device. It is also referred to as 'trigger word', or 'wake up word'. There are a few famous wake phrases such as "Alexa" "Hey Siri", and "OK Google" . The need for wake word is because we can’t have the whole AI engine running at all times as it will compute a lot of resources so we have a small model running in background which can wake up the rest of our AI engine this makes it resource efficient For our project we have used “happy” as our wake word which when said will wake up the AI engine.

2.2 Data Analysis and Preprocessing

For wakeword we used Google speech commands dataset this dataset contains 1,10,000 one-second long utterances of 35 short words. For exploration and processing we used librosa and torchaudio libraries.
The sounds had the sampling rate of 16,000 samples/sec with varying length of about one second so before using it we truncated or padded to make it of length 1second each.

SpecAugment is an augmentation technique that modifies the spectrogram by warping it in the time direction, by masking certain frequencies, cutting certain parts of the spectrogram or changing the pitch at certain points. These augmentations helps the model to train better as it has to learn from partial or deformed data. It is especially helpful if your dataset is small.

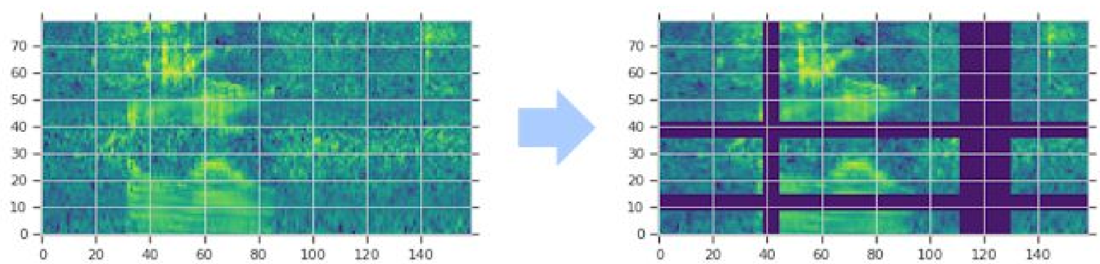


Fig 2.1 SpecAugment applied to a spectogram

We applied SpecAugmentation to our training dataset so that our model can be more general and is able to learn better.

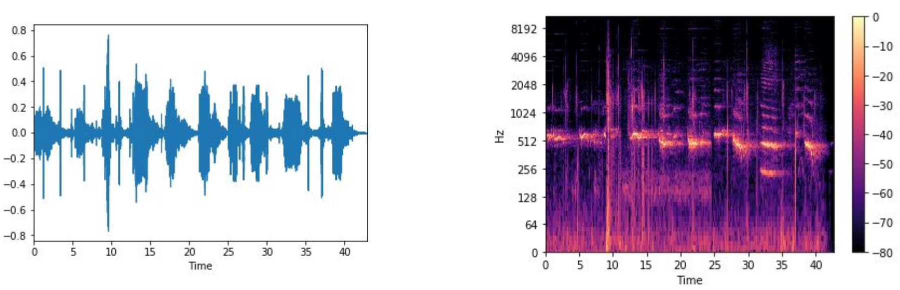
```
class SpecAugment(nn.Module):

    def __init__(self, rate=0.5, policy=3, freq_mask=15, time_mask=35):
        super(SpecAugment, self).__init__()

        self.rate = rate

        self.specaug = nn.Sequential(
            torchaudio.transforms.FrequencyMasking(freq_mask_param=freq_mask),
            torchaudio.transforms.TimeMasking(time_mask_param=time_mask)
        )

        self.specaug2 = nn.Sequential(
            torchaudio.transforms.FrequencyMasking(freq_mask_param=freq_mask),
            torchaudio.transforms.TimeMasking(time_mask_param=time_mask),
            torchaudio.transforms.FrequencyMasking(freq_mask_param=freq_mask),
            torchaudio.transforms.TimeMasking(time_mask_param=time_mask)
        )
```

Fig 2.2 Code for Class SpecAugment

2.3 Feature Extraction

Mel-Spectrogram: The MelSpectrogram is a spectrogram in which the y-axis is the Mel scale and the x-axis is time

For our wakeword model after preprocessing the audio wave they are converted into Mel Spectrogram. We have used 128 Mel filter bank to extract features because it helps to simulate the way human ears works. It corresponds to better resolution at low frequencies and less at high. From these Mel Spectrogram we will be extracting features using deep learning



Initial audio signal Corresponding Mel Spectrogram

Fig 2.3 Initial audio signal and its Corresponding Mel Spectrogram

```python
class LogMelSpec(nn.Module):

    def __init__(self):
        super(LogMelSpec, self).__init__()
        self.transform = torchaudio.transforms.MelSpectrogram(n_mels=128)

    def forward(self, x):
        x = self.transform(x)  # mel spectrogram
        x = np.log(x + 1e-14)
        return x
```

Fig 2.4 Code for Class LogMelspec

2.4 Training

Once we have got a mel spectrogram we will be using Convolutional Neural Network (ConvNet/CNN) and dense layers to make a deep learning model that can be trained on this.

We have used CNN because it is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters.

The various layers present in a CNN are :

Convolutional Layer: The method of convolution of a 2d data with a filter is the summation of all the values in the dot product of the filter with the data. Applying the same filter throughout the whole image results in a feature map.

Pooling Layer: This layer is present to reduce the size of the feature map so that the computational power required to process the data decreases. We have used Max Pooling which takes the maximum of the values of the portion of the data on which the filter is applied.

Dense Layer: this is just a regular layer consisting of neurons. Every neuron in this layer receives the input from each neuron in the previous layer thus is called fully connected or dense

We have used three convolution layers with kernel size of (3X3)

We have kept 30% dropout and have also used batch normalisation to avoid over fitting.

We have used 3 fully connected layers

We have used CrossEntropyLoss() as our loss function and AdamW as our optimiser with a learning rate of 0.005

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(net.parameters(),lr=0.0005)
scheduler = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr=0.0005,
                                                steps_per_epoch=int(len(trainloader)),
                                                epochs=25,
                                                anneal_strategy='linear')
```

. Fig 2.5 Code showing loss function, optimizer and scheduler

Model Architecture:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 8, 126, 79]              80
           Dropout-2            [-1, 8, 42, 26]               0
            Conv2d-3           [-1, 32, 40, 24]           2,336
           Dropout-4            [-1, 32, 13, 8]               0
            Conv2d-5            [-1, 64, 11, 6]          18,496
           Dropout-6            [-1, 64, 11, 6]               0
            Linear-7                  [-1, 256]       1,081,600
       BatchNorm1d-8                  [-1, 256]             512
           Dropout-9                  [-1, 256]               0
           Linear-10                  [-1, 128]          32,896
      BatchNorm1d-11                  [-1, 128]             256
          Dropout-12                  [-1, 128]               0
           Linear-13                   [-1, 35]           4,515
================================================================
Total params: 1,140,691
Trainable params: 1,140,691
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.04
Forward/backward pass size (MB): 1.01
Params size (MB): 4.35
Estimated Total Size (MB): 5.40
----------------------------------------------------------------
```

Fig 2.6 Various layers of model and number of parameters in it

```
NN2DMEL(
  (conv1): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1))
  (dropout1): Dropout(p=0.3, inplace=False)
  (conv2): Conv2d(8, 32, kernel_size=(3, 3), stride=(1, 1))
  (dropout2): Dropout(p=0.3, inplace=False)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=4224, out_features=256, bias=True)
  (batch1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout5): Dropout(p=0.3, inplace=False)
  (fc2): Linear(in_features=256, out_features=128, bias=True)
  (batch2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout6): Dropout(p=0.3, inplace=False)
  (fc3): Linear(in_features=128, out_features=35, bias=True)
)
```

Fig 2.7 Model Architecture

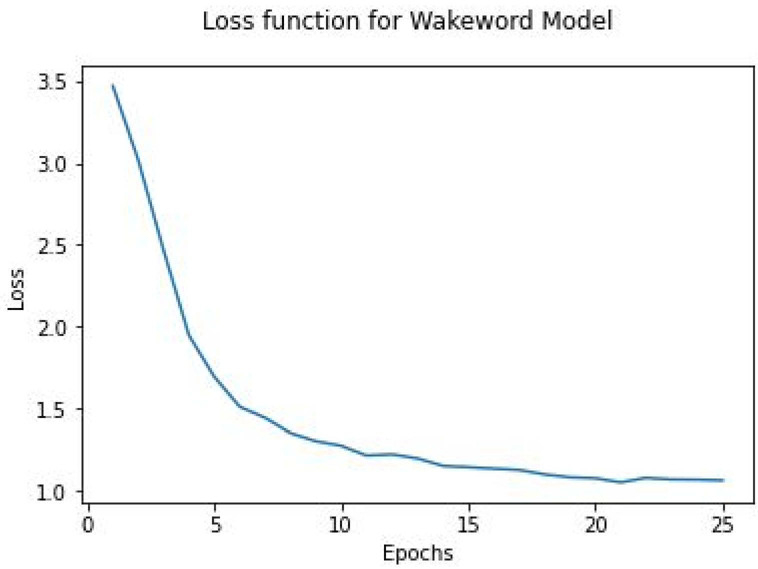


Fig 2.8 Loss Function for Wakeword Model

2.5 Results

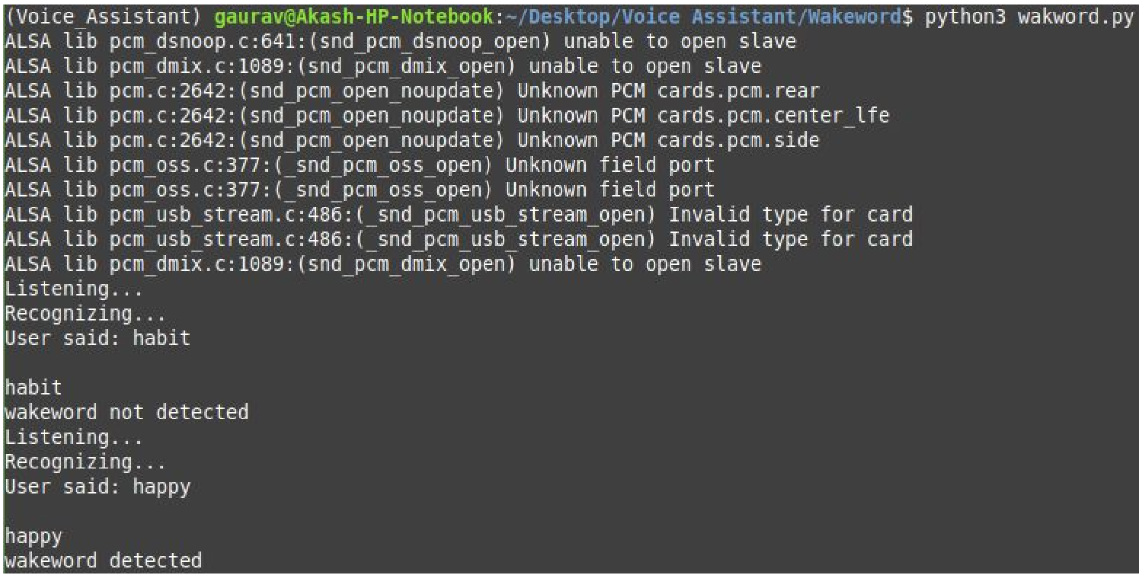
Result of testing on validation dataset

Validation Epoch	Loss and Accuracy
Validation Epoch #1	Loss: 3.2800 Acc@1: 11.79%
Validation Epoch #2	Loss: 2.6969 Acc@1: 25.76%
Validation Epoch #3	Loss: 2.0980 Acc@1: 42.34%
Validation Epoch #4	Loss: 1.6870 Acc@1: 53.46%
Validation Epoch #5	Loss: 1.4906 Acc@1: 59.01%
Validation Epoch #6	Loss: 1.3383 Acc@1: 63.43%
Validation Epoch #7	Loss: 1.2732 Acc@1: 65.56%
Validation Epoch #8	Loss: 1.1634 Acc@1: 67.45%
Validation Epoch #9	Loss: 1.1470 Acc@1: 68.11%
Validation Epoch #10	Loss: 1.0759 Acc@1: 69.95%
Validation Epoch #11	Loss: 1.0430 Acc@1: 70.85%
Validation Epoch #12	Loss: 1.0150 Acc@1: 71.47%
Validation Epoch #13	Loss: 0.9920 Acc@1: 72.46%
Validation Epoch #14	Loss: 0.9659 Acc@1: 72.95%
Validation Epoch #15	Loss: 0.9493 Acc@1: 73.21%
Validation Epoch #16	Loss: 0.9632 Acc@1: 72.74%
Validation Epoch #17	Loss: 0.9258 Acc@1: 73.97%
Validation Epoch #18	Loss: 0.9238 Acc@1: 73.49%
Validation Epoch #19	Loss: 0.8984 Acc@1: 74.74%
Validation Epoch #20	Loss: 0.8961 Acc@1: 74.53%
Validation Epoch #21	Loss: 0.9188 Acc@1: 73.60%
Validation Epoch #22	Loss: 0.8814 Acc@1: 74.93%
Validation Epoch #23	Loss: 0.8661 Acc@1: 75.10%
Validation Epoch #24	Loss: 0.8758 Acc@1: 75.17%
Validation Epoch #25	Loss: 0.8678 Acc@1: 75.09%

Table 2.1 Testing loss and Accuracy

After training for 25 epochs our accuracy was 75.09%.

Screenshot below show its working



```
(Voice_Assistant) gaurav@Akash-HP-Notebook:~/Desktop/Voice Assistant/Wakeword$ python3 wakword.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
Listening...
Recognizing...
User said: habit

habit
wakeword not detected
Listening...
Recognizing...
User said: happy

happy
wakeword detected
```

Fig 2.9 Working of wakeword model

CHAPTER 3 SPEECH TO TEXT

3.1 Introduction

Speech to text or automatic speech recognition (ASR), is the process of converting spoken words into written text it is also known as speech recognition. In our project we have used DeepSpeech a deep learning based speech recognition model that will convert speech into text for us. The reason we went for a pre trained model is because training a model with 100 million parameters on 1000 hour speech would not have been possible for us as we don’t have that much resource and it aligns with our goal that is it is privacy friendly and accurate . Many year's worth of work and research has brought DeepSpeech, which we have utilized. Here we have utilized the second era of this framework that epitomizes the critical endpoints of interest of to-end learning. This Deep Speech pipeline sent here methodologies or surpasses Amazon Mechanical Turk human works' exactness on a few benchmarks, and even in different dialects with simply some slight alteration. Since the framework is based on profound start to finish learning, it can utilize an expansive range of profound learning strategies: catching enormous preparing sets, preparing bigger models with superior registering, and efficiently investigating the space of neural organization designs. We can decrease the mistake paces of our start to finish framework by up to 40% contrasted with conventional frameworks through this strategy.

Since Deep Speech is a start to finish a profound learning framework, it can accomplish execution gains by zeroing in on essentially three segments that are model engineering, enormous marked preparing datasets, and computational scale. Specifically, we portray various tests with neural organizations prepared with the Connectionist Temporal Classification (CTC) misfortune capacity to foresee discourse records from the sound.

3.2 Data Preprocessing

The model expects the audio file that is being used as input should be sampled at 16,000Hz but depending upon the type of microphone being used and the operating system settings, audio could be sampled at any frequency, so to handle this, we need to resample the frequency at 16,000Hz and convert it to NumPy int so the model can understand it and work with it.

```
def predicts_audio(audio):
    w = wave.open(audio, 'r')
    sound = am.from_file(audio, format='wav', frame_rate=w.getframerate())
    sound = sound.set_frame_rate(16000)
    sound.export('1', format='wav')
    w = wave.open('1', 'r')
    frames = w.getnframes()
    buffer = w.readframes(frames)
    data16 = np.frombuffer(buffer, dtype=np.int16)
```

Fig 3.1 Code for Data Preprocessing

This data16 when feed into the model will give text as output

3.3 Model Architecture

A basic multi-layer model with a solitary intermittent layer can't abuse a very long time of named discourse. To gain from datasets this huge, we have expanded the model limit by means of profundity. We have executed the models with up to 11 layers.
These models contain around multiple times the measure of calculation per information model contrasted with the normal standard Voice Recognition models and thus making it a quick streamlining and calculation basic framework. To advance these models effectively, we have utilized Batch Normalization for RNNs and a novel improvement educational plan. At last, however many examination results utilize bidirectional intermittent layers, we locate that fantastic models exist utilizing just unidirectional repetitive layers—an element that makes such models a lot simpler to convey. These highlights permitted us to workably streamline profound RNNs and improve execution by over 40% blunder rates over the more modest pattern models.

The Complete Speech to Text have been divided into three programs namely predict.py , recordaudio.py , speechtotext.py. recordaudio.py file is responsible for input of voice commands from the user using deepspeech and wave file .

Fig 3.2 Code to record audio

Next comes speechtottext.py file which converts the captured audio into text

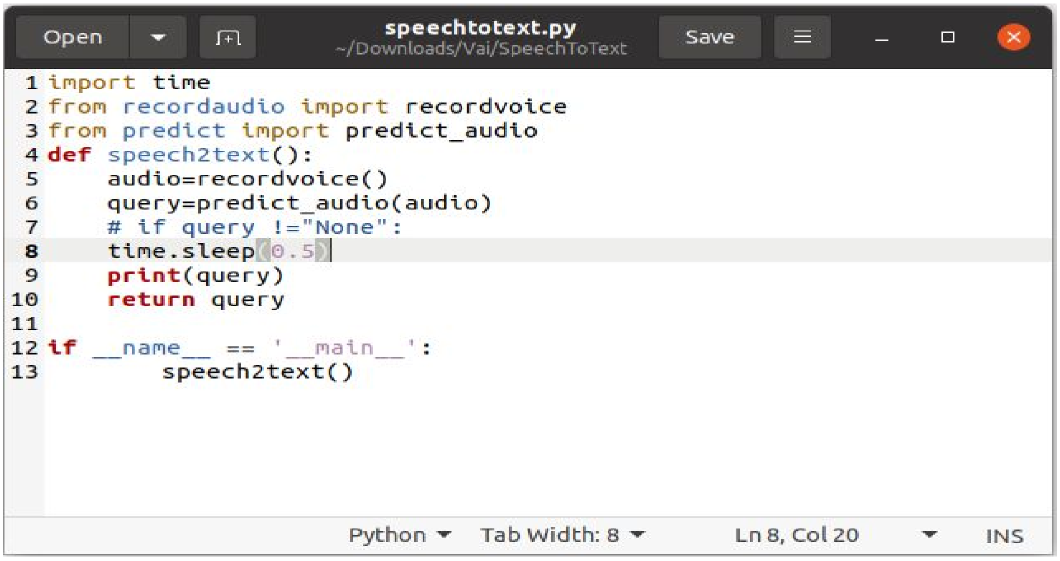


Fig 3.3 Code of main speech function

And the Final and the most important is to convert the text to an meaning full sentence which is done by predict.py

```python
def predicts_audio(audio):
    w = wave.open(audio, 'r')
    sound = am.from_file(audio, format='wav', frame_rate=w.getframerate())
    sound = sound.set_frame_rate(16000)
    sound.export('1', format='wav')
    w = wave.open('1', 'r')
    frames = w.getnframes()
    buffer = w.readframes(frames)
    data16 = np.frombuffer(buffer, dtype=np.int16)
```

Fig 3.4 Code that shows audio prediction

3.4 Training

Preparing enormous amounts of information generally requires the utilization of bigger models. Preparing a solitary model at these scales requires several exaFLOPs1 that would require three a month and a half to execute on a solitary GPU. As opposed to past huge scope preparing approaches that utilization boundary workers and nonconcurrent refreshes we utilize coordinated SGD, which is simpler to investigate while testing novel thoughts, and furthermore joins quicker for a similar level of information parallelism. To make the whole framework effective, we depict enhancements for a solitary GPU just as upgrades to adaptability for various GPUs. We utilize streamlining procedures which improved adaptability. These improvements incorporate a quick execution of the CTC misfortune work on the GPU and a custom memory allocator.

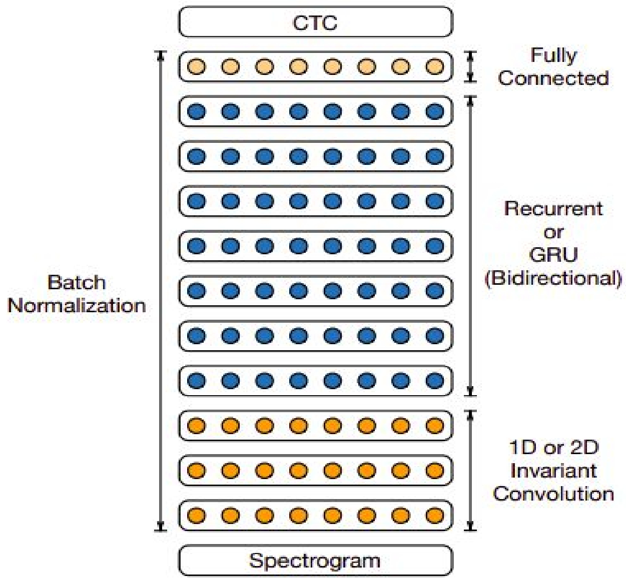


Fig 3.5. Architecture of the system Used to Train the system

The framework has been benchmarked on a few openly accessible test sets. Our objective was to, in the long run, arrive at human-level execution not just on explicit benchmarks. Going forward, we have similarly estimated the presentation of human specialists on every benchmark for correlation.

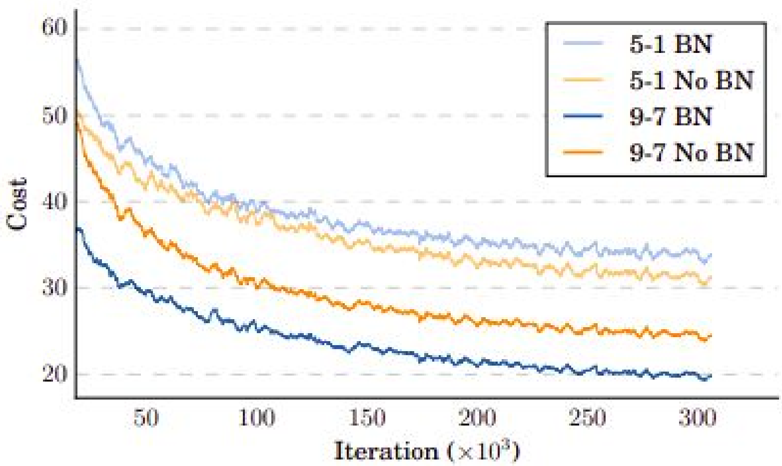


Fig 3.6 Training curves of two models trained with and without BatchNorm

3.5 Results

To all the more likely surveys this present reality relevance of our discourse framework, we assess a wide scope of test sets. We have utilized different freely accessible benchmarks, and a few test sets gathered inside. Together these test sets speak to a wide scope of testing discourse conditions, including low sign-to-commotion proportions (loud and far-field), complemented, read, unconstrained and conversational discourse.

Fraction of Data	Hours	Regular Dev	Noisy Dev
1%	120	29.23	50.97
10%	1200	13.80	22.99
20%	2400	11.65	20.41
50%	6000	9.51	15.90
100%	12000	8.46	13.59

Fig 3.7 Comparison of Regular and Noisy development sets on increasing training dataset size.

Consequently, this End-to-end profound learning presents the energizing occasion to improve discourse acknowledgment frameworks persistently with increments in information and calculation. To be sure, our outcomes show that, contrasted with the past manifestation.

To accomplish these outcomes, we have investigated different organization structures, finding a few powerful procedures: improvements to mathematical streamlining through SortaGrad and Batch Normalization, assessment of RNNs with bigger steps with bigram yields, looking through both bidirectional and unidirectional models.

Final working:-

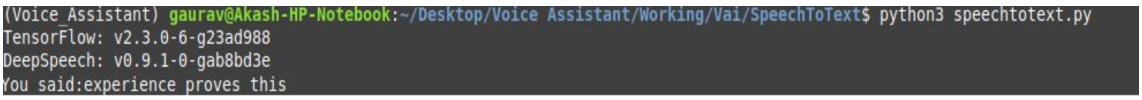


Fig 3.8 Final output of speech to text

CHAPTER 4 NLU

4.1 Introduction

Natural language understanding (NLU) comes under Artificial Intelligence (AI), which involves breaking down the human language into a machine understandable format. NLU tries to understand grammatical rules and syntax of the language so that it can make sense of the data. NLU makes it possible for machines to understand the overall context and meaning of "natural language," beyond literal definitions. Its goal is to understand the language the same way a human would. In our project we have used NLU to classify the intent of our speech in various categories so we can tell the machine whether it needs to open a file, search for something or do some calculations.
To help model provide more of an individualized experience to the user's we have provided the user with an option to enter his google credentials and then we will fetch his history using selenium and the model will train on it automatically from time to time.

4.2 Data Preprocessing and Feature Extraction

Our dataset is in json format containing three labels tag, intent and its corresponding response so our first step is to make the string in lowercase and remove punctuations and stopwords from it.

Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords. We have done word tokenization converting user's query into tokens using nltk module.

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. After we have done tokenization every token is converted into its root word for this we have used PorterStemmer.

Bag_of_words: A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. We have used our training data to make a bag of words.

4.3 Training

To train our model we have used a Deep Learning model which uses LSTM and fully connected layers. We are not using RNN as they suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones.

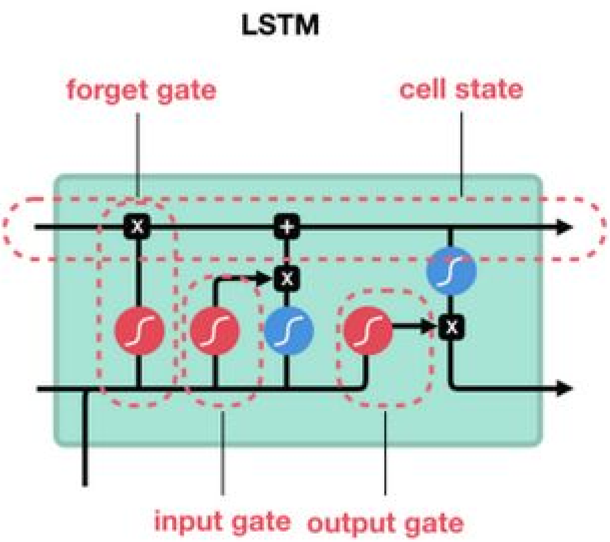


Fig 4.1 Block Diagram of LSTM

For our lstm we will be keeping embedding dimension to 20, number of layers will be equal to 2, is biredictonal in nature and has a dropout percentage of 20

```
class LSTMClassifier(nn.Module):

    def __init__(self, embedding_dim, hidden_dim, vocab_size,out_dim):

        super(LSTMClassifier, self).__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=0)
        self.lstm = nn.LSTM(embedding_dim, hidden_dim,batch_first=True)
        self.dense = nn.Linear(in_features=hidden_dim, out_features=out_dim)
        self.sig = nn.Sigmoid()

        self.word_dict = None

    def forward(self, x):
        embeds = self.embedding(x)
        lstm_out, (ht,ct) = self.lstm(embeds)
        out = self.dense(ht[-1])
        return out
```

Fig 4.2 Code showing NLU model declaration

Model was trained for 25 epochs
Batch size was set to 8
Learning rate was set to 0.001
Loss Function we used was BinaryCrossEntropy
AdamW was used as our optimiser to optimize our neural network during training.

```
Epoch [1/25], Loss: 1.9534
Epoch [2/25], Loss: 1.5045
Epoch [3/25], Loss: 0.7309
Epoch [4/25], Loss: 0.8868
Epoch [5/25], Loss: 0.5004
Epoch [6/25], Loss: 0.4838
Epoch [7/25], Loss: 0.3771
Epoch [8/25], Loss: 1.8504
Epoch [9/25], Loss: 0.2132
Epoch [10/25], Loss: 0.3368
Epoch [11/25], Loss: 0.2858
Epoch [12/25], Loss: 0.1092
Epoch [13/25], Loss: 0.1184
Epoch [14/25], Loss: 0.0742
Epoch [15/25], Loss: 0.0509
Epoch [16/25], Loss: 0.0375
Epoch [17/25], Loss: 0.0691
Epoch [18/25], Loss: 0.0987
Epoch [19/25], Loss: 0.0463
Epoch [20/25], Loss: 0.0305
Epoch [21/25], Loss: 0.0597
Epoch [22/25], Loss: 0.0497
Epoch [23/25], Loss: 0.0510
Epoch [24/25], Loss: 0.0228
Epoch [25/25], Loss: 0.0226
final loss: 0.0226
```

Fig 4.3 Training Loss of NLU Model

4.4 Results

After training the model for 25 epochs we got final loss equal to 0.0226

Screenshot showing working of NLU model

```
(Voice_Assistant) gaurav@Akash-HP-Notebook:~/Desktop/Voice Assistant/NLU$ python3 Chat.py
You: hello
bot: Hi there, how can I help?
You: How are you doing
bot: I am good Wbu?
You: Tell me about yourself
bot: I am a voice assistant created by Akash, Ankit, Abhishek and Rahul as a BTP project under the guidance of Dr Veenu
You: What is Nsut
bot: Searching
```

Fig 4.4 Output of NLU Model

CHAPTER 5 TEXTTOSPEECH

5.1 Introduction

A text-to-speech (TTS) is that the artificial production of human speech for a given text. A text-to-speech system is created from 2 parts: a front-end and a back-end. The front-end has 2 tasks, beginning is to try to do pre-processing, or tokenization within which we tend to convert raw text containing numbers and abbreviations into their equivalent written-out words. Then every word is assigned phonetic transcriptions to, the method of assignment phonetic transcriptions to words is termed text-to-phoneme or grapheme-to-phoneme conversion. The back-end conjointly said because the synthesizer— it's chargeable for the conversion of the symbolic linguistic illustration into sound.

5.2 Data Analysis and Preprocessing

For speech to text we used Lj Speech Dataset which consists of 13,100 short audio clips transcription is also provided for each clip in metadata.csv. Clips present in the dataset vary from length 1 to 10 seconds and have a total length of approximately 24 hours.
Each audio file present in the dataset is a single-channel 16-bit PCM WAV with a sampled at the rate of 22050 Hz.

Metadata about the dataset is provided in **metadata.csv**. The fields present in the metadata.csv are:

1. **ID**: denoting name of the.wav file
2. **Transcription**: words spoken in the audio file
3. **Normalized Transcription**: This contains transcription with numbers, ordinals, and monetary units expanded into full words (UTF-8).

First we trim our audio to remove silence from the beginning and from ending of our audio file using the librosa library.Since our vocabulary is limited only to letters if there happens to be a number in it we convert that number into its corresponding word so that it can be further processed upon.

5.3 Feature Extraction

Mel-Spectrogram: The MelSpectrogram is a spectrogram in which the y-axis is the Mel scale and the x-axis is timeIn our SpeechToText model after preprocessing the audio wave they are converted into normalized melspectogram and magnitude spectrogram. We have used 80 mel banks filters to extract features from the audio file.

```
def get_spectrograms(fpath):

    # Loading sound file
    y, sr = librosa.load(fpath, sr=hp.sr)

    # Trimming
    y, _ = librosa.effects.trim(y)
    y = np.append(y[0], y[1:] - hp.preemphasis * y[:-1])
    linear = librosa.stft(y=y,
                          n_fft=hp.n_fft,
                          hop_length=hp.hop_length,
                          win_length=hp.win_length)

    # magnitude spectrogram
    mag = np.abs(linear)
    # mel spectrogram
    mel_basis = librosa.filters.mel(hp.sr, hp.n_fft, hp.n_mels)
    mel = np.dot(mel_basis, mag)
    mel = 20 * np.log10(np.maximum(1e-5, mel))
    mag = 20 * np.log10(np.maximum(1e-5, mag))

    # normalize
    mel = np.clip((mel - hp.ref_db + hp.max_db) / hp.max_db, 1e-8, 1)
    mag = np.clip((mag - hp.ref_db + hp.max_db) / hp.max_db, 1e-8, 1)

    # Transpose
    mel = mel.T.astype(np.float32)
    mag = mag.T.astype(np.float32)

    return mel, mag
```

Fig 5.1 Code for generation of Melspectogram and Magnitude Spectrogram

This Mel Spectrogram and Magnitude Spectrogram generated from the audio will be used to train our models.

5.4 Training

We first trained our text2mel model and after that we trained our ssrn model. We trained text2mel for 55 epochs and ssrn for 30 epochs with text2mel having an initial learning rate of 0.005 which we was gradually decreased and ssrn having an initial learning rate of 0.0005 both of them has used Adam optimiser to tune model while training.

For our text2mel we take input as text and mel spectrogram and it also need a vocabulary file for its working, vocab that is given is encoded using a function called text2enc which is made up of convolution layers, We have used sigmoid activation function at the output layer.

```python
class Text2Mel(nn.Module):
    def __init__(self, vocab, d=hp.d):
        super(Text2Mel, self).__init__()
        self.d = d
        self.text_enc = TextEnc(vocab)
        self.audio_enc = AudioEnc()
        self.audio_dec = AudioDec()

    def forward(self, L, S, monotonic_attention=False):
        K, V = self.text_enc(L)
        Q = self.audio_enc(S)
        A = torch.bmm(K.permute(0, 2, 1), Q) / np.sqrt(self.d)

        if monotonic_attention:
            B, N, T = A.size()
            for i in range(B):
                prva = -1  # previous attention
                for t in range(T):
                    _, n = torch.max(A[i, :, t], 0)
                    if not (-1 <= n - prva <= 3):
                        A[i, :, t] = -2 ** 20  # some small numbers
                        A[i, min(N - 1, prva + 1), t] = 1
                    _, prva = torch.max(A[i, :, t], 0)

        A = F.softmax(A, dim=1)
        R = torch.bmm(V, A)
        R_prime = torch.cat((R, Q), 1)
        Y_logit = self.audio_dec(R_prime)
        Y = F.sigmoid(Y_logit)
        return Y_logit, Y, A
```

Fig 5.2 Code of Text2mel model

Text2Mel Model Architecture

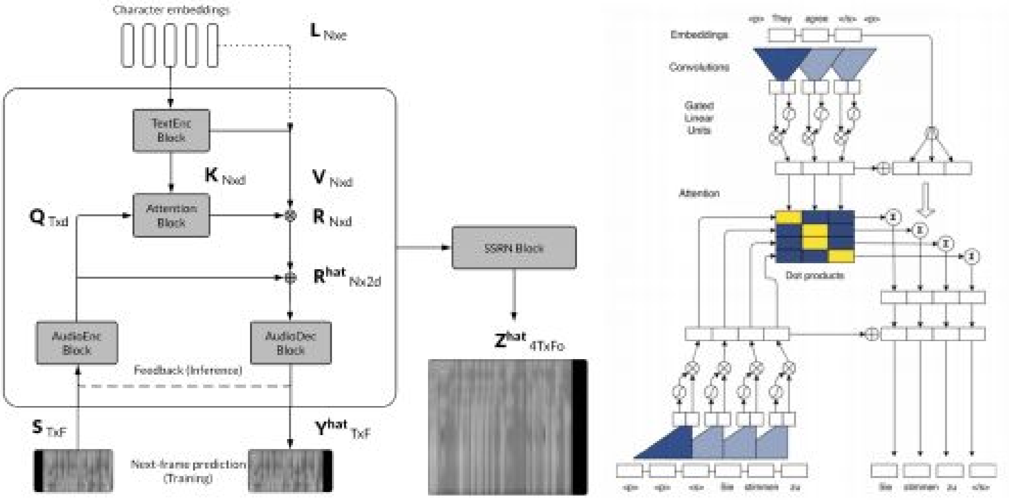


Fig 5.3 Text2Mel Model Architecture

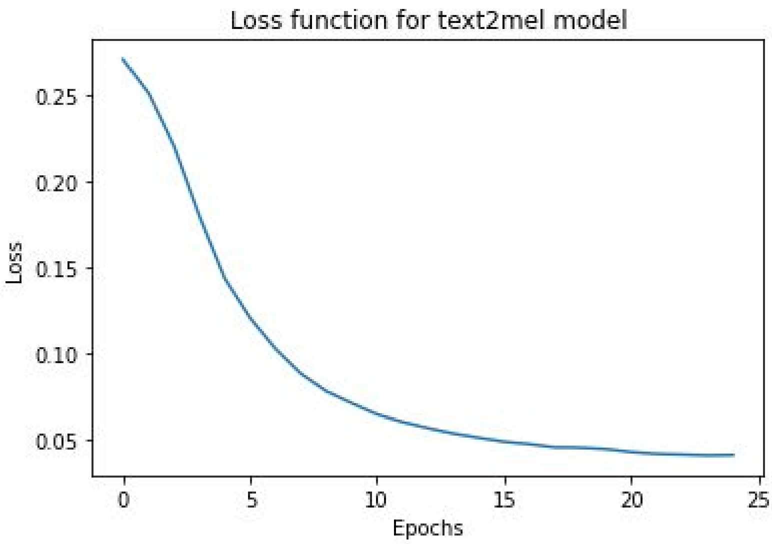


Fig 5.4 Loss function for text2mel model

SSRN input is the output of text2mel model it contains a series of convolution and deconvolution layers; its output is a spectogram which can be converted into an audio file.

We have used sigmoid as an activation function at the output layer and have used Adam optimiser to optimise it while we are training it.

```
class SSRN(nn.Module):
    def __init__(self, c=hp.c, f=hp.n_mels, f_prime=(1 + hp.n_fft // 2)):

        super(SSRN, self).__init__()
        self.layers = nn.Sequential(
            Conv(f, c, 1, 1),

            BasicBlock(c, 3, 1), BasicBlock(c, 3, 3),

            DeConv(c, c, 2, 1), BasicBlock(c, 3, 1), BasicBlock(c, 3, 3),
            DeConv(c, c, 2, 1), BasicBlock(c, 3, 1), BasicBlock(c, 3, 3),

            Conv(c, 2 * c, 1, 1),

            BasicBlock(2 * c, 3, 1), BasicBlock(2 * c, 3, 1),

            Conv(2 * c, f_prime, 1, 1),

            # Conv(f_prime, f_prime, 1, 1, nonlinearity='relu'),
            # Conv(f_prime, f_prime, 1, 1, nonlinearity='relu'),
            BasicBlock(f_prime, 1, 1),

            Conv(f_prime, f_prime, 1, 1)
        )

    def forward(self, x):
        Z_logit = self.layers(x)
        Z = F.sigmoid(Z_logit)
        return Z_logit, Z
```

Fig 5.5 Code of ssrn model

SSRN Model Architecture

```
SSRN(
  (layers): Sequential(
    (0): C(
      (conv): Conv1d(80, 640, kernel_size=(1,), stride=(1,))
    )
    (1): ResidualBlock(
      (C1): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
      )
      (C2): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
      )
    )
    (2): ResidualBlock(
      (C1): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
      )
      (C2): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
      )
    )
    (3): D(
      (deconv): ConvTranspose1d(640, 640, kernel_size=(2,), stride=(2,))
    )
    (4): ResidualBlock(
      (C1): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
      )
      (C2): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
      )
    )
    (5): ResidualBlock(
      (C1): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
      )
      (C2): C(
        (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
      )
    )
```

```
(6): D(
  (deconv): ConvTranspose1d(640, 640, kernel_size=(2,), stride=(2,))
)
(7): ResidualBlock(
  (C1): C(
    (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
  )
  (C2): C(
    (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(1,))
  )
)
(8): ResidualBlock(
  (C1): C(
    (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
  )
  (C2): C(
    (conv): Conv1d(640, 640, kernel_size=(3,), stride=(1,), padding=(3,), dilation=(3,))
  )
)
(9): C(
  (conv): Conv1d(640, 1280, kernel_size=(1,), stride=(1,))
)
(10): ResidualBlock(
  (C1): C(
    (conv): Conv1d(1280, 1280, kernel_size=(3,), stride=(1,), padding=(1,))
  )
  (C2): C(
    (conv): Conv1d(1280, 1280, kernel_size=(3,), stride=(1,), padding=(1,))
  )
)
(11): ResidualBlock(
  (C1): C(
    (conv): Conv1d(1280, 1280, kernel_size=(3,), stride=(1,), padding=(1,))
  )
  (C2): C(
    (conv): Conv1d(1280, 1280, kernel_size=(3,), stride=(1,), padding=(1,))
  )
)
(12): C(
  (conv): Conv1d(1280, 1025, kernel_size=(1,), stride=(1,))
)
(13): ResidualBlock(
  (C1): C(
    (conv): Conv1d(1025, 1025, kernel_size=(1,), stride=(1,))
```

```
    )
    (C2): C(
      (conv): Conv1d(1025, 1025, kernel_size=(1,), stride=(1,))
    )
  )
  (14): C(
    (conv): Conv1d(1025, 1025, kernel_size=(1,), stride=(1,))
  )
 )
)
```

```
================================================================
Total params: 41,399,175
Trainable params: 41,399,175
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 62.66
Params size (MB): 157.93
Estimated Total Size (MB): 220.61
----------------------------------------------------------------
```

Fig 5.6 Parameters of ssrn model

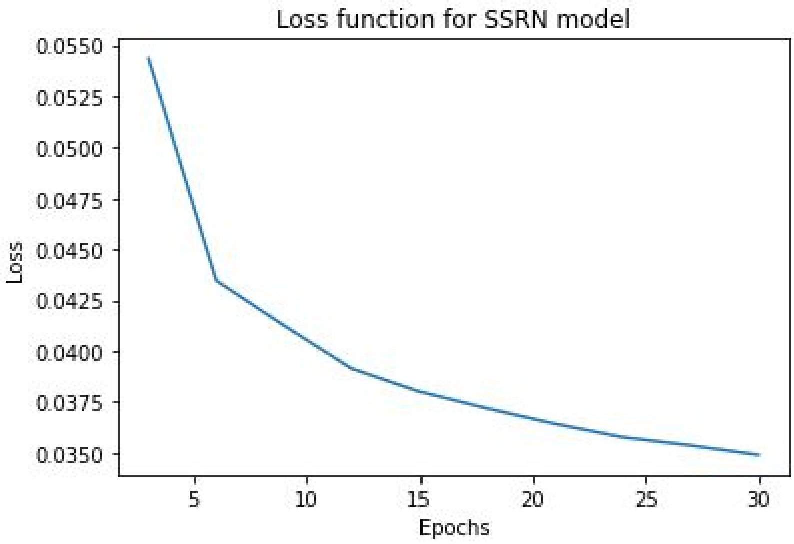


Fig 5.7 Loss function for ssrn model

5.5 Results

Result obtained from tensorboard for Text2mel showing its training and validation loss

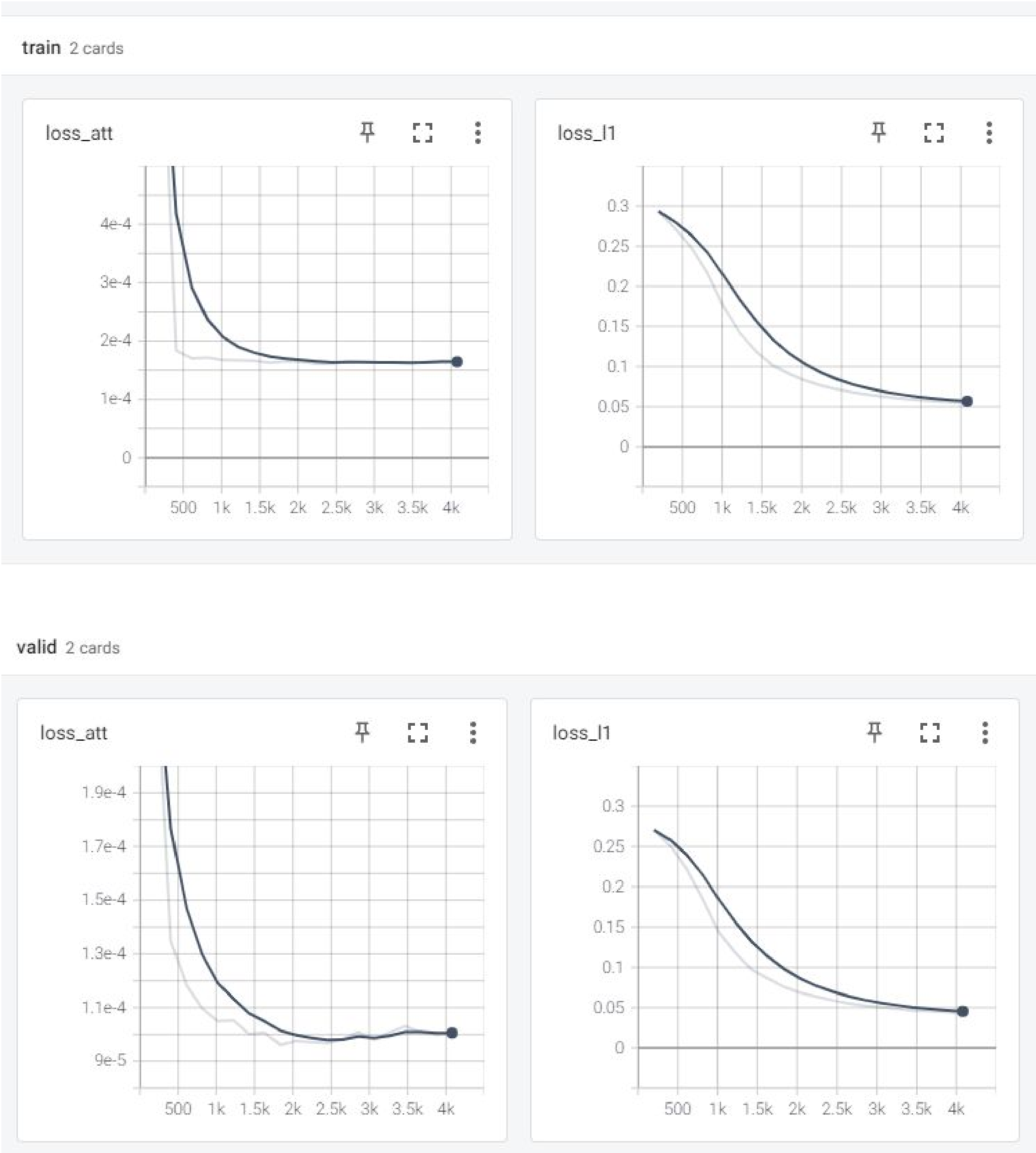
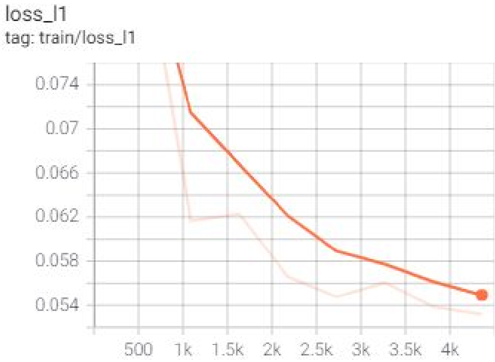


Fig 5.8 Result obtained from tensorboard for Text2mel showing its training and validation loss



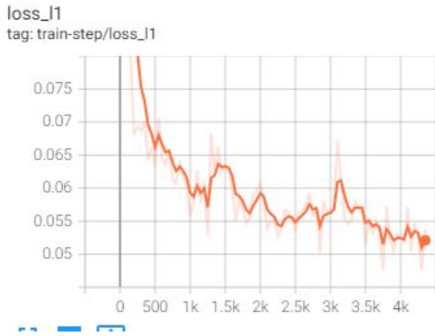


Fig 5.9 Result obtained from tensorboard for ssrn showing its training and validation loss

The function below takes a sentence as an input then it is normalized and it passes through our text2mel and ssrn and then the spectrogram obtained is converted into an audio file and is played through the speaker.

```
def say(sentence):
    new_sentence=" " .join([num2words(w) if w.isdigit()  else w for w in sentence.split()])
    normalized_sentence = "".join([c if c.lower() in vocab else '' for c in new_sentence])
    print(normalized_sentence)
    sentences = [normalized_sentence]
    max_N = len(normalized_sentence)
    L = torch.from_numpy(get_test_data(sentences, max_N))
    zeros = torch.from_numpy(np.zeros((1, hp.n_mels, 1), np.float32))
    Y = zeros
    A = None

    for t in range(hp.max_T):
      _, Y_t, A = text2mel(L, Y, monotonic_attention=True)
      Y = torch.cat((zeros, Y_t), -1)
      _, attention = torch.max(A[0, :, -1], 0)
      attention = attention.item()
      if L[0, attention] == vocab.index('E'):  # EOS
          break

    _, Z = ssrn(Y)
    i=int(0)
    Z = Z.cpu().detach().numpy()
    save_to_wav(Z[0, :, :].T, '%d.wav' % (i + 1))
    playsound('1.wav')
```

Fig 5.10 Code for generation of speech from text

CHAPTER 6 IMPLEMENTATION

6.1 Modules and Packages

- Python>3.6
- appdirs==1.4.4
- audioread==2.1.9
- beautifulsoup4==4.9.3
- certifi==2020.12.5
- cffi==1.14.4
- chardet==4.0.0
- click==7.1.2
- cycler==0.10.0
- decorator==4.4.2
- deepspeech==0.9.1
- docopt==0.6.2
- Flask==1.1.2
- Flask-SQLAlchemy==2.4.4
- future==0.18.2
- idna==2.10
- imageio==2.9.0
- inflect==5.0.2
- itsdangerous==1.1.0
- jaraco.itertools==5.0.0
- jellyfish==0.8.2
- Jinja2==2.11.2
- joblib==1.0.0
- kiwisolver==1.3.1
- librosa==0.8.0
- llvmlite==0.34.0
- MarkupSafe==1.1.1
- matplotlib==3.3.3
- more-itertools==8.6.0
- networkx==2.5
- nltk==3.5
- num2words==0.5.10
- numba==0.51.2
- numpy==1.19.4
- packaging==20.8
- Pillow==8.0.1
- playsound==1.2.2

- pooch==1.3.0
- protobuf==3.14.0
- PyAudio==0.2.11
- pycparser==2.20
- pydub==0.24.1
- pyparsing==2.4.7
- python-dateutil==2.8.1
- PyWavelets==1.1.1
- regex==2020.9.27
- requests==2.25.1
- resampy==0.2.2
- scikit-image==0.17.2
- scikit-learn==0.23.2
- scipy==1.5.4
- selenium==3.141.0
- six==1.15.0
- SoundFile==0.10.3.post1
- soupsieve==2.1
- SpeechRecognition==3.8.1
- SQLAlchemy==1.3.22
- tensorboardX==2.1
- threadpoolctl==2.1.0
- tifffile==2020.12.8
- torch==1.7.1
- torchaudio==0.7.2
- torchvision==0.8.2
- tqdm==4.55.0
- typing-extensions==3.7.4.3
- urllib3==1.26.2
- Werkzeug==1.0.1
- wikipedia==1.4.0

6.2 Front end Implementation

To make our Voice Assistant easier to use we have made a web app using Flask which is responsible for handling backend its job is to integrate machine learning models into our web pages. So by using flask we have made a local server so in the future if we want to use our Voice Assistant on Smartphone Devices it can be an api call to our local server and fetch information.

To give our website a dynamic look we have used javascript, css and html.

Our website has a dynamic navbar using which users can easily navigate through our web app.

Flask

We have used flask to render our html documents of the user interface. Flask is a web framework written in python. This enabled us to write our ML models using python and link them to our UI components.

Javascript

Javascript is a high-level programming language. It allows us to handle events that occur in our DOM. We have used javascript in our UI to execute a certain piece of code when a button is clicked or a request is made to submit a html form.

CSS

CSS is a style sheet language used to describe the presentation of a document written in markup language such as html. We have used CSS in our UI to style the html elements and to make the navigation bar dynamic. To make the navigation bar dynamic we have used css pseudo classes such as hover, focus, active etc.

HTML

HTML is the standard markup language for documents designed to be displayed in a web browser. Web browsers receive html documents from the web server, in our case the web server is a local server which will be serving the web browser the html documents.

Code snippets

```html
    <body>
        <div class="header">
            <img id="logo-img" src="{{url_for('static',filename='logo.png')}}" alt="">
            <div class="heading"><h1>AI Voice Assistant</h1></div>
        </div>
        <div class="main">
            <div class="nav-bar">
                <ul class="nav-list">
                    <li id="nav-home"><img class="nav-img" id="home" src="{{url_for('static',filename
                    <li id="nav-history"><img class="nav-img" id="history" src="{{url_for('static',fi
                    <li id="nav-details"><img class="nav-img" id="details" src="{{url_for('static',fi
                    <li id="nav-link"><img class="nav-img" id="link" src="{{url_for('static',filename
                </ul>
            </div>
            <div class="content">
                <div class="content-body">
                    <form method="post", action="\">
                        <input id="start-bttn" type="submit" value="Start" />
                    </form>
                    <div class="response">

                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

```html
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Voice Recognition</title>
        <style>
            *{
                box-sizing: border-box;
                padding:0;
                margin:0;
            }

            body{
                display: flex;
                flex-direction: column;
            }

            .header{
                width:100%;
                height:65px;
                padding: 5px 5px;
                background-color: #ddc1d3;
                display: flex;
            }

            .heading{
                display: flex;
                align-items: center;
            }

            #logo-img{
                height:100%;
            }
```

```
        #details{
            width:75%;
            height:75%;
        }

        #link{
            width:75%;
            height:75%;
        }

        .content-body{
            width:100%;
            display: flex;
            flex-direction: column;
            align-items: center;
            padding:35px;
        }

        #start-bttn{
            width:160px;
            height:160px;
            border:3px solid #EC9ED3;
            border-radius: 50%;
            font-size: 1.4em;
            color: #ffffff;
            background-color: #4F3447;
        }

        #start-bttn:hover{
            background-color: #7A516D;
        }
```

```

        .response{
            color: #ffffff;
            font-size: 1.5em;
            width:100%;
            min-height:100px;
            display:flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            border:2px solid #ffffff;
            margin:50px;
            padding:25px;
        }
    </style>
    <script type="text/javascript">
        function myFunction() {
            location.replace("/nlu")
        }
    </script>

</head>
<body>
    <div class="header">
        <img id="logo-img" src="{{url_for('static',filename='logo.png')}}" alt="">
        <div class="heading"><h1>AI Voice Assistant</h1></div>
    </div>
    <div class="main">
        <div class="nav-bar">
            <ul class="nav-list">
                <li id="nav-home"><img class="nav-img" id="home" src="{{url_for('static',filename
```

6.3 Working

Outputs of Working Project

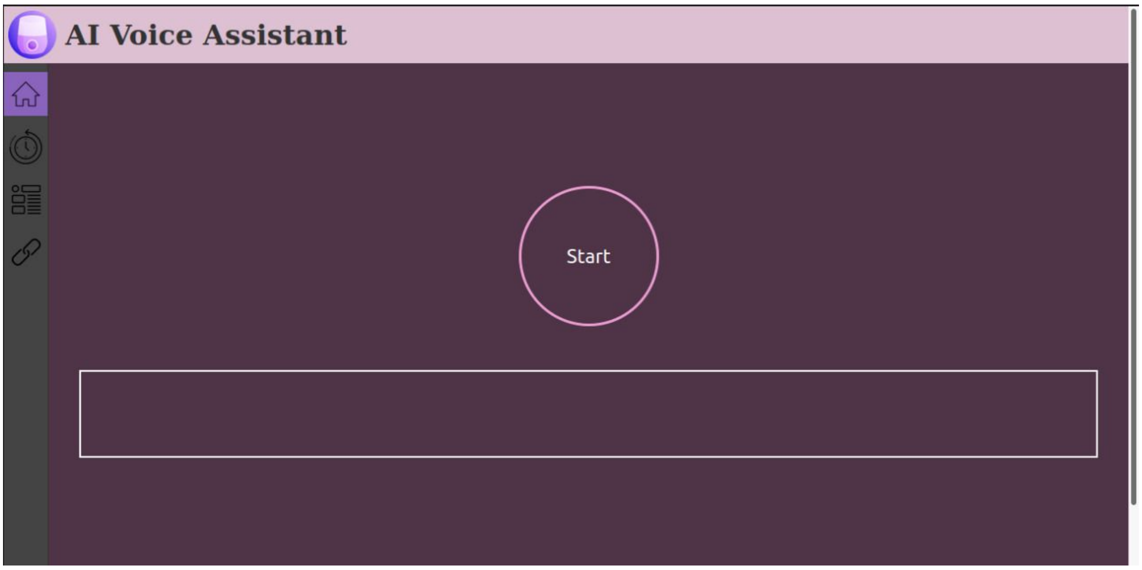


Fig 6.1 Homepage of the AI Voice Assistant Webapp

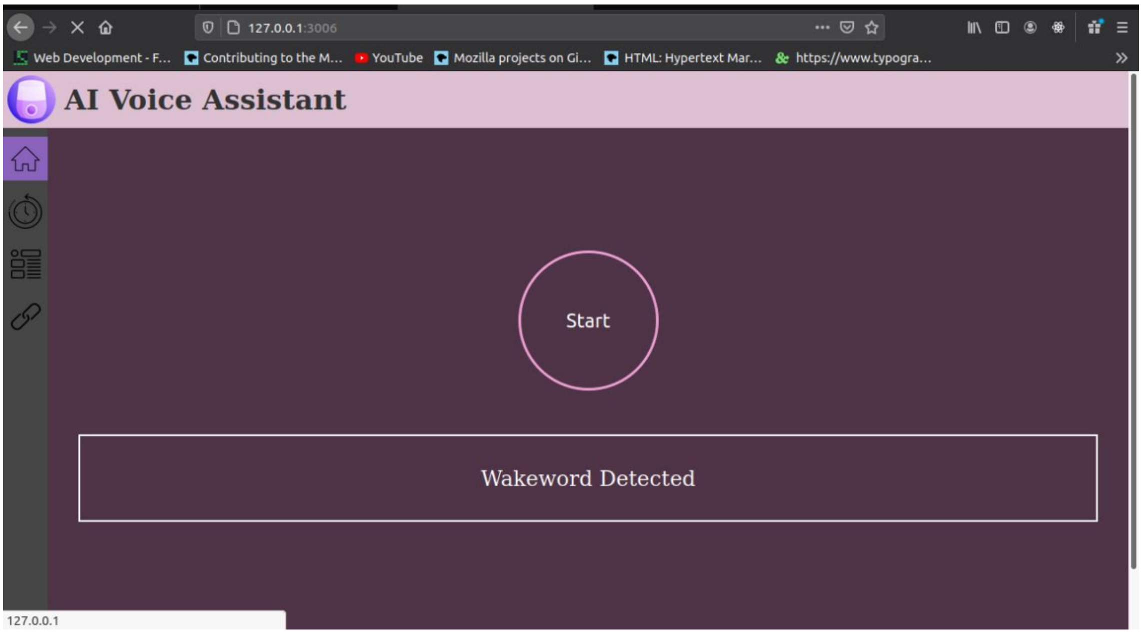


Fig 6.2 Wakeword Detected

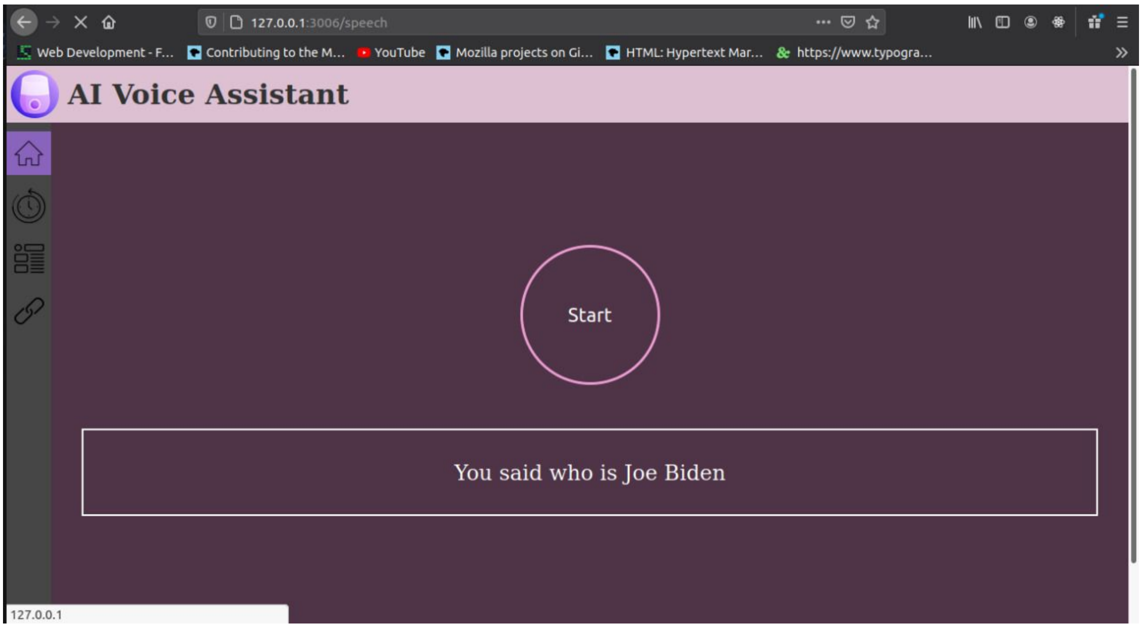


Fig 6.3 Detecting user query

Fig 6.4 Showing results

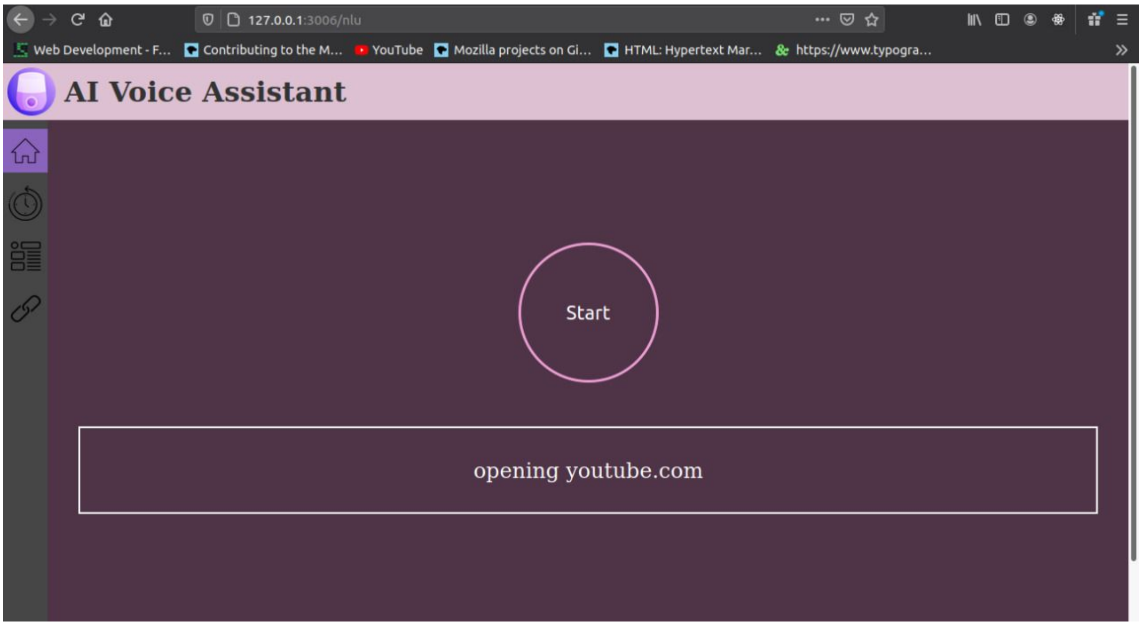


Fig 6.5 Performing action as told

CHAPTER 7 : CONCLUSION

7.1 Advantages / Disadvantages

Advantages:

- No need to worry about privacy or data being stolen from cloud
- Greater security as everything is on user's computer
- Is able to learn from users history so gets better with time
- Is easy to set up and use and is more portable

Disadvantages:

- Not as accurate or having that much functionality as our current voice Assistant
- Takes space as it runs on user's computer
- Is not resource efficient as it requires the resources of the user's computer on which it is running

7.2 Applications of our System

Voice assistants are capable of providing a wide variety of services like

1. Provide information regarding current events from around the world, the weather report, information from Wikipedia, set an alarm, read out notifications and a lot more.
2. Play videos, shows and other content directly from youtube or from one of the streaming OTT platforms like Netflix, Amazon Prime, etc.
3. The Al voice assistant can be helpful for the Visually impared for smooth operations of their device.

7.3 Future Aspects / Scope of our work

1. Individualized Experiences

Since every user is unique and so are there needs so in the future it should be possible for the Voice Assistant to learn the expectation of a user when they are searching for something and produce results accordingly or if they want to see their schedule or remainders it should act like a personal assistant not as a general assistant.

2. Voice Push Notifications

Using Voice Assistant users should be able to use other different apps like scheduling meetings, booking cab and voice assistant should be able to speak or show us those notifications from many different applications.

3. Focus on Security

With the increase of dependence of a user on technologies their security should also increase in the future as Voice Assistant learns more about us it will be able to book restaurants for us to plan out our evening buy tickets so it becomes important for us to make sure that no one else has the access to our data.

REFERENCES

[1]Hideyuki Tachibana, Katsuya Uenoyama, Shunsuke Aihara, **"Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention"**

[2]Christin Jose, Yuriy Mishchenko, Thibaud Senechal, Anish Shah, Alex Escott, Shiv Vitaladevuni**, "Accurate Detection of Wake Word Start and End Using a CNN"**

[3]Giovanni Di Gennaro, Amedeo Buonanno, Antonio Di Girolamo, Armando Ospedale, Francesco A.N. Palmieri**, "Intent Classification in Question-Answering Using LSTM Architectures"**

[4]Dario Amodei, Rishita Anubhai, **"Deep Speech 2: End-to-End Speech Recognition in English and Mandarin"**

[5]Jakub Nowak(Czestochowa University of Technology), Ahmet Taspinar, Rafal Scherer(Czestochowa University of Technology), **"LSTM Recurrent Neural Networks for Short Text and Sentiment Classification"**

https://github.com/r9y9/deepvoice3_pytorch

https://github.com/Kyubyong/dc_tts

https://github.com/adiyoss/GCommandsPytorch

PLAGIARISM REPORT

AI_voice_btp2020

ORIGINALITY REPORT

%16	%9	%10	%8
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Netaji Subhas Institute of Technology Student Paper	%2
2	E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, S. V. Polyakov. "Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning", 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT), 2018 Publication	%2
3	"Progresses in Artificial Intelligence and Neural Systems", Springer Science and Business Media LLC, 2021 Publication	%1
4	Sujuan Hou, Jianwei Lin, Shangbo Zhou, Maoling Qin, Weikuan Jia, Yuanjie Zheng. "Deep Hierarchical Representation from Classifying Logo-405", Complexity, 2017 Publication	%1

5	Submitted to South Dakota Board of Regents Student Paper	%1
6	export.arxiv.org Internet Source	%1
7	S S M Saqquaf, Sarvesh Araballi, P Bhagyalakshmi, Shruti S Mahadeek, B M Varsha. "Dynamically automated interactive voice response system for smart city surveillance", 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2016 Publication	%1
8	monkeylearn.com Internet Source	%1
9	Submitted to Kookmin University Student Paper	%1
10	keithito.com Internet Source	<%1
11	Shengzhou Gao, Wenxin Hou, Tomohiro Tanaka, Takahiro Shinozaki. "Spoken Language Acquisition Based on Reinforcement Learning and Word Unit Segmentation", ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020	<%1

Publication

12	Jerry George Thomas, Sudhir P Mudur, Nematollaah Shiri. "Detecting Anomalous Behaviour from Textual Content in Financial Records", IEEE/WIC/ACM International Conference on Web Intelligence, 2019 Publication	<%1
13	towardsdatascience.com Internet Source	<%1
14	www.michaelphi.com Internet Source	<%1
15	www.coursehero.com Internet Source	<%1
16	www.analyticsvidhya.com Internet Source	<%1
17	arxiv.org Internet Source	<%1
18	Submitted to De Montfort University Student Paper	<%1
19	Submitted to Engineers Australia Student Paper	<%1
20	www.ijitee.org Internet Source	<%1
21	Ting Huang, Hongxia Wang, Yi Chen, Peisong	<%1

	He. "Chapter 9 GRU-SVM Model for Synthetic Speech Detection", Springer Science and Business Media LLC, 2020 Publication	
22	researchmap.jp Internet Source	<%1
23	Maliheh Shirvanian, Manar Mohammed, Nitesh Saxena, S Abhishek Anand. "Voicefox: Leveraging Inbuilt Transcription to Enhance the Security of Machine-Human Speaker Verification against Voice Synthesis Attacks", Annual Computer Security Applications Conference, 2020 Publication	<%1
24	Ankit Sharma, Puneet Kumar, Vikas Maddukuri, Nagasai Madamshetti et al. "Fast Griffin Lim based waveform generation strategy for text-to-speech synthesis", Multimedia Tools and Applications, 2020 Publication	<%1
25	Submitted to University of Information Technology, Yangon Student Paper	<%1
26	erp4cables.net Internet Source	<%1
27	gitlab.cvc.uab.es Internet Source	<%1

28	"Computer Vision – ECCV 2020", Springer Science and Business Media LLC, 2020 Publication	<%1
29	github.com Internet Source	<%1
30	Submitted to Universiti Tenaga Nasional Student Paper	<%1
31	Submitted to Nottingham Trent University Student Paper	<%1
32	developer.nvidia.com Internet Source	<%1
33	"Recent Trends in Image Processing and Pattern Recognition", Springer Science and Business Media LLC, 2019 Publication	<%1
34	mafiadoc.com Internet Source	<%1
35	openaccess.city.ac.uk Internet Source	<%1
36	ktree.cs.ucy.ac.cy Internet Source	<%1
37	www.macs.hw.ac.uk Internet Source	<%1
	docplayer.net	

38	Internet Source	<%1
39	digitalcommons.usu.edu Internet Source	<%1

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		