Analysis and Implementation of Deep Neural Network based Text-to-Speech (TTS) System

A **Project report** submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

by

Prathamesh Doundkar	**161307**
Apoorva Mahajan	**161393**
Shreya Killedar	**161536**
Shoumik Nandi	**161009**

Under the guidance of
Prof. (Dr.) Ashutosh Marathe

DEPARTMENT OF
ELECTRONICS & TELECOMMUNICATION ENGINEERING
VISHWAKARMA INSTITUTE OF TECHNOLOGY PUNE 2019-20

Bansilal Ramnath Agarwal Charitable Trust's

VISHWAKARMA INSTITUTE OF TECHNOLOGY, PUNE - 37

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

CERTIFICATE

This is to certify that the **Project Report** entitled **Analysis and Implementation of Deep Neural Network based Text-to-Speech (TTS) System** has been submitted in the academic year **2019-20** by

Prathamesh Doundkar	**161307**
Apoorva Mahajan	**161393**
Shreya Killedar	**161536**
Shoumik Nandi	**161009**

under the supervision of **Prof. (Dr.) Ashutosh Marathe** in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Electronics and Telecommunication Engineering** as prescribed by Savitribai Phule Pune University.

Guide/Supervisor	**Head of the Department**
Prof. (Dr.) Ashutosh Marathe	Prof. (Dr.) Shripad Bhatlawande
Signature:	Signature:

External Examiner

Name:

Signature:

cādence®

May 20, 2020

Training Certificate

<u>To Whomsoever It May Concern</u>

This is to certify that **Prathamesh Doundkar** had undergone training with Cadence Design Systems (India) Private Limited, as part of the academic curriculum from **January 27, 2020** till **May 20, 2020**.

We wish him success in all future endeavours.

for Cadence Design Systems (India) Private Limited,

Gauri

Gauri Deval
Sr. HR Representative

cādence®

May 20, 2020

Training Certificate

<u>To Whomsoever It May Concern</u>

This is to certify that **Apoorva Mahajan** had undergone training with Cadence Design Systems (India) Private Limited, as part of the academic curriculum from **January 27, 2020** till **May 20, 2020**.

We wish her success in all future endeavours.

for Cadence Design Systems (India) Private Limited,

Gauri

Gauri Deval
Sr. HR Representative

cādence®

May 20, 2020

Training Certificate

To Whomsoever It May Concern

This is to certify that **Shreya Killedar** had undergone training with Cadence Design Systems (India) Private Limited, as part of the academic curriculum from **January 27, 2020** till **May 20, 2020**.

We wish her success in all future endeavours.

for Cadence Design Systems (India) Private Limited,

Gauri

Gauri Deval
Sr. HR Representative

cādence®

May 20, 2020

Training Certificate

<u>To Whomsoever It May Concern</u>

This is to certify that **Shoumik Nandi** had undergone training with Cadence Design Systems (India) Private Limited, as part of the academic curriculum from **January 27, 2020** till **May 20, 2020**.

We wish him success in all future endeavours.

for Cadence Design Systems (India) Private Limited,

Gauri

Gauri Deval
Sr. HR Representative

ACKNOWLEDGEMENT

We wish to express our gratitude to **Prof. (Dr.) R. Jalnekar** Director, VIT Pune for providing the facilities of the Institute and for his encouragement during the course of this work. We also express deep gratitude to **Prof. (Dr.) Shripad Bhatlawande** , the Head of the Department of Electronics and Telecommunication Engineering, VIT Pune for his guidance and support.

We would like to thank our industry mentors **Mr. Ajay Homkar**, Senior Design Engineering Manager and **Mr. Saitsh Patil**, Lead Design Manager along with support of **Mrs. Gauri Deval**, Senior HR Representative at Cadence Design Systems, Pune for providing us with this opportunity guidance and help on every step of the way during the course of this project and for imparting us with invaluable knowledge.

We would also 1ike to gratefully acknowledge the enthusiastic supervision of our project guide, **Prof. (Dr.) Ashutosh Marathe** for his continuous, valuable guidance, patience, constant care, and kind encouragement throughout the project work that helped us present this project report.

ABSTRACT

Speech synthesis is the process of converting a written message in form of text to equivalent message in spoken form. A Text-To-Speech (TTS) synthesizer works as a computer-based system that should be able to read text. We propose a text to speech (TTS) method (voiceloop by facebook) that is able to convert text to speech in voices that are sampled in the wild. Unlike other systems, this approach was able to handle unconstrained voice samples and without requiring aligned phonemes or complex linguistic features extractions. It used an attention model to extract the context and also ensured support to multiple speakers. It is a memory efficient solution as a single buffer is used for all the computations.GMM based attention was used for sequence to sequence modelling. Instead of conventional RNNs, a memory buffer was employed in this system. Input encoding part using a context-free lookup table mechanism is an extremely simple approach which was incorporated hence making the model less complex. Finally WORLD vocoder was used for the actual synthesis of the speech from the feature vector.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

Text processing and speech generation are two primary components of a text to speech conversion. The main aim of the text processing component is to operate on the given input text and output appropriate sequences of phonemic units. These phonemic units are converted into the actual speech by the speech generation component either by using parameters or by selection of the best unit from an available speech corpus.Audio reading devices for blind people make use of this technique. In recent times use of text-to-speech conversion technology has gone far beyond the dissabled community to become a major additive to the fast expanding use of digital voice storage for voice mail and voice response systems.

For natural sounding speech synthesis, it is necessary that the text processing component produces a suitable sequence of phonemic units corresponding to input text. With conventional speech synthesis methods, this objective has been achieved by dividing the problem into two stages. The frontend,which converts the text into linguistic features. These linguistic features are phones, syllables, words, phrases and also utterance-level features . Backend (second part), takes linguistic features as input made available by the frontend and produces the corresponding sound.[2] Modern TTS systems are based on complex,numerous stage dependent processing pipelines, each of which may again depend on other custom features and heuristics. This paper presents the modern approach used by facebook voiceloop without requiring aligned phonemes or linguistic features.A more simplified framework using sequence-to-sequence models with attention was proposed for this TTS.

CHAPTER 2

2. LITERATURE SURVEY

Text to speech (TTS) methods are of four types:Articulatory synthesis, formant(rule-based), concatenative, statistical-parametric (mostly HMM based), and neural.Computational techniques for articulatory synthesis speech based on models of the human vocal tract and the articulation processes occurring there. It is one of the most complex approaches to use and the computational load is also more than with other familiar techniques. Advantages of articulatory synthesis are that the vocal tract models allow accurate modeling of transients due to abrupt area changes.[4].Homer Duley vocoder(1939) and Gunnar Fant's OVE synthesizer(1950) were the models that used this synthesis technique

Formant synthesis starts with acoustics, creates rules/filters to form each formant. It is derived from the concept of source-filter-model of speech. There are two structures namely, parallel and series, but for good performance a combination of these is commonly used. It is not a dynamic method because of the static rules.

Concatenative synthesis uses databases of stored speech to assemble new utterances. Different phonemes concatenated as per text. Diphone and unit selection methods are the broad categories of concatenative synthesis. Diemo Schwarz (2000) used Concatenative data based synthesis technique. In this paper they compared the musical and speech synthesis .The tone, naturalness and intelligibility contrasts were found. Manual based systems for segmentation were used. The distance function was not accurately tuned, there was no practical target related timbre space and phase coupling for speech segmentation and synthesis. The author proved that using the technique of data-driven concatenative synthesis primarily based on varying unit selection to musical synthesis application is a well grounded and efficient method.[6]

Kalyan D. Bamane review showed that the author applied a unit selection algorithm in language synthesis(marathi) ,2011. They had only worked on Marathi Language. The author used the different choice of units like words, Di-phone and tri-phones as databases. In this paper they focused deeply on Di-phone and Tri-phone. They comment that old style TTS is not particularly positive. In this paper they generated speech signals from "from scratch". Here also the most important quality of speech synthesis system is naturalness and intelligibility. They show that the result is 95% quality voice.[7]

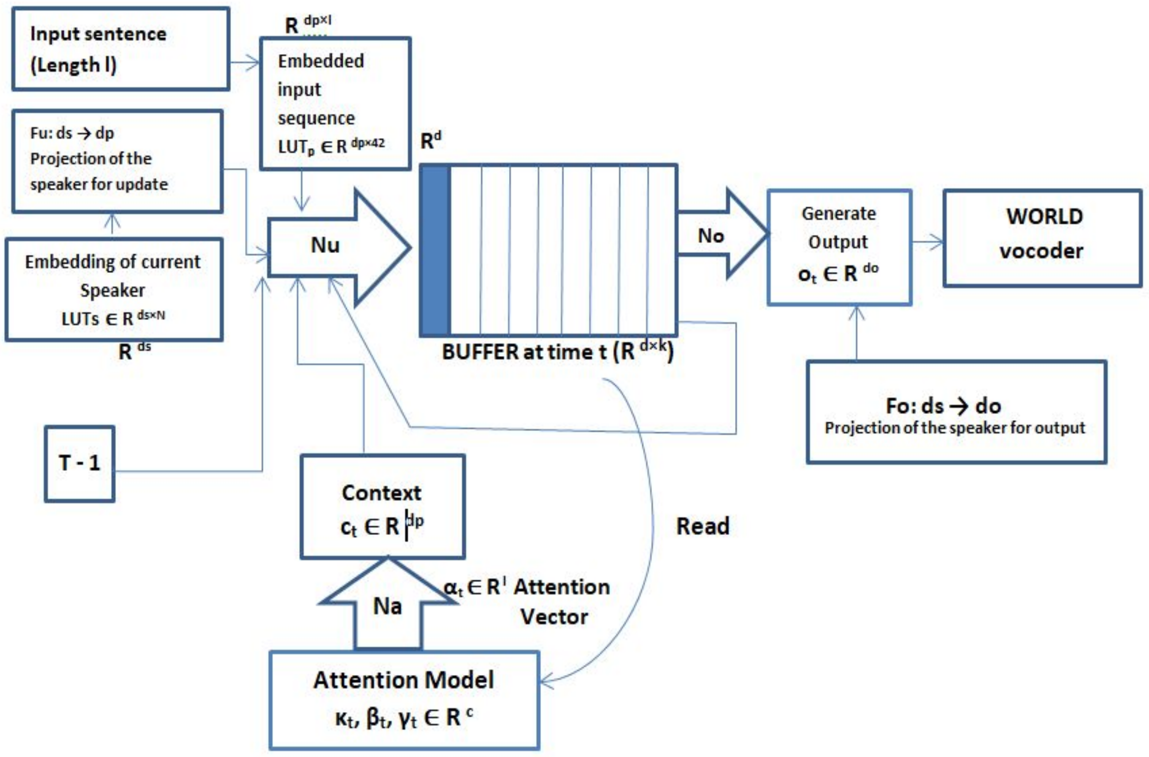
Statistical machine translation (SMT): It is distinguished by the use of machine learning methods. SMT is a data-driven method which makes use of simultaneously aligned corpora and considers translation as a mathematical reasoning based problem. In that, each sentence in the output language is a translation with probability from the input language. The more the probability, the higher is the accuracy of translation and the other way around. Catalin Ungurean (2011) had addressed the NLP and TTS. TTS automatic syllabification is necessary for lexical stress assignment, prosody generation and letter to phone conversion of the input text. They had implemented a system by using a hybrid strategy, a minimal set of rules, followed by a data driven approach. They had also used a set of phonetic transcription rules with the correctly syllabified words. Moreover, they demonstrated that lexical stress prediction can help the letter to phone process, by solving some additional ambiguities.[5]

These systems learn a fixed set of speaker embeddings and therefore only support synthesis of voices seen during training. In contrast, VoiceLoop [1] proposed an unique architecture mainly based on a fixed size memory buffer which can generate speech from voices in youtube videos.On the contrary to other TTS models, this network is trained on untranscribed speech containing echoes and background noise from a huge number of speakers. The multi speaker support along with the optimized use for single buffer memory marks the efficiency and accuracy of this model.

CHAPTER 3

3. VOICELOOP ARCHITECTURE

The reader combines the encoding of the sentence's phonemes using the attention weights to create the current context. A new representation is created by a shallow network that receives the context, the speaker ID, the previous output, and the buffer. The new representation is inserted into the buffer and the earliest vector in the buffer is discarded. The output is obtained by another shallow network that receives the buffer and the speaker as inputs. Once trained, fitting a new voice is done by freezing the network, except for the speaker embedding



Input sentence (Length l)
R dp×l
Embedded input sequence LUTp ∈ R dp×42
Fu: ds → dp Projection of the speaker for update
Embedding of current Speaker LUTs ∈ R ds×N
R ds
R d
Nu
BUFFER at time t (R d×k)
No
Generate Output ot ∈ R do
WORLD vocoder
Fo: ds → do Projection of the speaker for output
T - 1
Context ct ∈ R dp
Read
αt ∈ R l Attention Vector
Na
Attention Model κt, βt, γt ∈ R c

Figure 1: Voiceloop Architecture

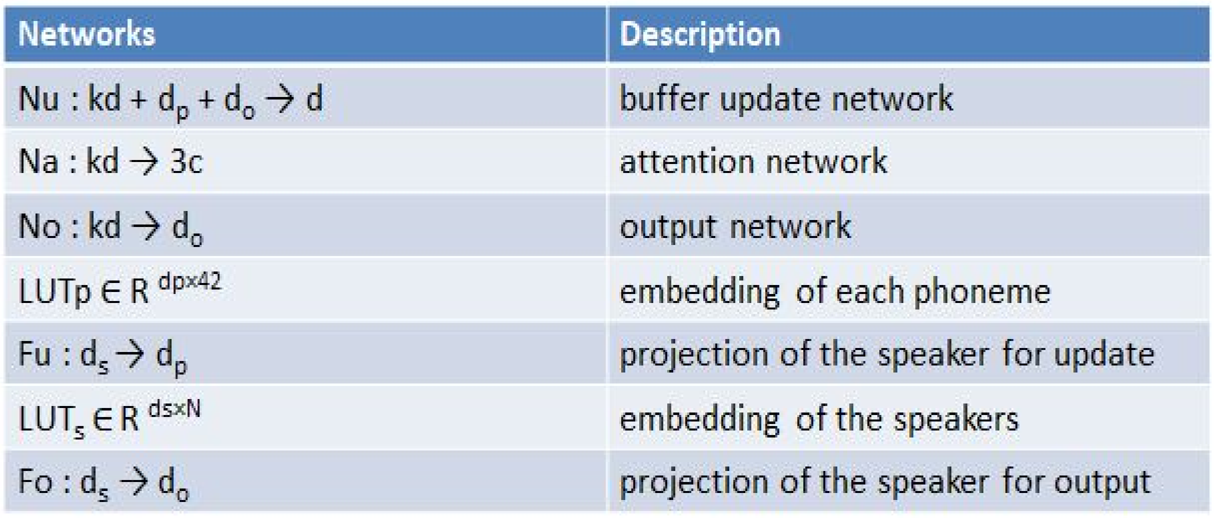
Parameter	Description	Computed as:
d	dimensionality of the buffer	dp + do
k	capacity of the buffer	20
dp	dim. of the input embedding LUT	256
do	dim. of the vocoder feature vector	63
ds	dim. of the speaker embedding	dp

c	# GMM component (attention model)	10
s1 . . . sl, $1 \leq si \leq 42$	input sequence	
l	length of the input sequence	
N	number of speakers in the training set	

Table 1: Parameters in the model

Variable	**Description**	**Computed as:**
$S_t \in \mathbf{R}^{d \times k}$	buffer at time t	$S_t[1] = u_t$; $S_t[i + 1] = S_{t-1}[i]$
$u_t \in \mathbf{R}^d$	new representation for the buffer	$Nu([S_{t-1}, [c_t + \tanh(F_u z), o_{t-1}]])$
$E \in \mathbf{R}^{dp \times l}$	embedding of the input sequence	$E[i] = LUT_p[s_i]$
$z \in \mathbf{R}^{ds}$	embedding of the current speaker	$LUT_s[id]$
$\kappa_t, \beta_t, \gamma_t \in \mathbf{R}^c$	2 attention model parameters	$Na(S_{t-1})$
$\mu_t, \sigma^2_t, \gamma'_t \in \mathbf{R}^c$	attention GMM parameters	$\mu_t = \mu_{t-1} + e^{\kappa t}$, $\sigma^2_t = e^{\beta t}$, $\gamma'_t = sm(\gamma_t)$
$\alpha_t \in \mathbf{R}^l$	attention vector at time t	
$c_t \in \mathbf{R}^{dp}$	context vector at time t	$c_t = E\alpha_t$
$o_t \in \mathbf{R}^{do}$	output vector at time t	$No(S_t) + F_o z$

Table 2: Variables used in the model



Networks	Description
Nu : $kd + d_p + d_o \rightarrow d$	buffer update network
Na : $kd \rightarrow 3c$	attention network
No : $kd \rightarrow d_o$	output network
LUTp $\in R^{dp \times 42}$	embedding of each phoneme
Fu : $d_s \rightarrow d_p$	projection of the speaker for update
$LUT_s \in R^{ds \times N}$	embedding of the speakers
Fo : $d_s \rightarrow d_o$	projection of the speaker for output

Table 3: Network input and output parameters

CHAPTER 4

4. PHONEMIZER

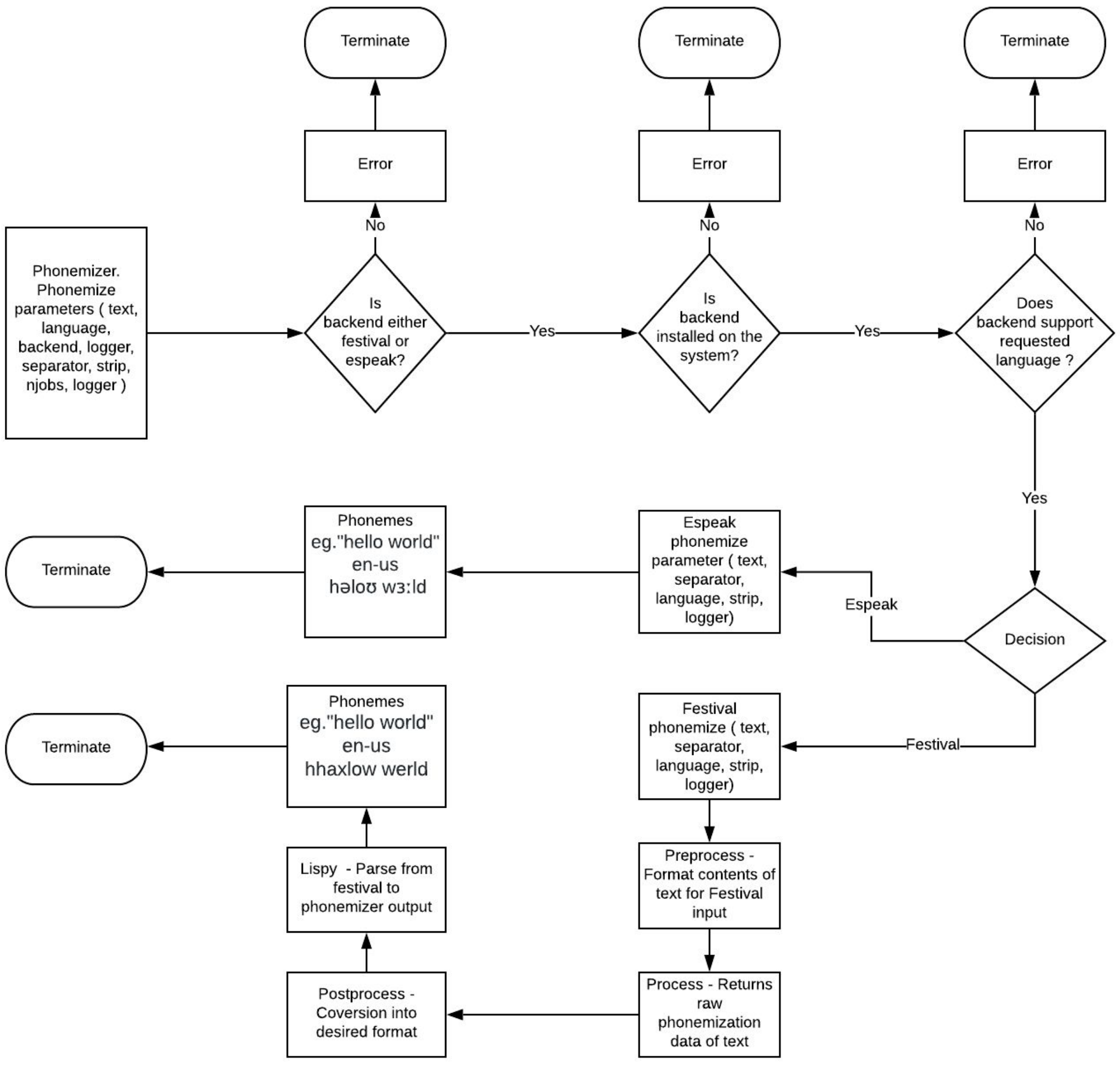
4.1 Phonemes

A phoneme is a speech sound of a word. It is the smallest unit of sound in a word. To hear the phonemes in a word, one must break the word into its sounds. This is called segmentation.

Eg. fun → /f/ /u/ /n/

In this case each phoneme is spelled by one letter. Phonemes can be spelled by 1 - 4 letters.

4.2 Phonemizer



Phonemizer. Phonemize parameters (text, language, backend, logger, separator, strip, njobs, logger)
Is backend either festival or espeak?
Yes
No
Error
Terminate
Is backend installed on the system?
Yes
No
Error
Terminate
Does backend support requested language ?
No
Error
Terminate
Yes
Decision
Espeak
Espeak phonemize parameter (text, separator, language, strip, logger)
Phonemes eg."hello world" en-us həloʊ wɜːld
Terminate
Festival
Festival phonemize (text, separator, language, strip, logger)
Preprocess - Format contents of text for Festival input
Process - Returns raw phonemization data of text
Postprocess - Coversion into desired format
Lispy - Parse from festival to phonemizer output
Phonemes eg."hello world" en-us hhaxlow werld
Terminate

Figure 2: Phonemizer Flow Diagram

The phonemizer allows simple phonemization of words and texts in many languages. It provides both the phonemize command-line tool and the Python function **phonemizer.phonemize**.

The model here uses two backends: espeak and festival

- espeak supports a lot of languages and IPA (International Phonetic Alphabet) output.
- festival currently supports only American English. It uses a custom phoneset, but it allows tokenization at the syllable level.

4.3 Phonemizer Code Flow

1. Phonemizer block takes two inputs from the user namely text and separators. There are three separators : phones, syllables and words.

2. After taking the two inputs from the user the code enters **phonemizer.phonemize** which has seven parameters : text, language, backend, separator, strip, njobs, logger.

3. First it ensures if the backend is espeak or festival. If the backend is neither of them, an error occurs and the program is terminated.

4. Then it checks if the selected backend is installed in the system or not. If the selected backend is not installed, an error occurs and the program is terminated.

5. Finally it checks if the language mentioned in the argument of phonemize is supported by the selected backend. If the language is not supported by the backend, an error occurs and the program is terminated.

6. If none of the above errors occur, the text entered is converted into string type if it isn't already a string. The model trained was trained with context to **festival**. Hence the default backend is set to festival. The code then enters the phonemize in **festival**. Festival uses Latin1 encoding.

7. In festival phonemize the assert language function checks if the language is supported or not. In this case the language is 'en-us' which is supported by festival. The festival script is imported.

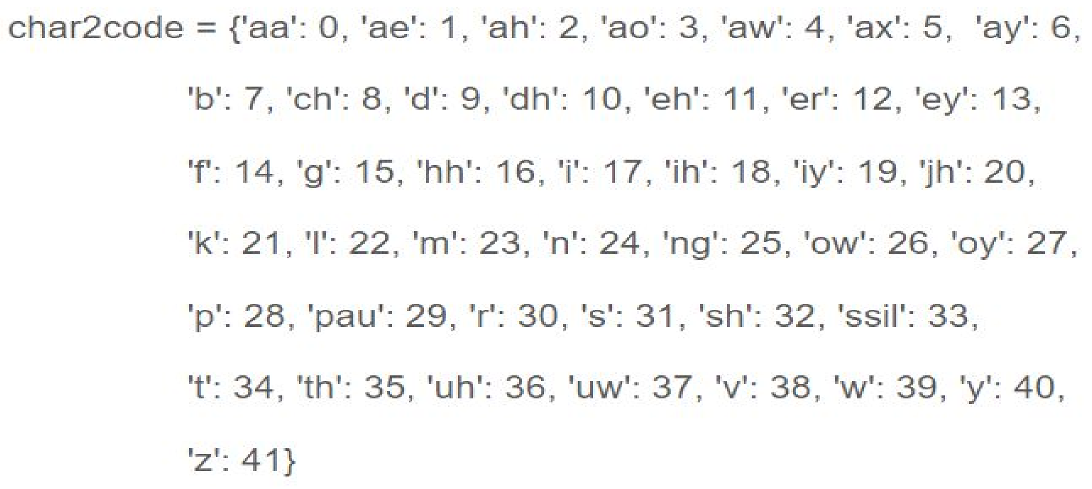
8. The code enters the preprocessing stage. The only argument here is the text . The text is brought into the format in which the processing can take place - the format in which the text can be used as an input in the .scm file to break it down into various characters and break it into SlyStructure relation tree. The text is taken line by line and is cleaned using the **_cleaned** function. Only the text is kept intact. Each line of the text is then put into double quotes to finally bring it in the required format for processing using **the _double_quoted function.**

9. Preprocessing is followed by processing. In processing 3 arguments are passed - Preprocessed Output , Script , Logger. A tempfile is created for writing the data in it. Operation can be carried out on the file using the word data. The data in the tempfile along with the script file is taken and a second tempfile is created. This file used scm as the keyword to carry out operations on the file. Using cmd we log into the file. The festival conversion into SylStructure tree needs certain formatting. The '+' are replaced by ' ' .

10. The above process is followed by processing. It converts the tree into unicode format . It segregates each line. It considers one line at a time. It checks if the line is empty or not. Since in preprocessing we removed all the empty lines, no empty lines are expected. If it comes across an empty line it considers the process of conversion into phonemes is complete.

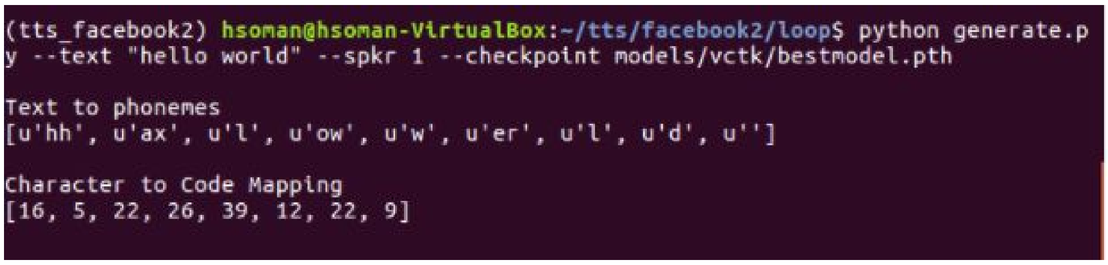
11. The lispy block is used within postprocessing to arrange the output in lists of syllables.

12. The output of festival is of the type string. The output is the mapped out phoneme by phoneme into code using a dictionary which is predefined according to the trained model. The final output is a list of codes.



```
char2code = {'aa': 0, 'ae': 1, 'ah': 2, 'ao': 3, 'aw': 4, 'ax': 5,  'ay': 6,

             'b': 7, 'ch': 8, 'd': 9, 'dh': 10, 'eh': 11, 'er': 12, 'ey': 13,

             'f': 14, 'g': 15, 'hh': 16, 'i': 17, 'ih': 18, 'iy': 19, 'jh': 20,

             'k': 21, 'l': 22, 'm': 23, 'n': 24, 'ng': 25, 'ow': 26, 'oy': 27,

             'p': 28, 'pau': 29, 'r': 30, 's': 31, 'sh': 32, 'ssil': 33,

             't': 34, 'th': 35, 'uh': 36, 'uw': 37, 'v': 38, 'w': 39, 'y': 40,

             'z': 41}
```

Figure 3: Character to code dictionary



```
(tts_facebook2) hsoman@hsoman-VirtualBox:~/tts/facebook2/loop$ python generate.p
y --text "hello world" --spkr 1 --checkpoint models/vctk/bestmodel.pth

Text to phonemes
[u'hh', u'ax', u'l', u'ow', u'w', u'er', u'l', u'd', u'']

Character to Code Mapping
[16, 5, 22, 26, 39, 12, 22, 9]
```

Figure 4: Input and output of phonemizer

CHAPTER 5

5. ATTENTION MODEL

5.1 Introduction to Attention Model

The speaker id is mapped (using a lookup table) onto a speaker embedding (Z)which is also trained by the neural network. The attention is computed using Graves Gaussian mixture model(GMM) based monotonic attention mechanism.GMM is a mixture density model .The basic approach of mixture density networks is to use the outputs of a neural network to parameterise a mixture distribution. The mixture weight outputs are normalised with softmax function to ensure they form a acceptable discrete distribution, and the other outputs are passed through suitable functions to keep their values within meaningful range (for example the exponential function is typically applied to outputs used as scale parameters, which must be positive).Mixture density network are trained by maximising the log probability density of the targets under the induced distributions.[17][18].The prior buffer state is fed to the GMM based attention model which outputs the gaussian parameters.Further the attention weights are calculated by performing operations on the obtained parameters.These weights are multiplied with input embedding to give the context vector as the output .[2][16].At each time step a new frame is generated using Nu, which takes as input the buffer from the previous time step, the context vector calculated using attention, and the previous output. The new buffer frame is added to the buffer in a FIFO(first in first out) manner.
We achieve speaker dependence by adding a projection of the speaker embedding Z computed earlier to the new buffer frame. The output is generated using No, which takes as input the entire buffer and adding the projection of Z. The output of the network are vocoder features, which when fed into a WORLD vocoder to produce sound.

5.2 Working of Attention Model

At each output time point $t = 1, 2, . . . ,$ the attention network Na receives the buffer from the previous time step S_{t-1} as input and outputs the GMM priors γt, shifts κt, and log-variances βt. For a GMM with c components, each of these is a vector in R c . Na has one hidden layer, of dimensionality dk 10 and a ReLU activation function for the hidden layer.Then, the softmax function is applied to the priors. The means of the GMMs are increased.
The variances are computed as σ 2 t = exp(βt). For each GMM component $1 \leq i \leq c$ and each point along the input sequence $1 \leq j \leq l$, we then compute: $\varphi[i, j]$. The attention weights αt are computed for each location in the sequence by summing along all c components.The context vector ct is then computed as weighted sum of the columns of the input sequence embedding matrix E as ct = Eαt.

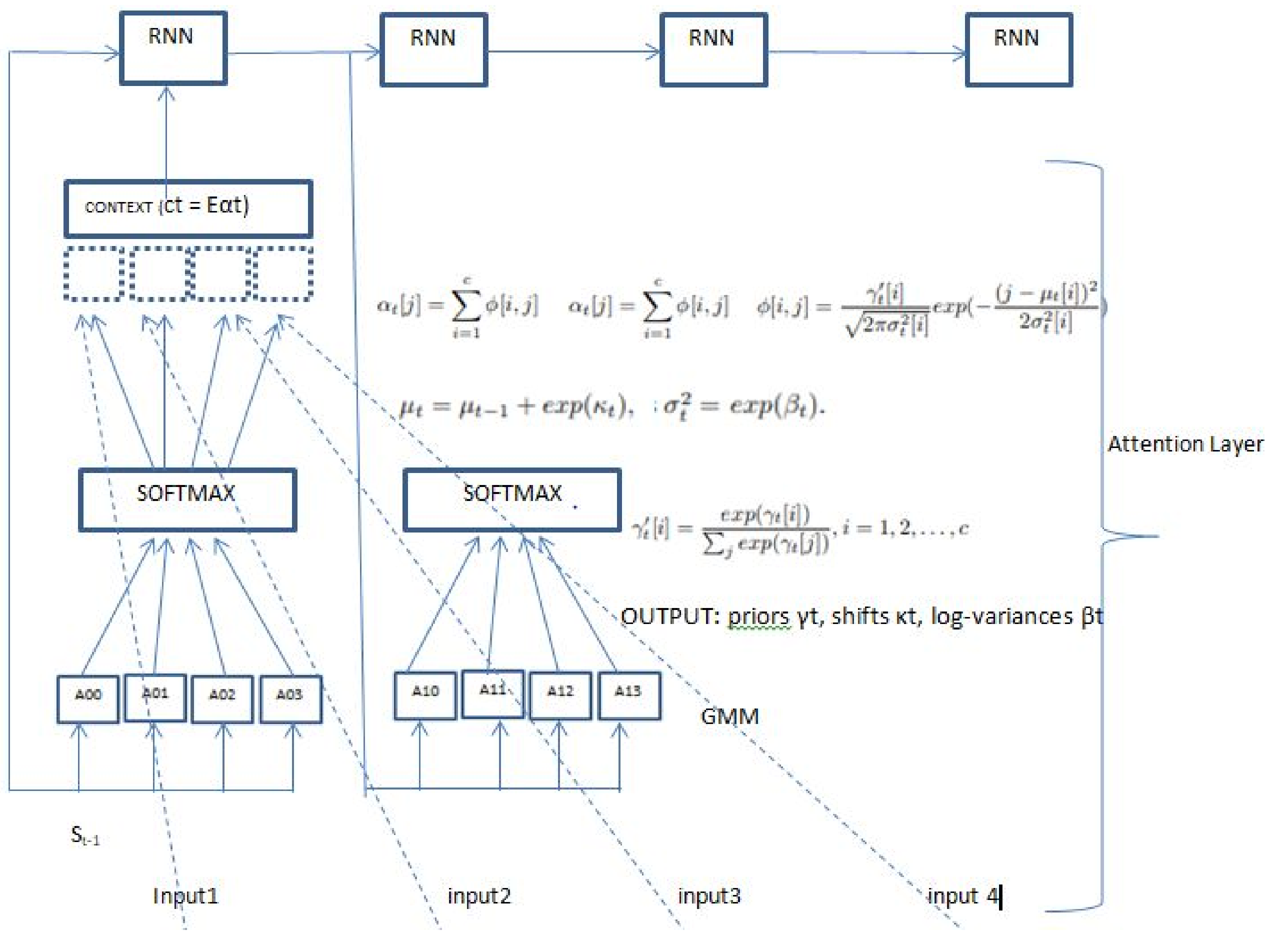

RNN
RNN
RNN
RNN
CONTEXT (ct = Eαt)
$\alpha_t[j] = \sum_{i=1}^{c} \phi[i,j]$
$\alpha_t[j] = \sum_{i=1}^{c} \phi[i,j]$
$\phi[i,j] = \frac{\gamma'_t[i]}{\sqrt{2\pi\sigma_t^2[i]}} exp(-\frac{(j-\mu_t[i])^2}{2\sigma_t^2[i]})$
$\mu_t = \mu_{t-1} + exp(\kappa_t), \; \sigma_t^2 = exp(\beta_t).$
Attention Layer
SOFTMAX
SOFTMAX
$\gamma'_t[i] = \frac{exp(\gamma_t[i])}{\sum_j exp(\gamma_t[j])}, i = 1, 2, \ldots, c$
OUTPUT: priors γt, shifts κt, log-variances βt
A00
A01
A02
A03
A10
A11
A12
A13
GMM
S_{t-1}
Input1
input2
input3
input 4

Figure 5: Attention model flow diagram

CHAPTER 6

6. WORLD VOCODER

WORLD is a high quality speech synthesis system that also meets real-time processing requirements. It contains three algorithms for obtaining three speech parameters and a synthesis algorithm that takes these parameters as input. Fundamental frequency F0 estimation is done using DIO algorithm. Second, the spectral envelope is estimated with CheapTrick, which uses not only the waveform but also the F0 information. Third, the excitation signal is estimated with PLATINUM and used as an aperiodic parameter. PLATINUM uses the waveform, F0, and spectral envelope information. [8]

Extraction of these parameters from speech helps in mapping the data with the text in sentences during training whereas synthesizer is used in the model to synthesize speech based on parameters obtained at the end of the attention model. The parameters are however modified before supplying to the synthesis function. The pseudo code is

```
for (int i = 0; i < f0_length; ++i) {
  for (int j = 0; j <= fft_size / 2; ++j)
  spectrum1[j] = log(spectrogram[i][j]);
   interp1(freq_axis1, spectrum1, fft_size / 2 + 1, freq_axis2,
  fft_size / 2 + 1, spectrum2);
  for (int j = 0; j <= fft_size / 2; ++j)
  spectrogram[i][j] = exp(spectrum2[j]);
  if (ratio >= 1.0) continue;
  for (int j = static_cast<int>(fft_size / 2.0 * ratio);
          j <= fft_size / 2; ++j)
  spectrogram[i][j] =
  spectrogram[i][static_cast<int>(fft_size / 2.0 * ratio) - 1];
```

DIO F0 estimation algorithm consists of three steps. The first step is to low-pass filter the signal with different cutoff frequencies. If the filtered signal only consists of the fundamental component, it forms a sine wave with a period of T0, which is the fundamental period. Since the target F0 is unknown, many filters with different cutoff frequencies are used in this step. The second step is to calculate the F0 candidates and their reliability in each filtered signal. A signal that consists of only the fundamental component forms a sine wave i.e., the four intervals of the waveform - the positive and negative zero-crossing intervals and peak and dip intervals, have the same value. Their standard deviation is therefore associated with the reliability measure and their average is defined as an F0 candidate. In the third step, the candidate with the highest reliability is selected. CheapTrick is based on the idea of pitch synchronous analysis and uses a Hanning window with the length of 3T0. First, the power spectrum is calculated on the basis of the windowed waveform. The overall power of the windowed waveform is temporarily stabilized. Then, the power spectrum is smoothed with a rectangular window of width 2ω0/3.

After that special liftering is carried out for smoothening logarithmic power spectrum and removing time components and for spectral recovery respectively. PLATINUM windows the waveform by using a window with a length of 2T0.

INPUT	OUTPUT
F0: F0 contour	Y: calculated speech
F0_length: Length of F0	
Spectrogram: estimated by CheapTrick	
Fft_size	
Aperiodicity: based on D4C	
Frame_period: Temporal period used for analysis	
Fs: sampling frequency	
Y_length: Length of output signal (memory for output signal has been allocated in advance)	

Table 4 : Inputs Outputs of Synthesis Function

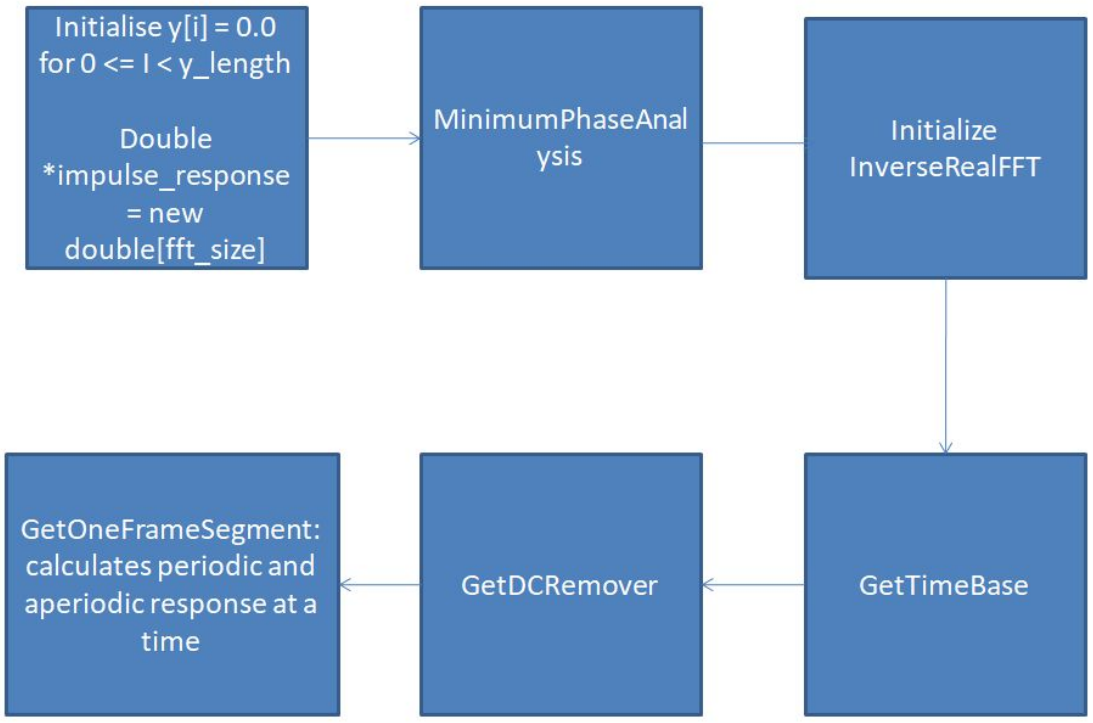

Initialise y[i] = 0.0 for 0 <= I < y_length
Double *impulse_response = new double[fft_size]
MinimumPhaseAnalysis
Initialize InverseRealFFT
GetTimeBase
GetDCRemover
GetOneFrameSegment: calculates periodic and aperiodic response at a time

Figure 6: Block diagram of Synthesis Function

CHAPTER 7

7. TESTING

Selecting test data wisely to evaluate a text to speech system is important. Here a test data was created which includes all possible variations including numerals, dates, address, salutations, tongue twisters, homographs, difficult words, semantically unpredictable words, punctuations and sentences which evaluate prosody. Such data helps in verifying the system specifications, features, and in defining its limitations. Module testing was done for diagnosing the issues that may degrade the performance of specific modules.
Our system consists of four main modules: Phonemizer, Context generation, Buffer updation and Output generation. These modules were evaluated over different variations present in the test data.

CHAPTER 8

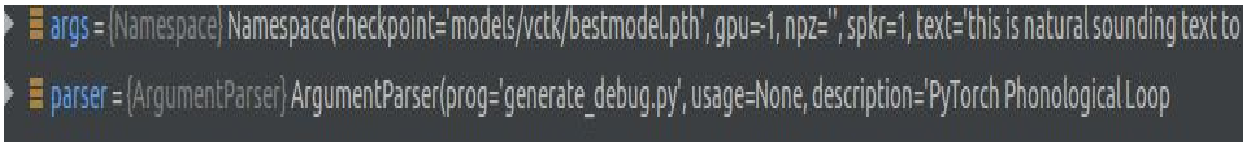
8.RESULTS AND ANALYSIS

Command line script for execution of code:

Cmd line arguments specifying pretrained model location, speaker id and the input text

text = “this is natural sounding text to speech converter”

Spkr = 1



```
args = {Namespace} Namespace(checkpoint='models/vctk/bestmodel.pth', gpu=-1, npz='', spkr=1, text='this is natural sounding text to
parser = {ArgumentParser} ArgumentParser(prog='generate_debug.py', usage=None, description='PyTorch Phonological Loop
```

Total amount of MACs, Load-Stores and Memory required was calculated for each sub system.

Total MACs : 12753470

Total Load-Stores : 25505454

Total memory(Persistent + Scratch + Tables) : 51403648 Bytes

This is calculated for the purpose of selecting the appropriate core, on which this system will be implemented.

CHAPTER 9

9. CONCLUSION AND FUTURE SCOPE

In experiments, single-speaker TTS and multi-speaker TTS along with speaker identification (ID) were tested on the model,which showed that the proposed approach outperforms baselines, namely, Tacotron and Char2wav.[20][2]. Finally, challenging Youtube data with background noise was used to train the model ,which showed promising results.

Pros:

1. It uses relatively simple and less number of parameters by using shallow fully-connected neural networks.

2. Using shifting buffer memory gives a novel solution for memory management..

3. The proposed approach outperforms baselines in several tasks, and the ability to fit to a novel speaker is quite accurate.

Cons:

1. Using the dataset the sentences containing the phoneme 'zh' could not be processed.

2.More efficient use of buffers is possible as all 20 buffer elements are not being used at each time instance.

3.Less accuracy for exceptional cases .It can be improved with training the model with huge and precise data but at the cost of more storage and time.

CHAPTER 10

10. REFERENCES

[1]Yaniv Taigman, Lior Wolf, Adam Polyak and Eliya Nachmani Facebook AI Research,"Voice Loop: Voice fitting and synthesis via a phonological loop", In Proc. International Conference on Learning Representations (ICLR),1 Feb 2018

[2]Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio. Char2wav: End-to-end speech synthesis. In ICLR workshop, 2017.

[3]Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, et al. Deep voice: Real-time neural text-to-speech. In Proc. of the 34th International Conference on Machine Learning (ICML), 2017b.

[4] O'Malley, M. H. (1990). Text-to-speech conversion technology. Computer, 23(8), 17–23. doi:10.1109/2.56867

[5]. Ungurean, D. Burileanu, "An advanced NLP framework for high-quality Text-to-Speech synthesis," SpeD 2011, pp. 1-6, Braşov,Romania,18-21 May 2011

[6] Diemo Schwarz. A System for Data-Driven Concatenative Sound Synthesis. Digital Audio Effects (DAFx), Dec 2000, Verona, Italy. pp.97-102. ⟨hal-01161115⟩

[7]KD Bamane, KN Honwadkar,Marathi speech Synthesized Using Unit selection Algorithm, Computer Engineering and Intelligent Systems ISSN, 2011

[8]WORLD: A Vocoder-based High-Quality Speech Synthesis System for Real-Time Applications - Masanori MORISE et. al, July 2016

[9] REFERENCE Manual for Speech Signal Processing Toolkit ver 3.9, December 25, 2015

[10]Robert L Weide. The CMU pronouncing dictionary. URL: http://www. speech. cs. cmu. edu/cgibin/cmudict, 1998.

[11]Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit, 2017.

[12]Zhizheng Wu, Oliver Watts, and Simon King. Merlin: An Open Source Neural Network Speech Synthesis System, pp. 218–223. 9 2016.

[13]Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio. Char2wav: End-to-end speech synthesis.

[14]Firoj Alam, Mumit Khan: Bangla Text to Speech using Festival

[15]Ramanpreet Singh, Dharamveer Sharma: An Improved System for Converting Text into Speech for Punjabi Language using eSpeak

[16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves,

Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw

audio. arXiv preprint arXiv:1609.03499, 2016.

[17]Alex Graves. Generating sequences with recurrent neural networks., arXiv:1308.0850v5 [cs.NE] 5 Jun 2014

[18]Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

[19]Paul Michel, Okko Rasanen, Roland Thiolliere, Emmanuel Dupoux: Blind Phoneme Segmentation With Temporal Prediction Errors, July 2017

[20]Yuxuan Wang,R. J.Skerry-Ryan,Tacotron:Towards End-to-End Speech Synthesis,INTERSPEECH 2017