Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network

By

Chala Sembeta Tashoma

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2021

Adama, Ethiopia

Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network

By

Chala Sembeta Tashoma

Advisor: Mesfin Abebe (PhD)

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2021

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled **"Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network"** is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation

Chala Sembeta Tashoma _____ _____

Name of the student Signature Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled **"Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network"** prepared under my guidance by **Chala Sembeta Tashoma** submitted in partial fulfillment of the requirements for the degree of Mater's of Science in **Computer Science and Engineering.**

Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

Mesfin Abebe (PhD) _____ _____

Major Advisor Signature Date

Approval of Board of Examiners

I, the advisors of the thesis entitled **"Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network"** and developed by **Chala Sembeta Tashoma**, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Mesfin Abebe (PhD) _____ _____

Major Advisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by **Chala Sembeta Tashoma** have read and evaluated the thesis entitled **"Text to Speech Synthesizer for Afaan Oromo Using Deep Neural Network"** and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in **Computer Science and Engineering**.

_____ _____ _____

Chairperson Signature Date

_____ _____ _____

Internal Examiner Signature Date

_____ _____ _____

External Examiner Signature Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____ _____ _____

Department Head Signature Date

_____ _____ _____

School Dean Signature Date

_____ _____ _____

Office of Postgraduate Studies, Dean Signature Date

ACKNOWLEDGEMENTS

Waanti hundinuus isumaan ta'e; waan ta'e keessaas isa malee kan ta'e tokko illee hin jiru.

(W. Yohaannis 1:3)

First and foremost**,** I would like to thank **GOD**, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research advisor, **Dr**. **Mesfin Abebe**, for his continuous support, patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. It was a great privilege and honor to work and study under his guidance.

I am so grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. My Special thanks goes to the man who I now call my second father, **Obbo Taakkalaa Hirphaa,** for his great role in my life, encouragement, and numerous sacrifices he has made to give me the life that I have

I thank my fellow lab mates in Adama Science and Technology University **Data Science Special Interested Group**, for the fruitful discussions, for those sleepless nights we spent working together before deadlines, and for all the funny moments we have experienced in the last two years.

I feel happy to thank Mettu University for sponsoring my studies. Finally, my thanks goes to all the people who have supported me to complete the research work directly or indirectly.

Dedication

I would like to dedicate this thesis to my beloved family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ACRONYMS ND ABBREVIATIONS

ANN:	Artificial Neural Network
AOTTS:	Afaan Oromo Text-to-Speech
API:	Application Programming Interface
bB:	Decibel
BRNN:	Bidirectional Recurrent Neural Network
CNN:	Convolutional Neural Network
CPU:	Central Processing Unit
CSV:	Comma-separated Values
CTC:	Connectionist Temporal Classification
CUDA:	Computer Unified Device Architecture
DV3	Deep Voice 3
DNN:	Deep Neural Network
DSP:	Digital Signal Processing
GLA:	Griffin-Lim Algorithm
GPU:	Graphical Processing Unit
GRU:	Gated Recurrent Unit
GTA:	Ground Truth Alignment
HMM**:**	Hidden Markov Model
LPC:	Linear predictive Code
LSTM:	Long Short Term Memory
MFCC:	Mel frequency cepstral Coefficients
MLP:	Multi-layer Perceptron
MOS:	Mean Opinion Score
NLP:	Natural Language Processing
NSW :	Non-Standard Words
PCM	Pulse-code Modulation
RNN:	Recurrent Neural Network
SSP:	Statistical Parametric Speech Synthesis
STFT	Short-time Fourier Transform
T2	Tacotron 2
TPU	Tensor Processing Unit
TTS:	Text-to-Speech

ABSTRACT

In a world where technology is emerging at an exponential rate, Speech synthesis technology is already a part of our everyday lives. Text-to-speech (TTS) synthesis systems are concerned with the artificial generation of a natural and intelligible human voice from given text transcriptions. The text-to-speech system serves as an assistive tool for people with visual impairments and reading disabilities allowing them to listen to written words such as online webpages, books, newspapers articles, and textbooks on a variety of devices. Despite the potential applications of the text-to-speech systems, it was a language-dependent discipline and most of the attempts are concerned with resourceful languages specifically the English language. Afaan Oromo is one of the under-resourced languages that have a shortage of previously existing language resources for developing a text-to-speech system. In this study, we have collected and prepared a speech dataset containing 8076 text and audio pairs from legitimate sources to develop a text-to-speech synthesizer for Afaan Oromo. Apart from standard words and names, the proposed model incorporates non-standard words including numbers, abbreviations, currency, and acronyms. The study focuses on the use of the deep neural network technique which is a machine learning algorithm implemented by several layers of neural networks. The deep neural network is chosen for this work because it can map complex linguistic features into acoustic feature parameters. Several experiments are conducted to determine the best performing model among the Tacotron 2 which is a recurrent neural network model and Deep Voice 3, a fully convolutional neural network model. The objective and subjective evaluations are carried out to assess the performance of the models. For objective evaluation, we used the attention error test and for subjective evaluation, we used the mean opinion score (MOS) test. From objective evaluation, Tacotron 2 made only 2 attention errors while Deep voice 3 made 16 out of 148 words in the evaluation sentence list. In addition, we found the MOS result of 4.32 and 4.21 out of five for Tacotron 2, and 3.28 and 3.02 out of five for Deep Voice 3 in terms of intelligibility and naturalness respectively. From evaluation results, we conclude that that the Tacotron 2, based on a recurrent neural network model provides an encouraging result, which makes our model appropriate for applications that need the text-to-speech service such as recommendation systems, telephone inquiry services, and smart educations.

Keywords: *text-to-speech, Tacotron 2, Deep voice 3, Afaan Oromo, Mean Opinion Score, Speech Processing, Deep Neural Network*

CHAPTER ONE

1. INTRODUCTION

1.1 Background of the Study

Speech synthesis mostly refers to the Text-to-Speech (TTS) system is the artificial generation of equivalent audio from the given text transcription. It is an interdisciplinary technology that takes into account several disciplines such as acoustics, linguistics, digital signal processing, and statistics. It is closely related to speech recognition, which converts speech into text, and machine translation, which translates text or speech in one language into text or speech in another (Ning et al., 2019).

The speech synthesizers system being considered in this study incorporates models of the human vocal tract and other human voice characteristics to produce a completely "synthetic" human voice output corresponding to input text. Since it is difficult to store pre-recorded audio clips of all words in one language, and for this reason, it is better to have the automatic pronunciation of words and sentences in Text-To-Speech systems. Nowadays Siri[1], Alexa[2] , and the Google assistant[3] are the most widely used personal assistants, these assistants have all been directly equipped with speech synthesis to be able to respond to the user.

The Text-to-Speech (TTS) synthesizer system consists of two main phases. The first one is the text analysis phase, where the text input is mapped into their phoneme or other linguistic representation. This process is also called natural language processing (NLP). The second one is speech analysis that generates the speech waveforms, where the phonetic and prosodic information are used to produce the output. This process is also called digital signal processing (DSP).

It is known that the qualities of modern speech synthesis systems are assessed by their naturalness and intelligibility. Naturalness measures how closely the synthesized speech resembles real human speech. Intelligibility, on the other hand, indicates how reliably the

[1] Siri is a virtual assistant that is part of Apple Inc.'s that offer you a seamless way of interacting with your apple products.
[2] Alexa is Amazon's cloud-based voice-controlled assistant that turns words into actions. Used in devices from Amazon and third-party device manufacturers.
[3] Google Assistant is an AI–powered virtual assistant proposed by Google mainly available on mobile and smart devices.

generated speech can be understood. The challenges in the TTS field lies in maximizing these aspects which is the main development goal of the discipline.

In the text to speech synthesis domain, concatenative speech synthesis directly concatenates the units (phonemes) in the large database and outputs a continuous speech stream was the state-of-the-art for many years (Hunt & Black, 1996). The concatenated units can be in the form of phones such as half-phones, diphones, syllables, words, morphemes, phrases, or sentences. Even though it provides high-quality speech output it often suffers from audible (glitches) in the output due to imperfect concatenation of the units. Generally, it is expensive and needs a larger database to include all possible utterances in the language.

Statistical parametric speech synthesis that uses the Hidden Markov Model (HMM) (Tokuda et al., 2002) showed an increase in popularity over time. It extracts speech parameters from the speech database, trains them, and produces the sound equivalent to the input text. There is no need for large a speech database as the concatenative method because it can generate speech that is not included in the original dataset by predicting the parameter values for a new context. The HMM-based synthesis requires a laborious feature engineering and deep domain knowledge and produces speech that is more robotic than speech produced by concatenative speech synthesis.

1.2 Motivation of the Study

According to the 2007 Ethiopian Statistical Agency, the total number of Afaan Oromo speakers is estimated to be more than 40 million ranging from Ethiopia to Tanzania. In addition to this, the language is also used as a medium of instruction in the primary schools and as an official language in Oromia region and some regional offices in Addis Ababa as well.

Text to speech synthesizer removed the barriers for peoples who have a wide range of disabilities such as visual impairment, dyslexia, pre-literate children, and speech impairment. Despite the advancement of TTS technologies, most TTS are developed for resourceful languages. However, TTs developed for one language are not used for another language because of language-dependent features that need to be considered while developing the synthesizer system. Because of the lack of high-quality speech data and the absence of language-specific text processing tools developing a good text-to-speech synthesizer for under-resourced languages like Afaan Oromo remains a challenging task.

Therefore, the above fact motivated the researcher to develop a deep neural network-based speech synthesizer that provides new and easy methods for feature engineering and smooth transitions in the synthesized speech; making it sound natural and human-like.

1.3 Statement of the Problem

With the progress of speech synthesis systems, human-computer interaction is improving over time. The ultimate goal of the Speech synthesis system is to generate natural sound and intelligible speech with small resource requirements. The speech synthesis system is language-dependent, for this reason, speech synthesis developed for one language is not applicable for others.

So far, many functional speech synthesizers are developed for resourceful languages such as English (Tokuda et al., 2002), Vietnam (Thu et al., 2016), and Macedonia (Mishev et al., 2020). Local researchers have been able to implement a text to speech synthesizer model for local languages such as Afaan Oromo (Morka, 2001; Samson, 2011; Wosho, 2021), Tigrigna (Araya Keletay & Seid Worku, 2020), Amharic (Zegeye, 2010) and Wolayta language (Gebreselassie, 2009). Text-to-speech synthesis techniques such as concatenative and statistical parametric speech based on the Hidden Markov model (HMM) have been studied and implemented by local researchers for Afaan Oromo.

Samson Tadesse (Samson, 2011) developed a model for Afaan Oromo text-to-speech synthesizer using a di-phone and triphone-based concatenative approach. But, there are various limitations in his work including distortion from discontinuities in concatenation points, the memory requirements are typically very high to incorporate every feature of the language, hence it requires huge data for training and recording speech which is time-consuming. Moreover, in this work, the researcher did not consider the non-standard words, and epenthesis, which are necessary parts of the language. The article concludes that the corpus preparation affected the overall performance because of noise while recording the sound.

Muhidin Kedir (Wosho, 2021) developed Statistical parametric speech synthesis based on HMM techniques. The researcher did not consider every feature of language due to the requirement of much feature engineering and deep linguistic knowledge. As we can understand from the test sentences used by the researcher, non-standard words (NSWs) such as abbreviated words, numbers, currency, dates, and punctuation marks were not considered

in this work. The size of the dataset is not enough to develop a more natural speech synthesizer only a few sentences are used for training in this work.

A speech synthesis system that will lead to hybrid techniques that incorporate both the benefits of statistical parametric speech based on HMM and concatenative synthesis method is a deep learning method or deep neural network (DNN) based speech synthesis technique. This technique is implemented for different foreign languages and has shown good performance in generating synthetic speech and it maps complex linguistic features into acoustic feature parameters. It also alleviates the laborious works of speech synthesis such as feature engineering and human annotation (Ning et al., 2019). However, this technique is not yet implemented for Afaan Oromo.

As there is a limitation of language resources and linguistic expertise in Afaan Oromo, feature engineering is very challenging and needs deep language expertise. For this reason, we aimed to apply the deep neural network that maps Afaan Oromo linguistic features extracted into acoustic features with limited human interference. The text-to-speech synthesis based on a deep neural network accepts Afaan Oromo characters as an input alleviating the need for the explicit character to phoneme conversion that needs an external phoneme dictionary or a pre-trained grapheme-to-phoneme model.

This research tries to answer the following research questions:

- What is the overall performance of the Deep Neural Network technique for Afaan Oromo on single-speaker speech dataset?
- How the performance of Afaan Oromo text-to-speech synthesizer can be improved?
- To what extent Afaan Oromo non-standard words are synthesized by Afaan Oromo text-to-speech synthesizer?

1.4 Objectives of the Study

1.4.1 General Objective

The general objective of this study is to develop a text-to-speech synthesizer model for Afaan Oromo using a deep neural network.

1.4.2 Specific Objectives

The following specific objectives have to be achieved to fulfill the above general objective:

- To review works done so far in the area of text to speech synthesis both for local and foreign languages and their approaches.
- To study the morphological and phonological structure of Afaan Oromo.
- To collect and prepare the Afaan Oromo speech dataset.
- To select an appropriate deep neural network based TTS model that can be trained well on our speech dataset.
- To develop a text-to-speech synthesis model for Afaan Oromo using a deep neural network.
- To evaluate the performance of the proposed text-to-speech synthesis model.

1.5 Scope and Limitations of the Study

1.5.1 Scope of the Study

This study focuses on the development of a TTS synthesizer for Afaan Oromo using a Deep Neural Network. As there is no available speech dataset for Afaan Oromo we prepared a speech dataset by using texts collected from news websites, non-fiction books, and Holy Bible with its corresponding audio recorded by a single speaker in a soundproof room. The study aimed to train the speech dataset on convolutional and recurrent-based models by incorporating the non-standard words including the abbreviated words, numbers, currency, acronyms, and punctuation marks. The scope of the study includes the use of objective and subjective evaluation as a model performance evaluation metrics.

1.5.2 Limitations of the Study

In this study, since only single-speaker utterances are prepared and used for training the multi-speaker technique is not supported. The diverse regional dialects are not taken into consideration. The voice cloning techniques such as speaker adaptation and speaker embedding were not considered in this work. Because of the need for linguistic expertise, the study did not consider the phoneme level embedding though it might improve the performance of the model. Some non-standard words are not considered in this study because of the difficulty to write a set of rules for handling them.

1.6 Applications of the Study

Applications fields of Speech synthesis technology are rapidly rising, while the quality of TTS systems is continually improving. We can use text to speech synthesizer in the following applications:

Application for the Blind: The most important application of speech synthesis is for reading and communication aid for visually impaired people. They can not see the length of an input text when listening to it using a voice synthesizer, so a key function is to provide some information about the text to be read ahead of time (Lemmetty, 1999).

Application for Deafened and Vocally handicapped: The synthesized speech gives people with speaking difficulties the ability to speak with people that do not understand sign language (Lemmetty, 1999).

***Application in Education*:** TTS is highly applicable for enhancing language educations. High-quality TTS syntheses are often coupled with a computer-aided learning system, and provide tools to assist a child or a student learn the correct pronunciation of words. The TTS can be programmed with different devices to give a continuous teaching service for students. We can also use TTS for ease of access to relevant information. By making information accessible to people with visual impairments, illiteracy, voice solutions make life easier for thousands of users: the user's voice replaces the screen with natural-sounding speech (Lemmetty, 1999).

***Automotive*:** Speech synthesizers are the quickest and most reliable ways of conveniently communicating with safety, without taking your hands off the wheel or your eyes off the road. It alleviates the problems in vice guidance, traffic information, e-mail reading, onboard alert and diagnostics systems, online reservation, Internet access, and more(Holmes & Holmes, 2002).

Application in telecommunication: TTS has been used in telecommunication to conveying important messages through audio signals from written texts. Synthesized speech has been used for decades in all kind of telephone inquiry systems that allows interactive information retrieval such as caller phone number, account balance, and service information (Lemmetty, 1999).

Application in Multimedia: TTS can be used for language learning, entertainment, talking characters, proofreading, and productivity tools, online talking assistance, talking browsers, and more (Lemmetty, 1999). TTS has been used for years by YouTubers to use the TTS to synthesize speeches from written text and use them in their videos instead of reading by themselves. This makes their video more attractive and saves time.

1.7 Organization of the Thesis

In this section, we provide a summary of topics that are covered in this thesis report. The thesis is organized into seven chapters.

Chapter one, the current section, is an introductory chapter that focuses on the background of the study, the motivation behind the study, statement of the problem with research questions to be addressed, objectives to be achieved, scope and limitations of the study, and application of the study.

Chapter two presents the basic concepts and theoretical background of the speech production system, the challenges in the TTS context, the basic architecture of the text-to-speech system, common speech synthesis methods, basics of Artificial Neural Network and Deep Neural Network, Afaan Oromo morphological analysis, and the works related to our study both from local and foreign languages.

Chapter three discusses the overall methodology of the study including the data collection and preparation, speech feature representations, the model architecture, and the model evaluation strategies.

Chapter four presents the proposed architecture including the training and inference or synthesis phases with their main components. The models used in the proposed model are both at the training and synthesis phase are described in this chapter. In the end, Afaan Oromo morphological analysis is discussed in detail.

Chapter five presents the implementation detail including the tools used, data preparation method, text and audio preprocessing, feature extraction, and model implementations.

Chapter six presents the objective and subjective evaluation of the models and discussion of major results obtained from the experimentation. Finally, chapter seven presents the conclusion and future works.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1 Fundamentals of human Speech Production

The overall anatomical structure of the human speech production process is shown in Figure 2.1. The gross anatomical components of the systems are the lungs, trachea, larynx (organ of speech production), pharyngeal cavity (throat), buccal cavity (mouth), and nasal cavity (nose) (Docio-Fernandez & García Mateo, 2015). The pharyngeal and oral cavities are usually grouped into one unit referred to as the vocal tract, and the nasal cavity is often called the nasal tract (Englewood Cliffs, 1993). The vocal tract extends from the opening of the vocal folds, or glottis, through the pharynx and mouth to the lips. The nasal tract extends from the velum (a trapdoor-like mechanism at the back of the oral cavity) to the nostrils of the nose.

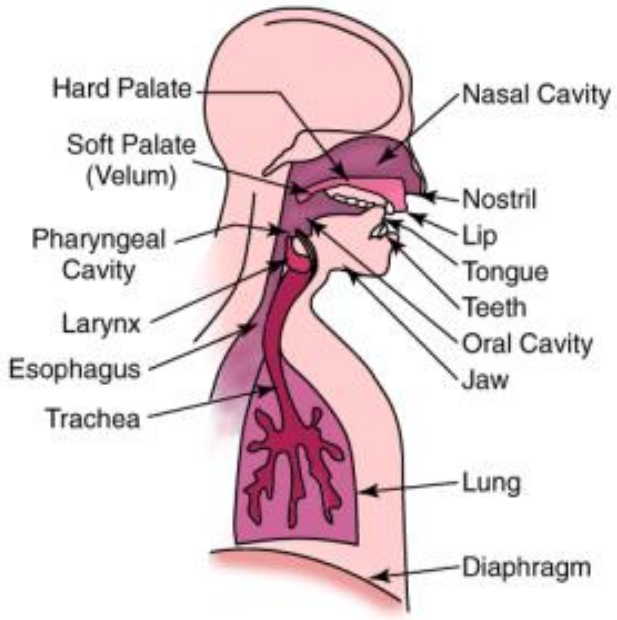

Figure 2.1 The human speech production process (Flanagan, 1971).

The process of human speech production can be summarized as follows. While speaking, the air thrown out from the lungs by muscular force travels into the trachea, then up into the glottis, where it is periodically interrupted by the movement of the vocal cords. The tension from the vocal cords is reshaped by the larynx so that the cords vibrate in an oscillatory fashion, resulting in the production of voiced speech. During the unvoiced speech,

constrictions within the vocal tract (oral cavities – mouth, throat, etc.) force air through the constriction to produce turbulence.

2.1.1 Modeling the Human Speech Production System

Speech is transmitted between humans in the acoustic domain, and fortunately, it can be easily measured and recorded as acoustic representation. The most relevant models of the speech production mechanism belong to acoustic modeling, that is, based on the acoustic theory of speech production (Burrows, 1996). According to this theory, a speech waveform is considered to be the output of a resonant network (namely, the vocal tract filter) that is excited by sound sources placed at the glottis. The main sections of the speech production mechanism, namely, the voice source, vocal tract, and radiation effects, are likely to be linearly modeled in a noncoupled manner following a source-filter arrangement. The assumption that the source and the filter can be separately modeled probably holds for most of the cases. However, this assumption is questionable for low frequencies because the nonlinear coupling may produce damping of the first formant. It is also disputable for unvoiced speech (excitation is due to turbulence originating at constrictions on the vocal tract itself).

As described in the previous section, human speech production can be illustrated by a simple source-filter model adopted from (Docio-Fernandez & García Mateo, 2015) indicated in Figure 2.2. A DC source replaces the lungs, an impulse generator replaces the vocal cords, and a linear filter device replaces the articulation tract. The unvoiced excitement is generated using a noise generator. In practice, all sounds have a mixed excitation, which means that there are voiced and unvoiced elements to the excitation. Of course, the relationship between these parts varies greatly depending on the sound being generated.

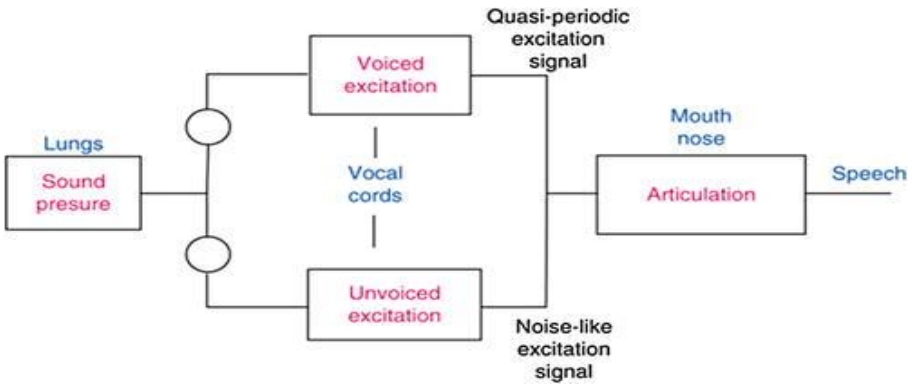

Figure 2.2 A general model for speech production(Docio-Fernandez & García Mateo, 2015)

2.2 Challenges of Text to Speech Synthesis

The problem domain of speech synthesis is a very wide and multidisciplinary research area. There are several problems in text pre-processing such as handling of the Nonstandard Words (NSWs), prosody and pronunciation analysis from the written text is also the main problem nowadays. Written text that contains no straightforward emotions and pronunciation of proper and foreign names is usually irregular. At the low-level synthesis, the contextual effects and discontinuities in wave concatenation methods are the foremost problematic. Speech synthesis is also harder with child and female voices because female voice features a pitch of almost twice and three times as high as male and children voices respectively (Lemmetty, 1999). The major challenges in the text to speech research domain are presented in this section.

Text Preprocessing: text preprocessing is a very complex task and includes several language-dependent problems. All Nonstandard words must be represented in their equivalent phonetic representation (Obius et al., 2003). Digits and numerals must be expanded into full words. For Example in Afaan Oromo, numeral 283 would be expanded as *dhibba lamaafi saddeettamii sadi* and 1995 as *kuma tokkoof dhibba sagaliif sagaltamii shan* (if measure).

Abbreviations must be expanded into full words, pronounced as written or pronounced letter by letter (Macon M. W., 1996). Example ABO can be pronounced letter by letter or expanded form as *Adda Bilisumma Oromo.* In Afaan Oromo, the word order is completely different from that of the English language. Example $100 expanded as *doolaara dhibba tokko*. Equivalent with *"dollar hundred one"* and 100 expanded as *dhibba tokko* equivalent with "*hundred one*" in English. For the above-mentioned reasons, the text processing tool is needed to handle Afaan Oromo texts.

Prosody and emotional content: Another challenging task in this domain is that of identifying correct prosodic features such as intonation, stress, and duration from written text. The intonation means how the pitch pattern or fundamental frequency changes in the speech. The prosodic features of continuous speeches depend on different aspects, such as the semantics of the sentence and the speaker's characteristics and emotions. Unfortunately, the written text usually includes very little information about prosodic features and some of them are changing dynamically while generating speech (Lemmetty, 1999).

2.3 Basic Architecture of Text-to-Speech (TTS) System

The total sum of the idea behind the development of the TTS system is about converting a text string into an acoustic speech waveform. The high-level architecture of a speech synthesizer is shown in Figure 2.3 below. TTS system is generally composed of two components with four sub-components. The first component is a linguistic or *Natural Language Processing (NLP)* front-end consists of Text Processing, Grapheme-to-Phoneme (G2P) Conversion, and Prosody Modeling. The second component is a *Digital Signal Processing (DSP)* back-end (Isewon et al., 2014).

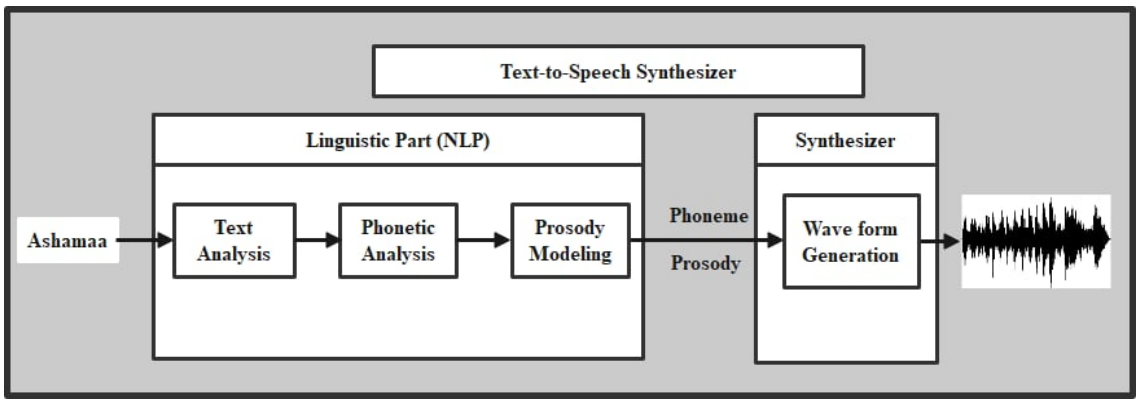

Figure 2.3 Working diagram of a general TTS architecture.

2.3.1 Natural Language Processing (NLP) front-end

The linguistic front-end first performs text normalization or pre-processing to convert numbers, acronyms, and abbreviations into equivalent written-out words. This task is not straightforward since the correct conversion often depends on the context of appearance. After text normalization, morphological and contextual analysis is applied, which categorizes words into different word classes (also called part of speech (POS) or lexical class) to help to predict their pronunciation, which in many languages depends on the POS tag. NLP front-end has three processing stages these are:

Text Processing*:* The basic function of this component is to convert non-orthographic items into speakable words. This is called text normalization from a variety of non-standard words (NSWs) including numbers, currency, abbreviations, and other non-orthographic entities of text into a "common orthographic transcription" (into equivalent Standard Words) suitable for the next phonetic conversion (Thu et al., 2016).

Phonetic Analysis: it is the process of converting lexical orthographic symbols to the phonemic representation of a word. The organization of this module can be organized in

different ways, often roughly classified into dictionary-based and rule-based strategies (Dutoit, 2008). In a dictionary-based approach, a maximum of phonological knowledge is stored in a lexicon in a limited boundary. In a rule-based approach, a set of letters to sound or grapheme to phoneme rules are generated to handle exceptions or unlimited words. Thus, a combination of the lexicon and rules is used in most text-to-speech systems. Primarily, a lexicon is used to store pronunciations, and rules are used as a resort for missing entries in the lexicon.

Prosody Generation: refers to the extraction of supra-segmental features such as intonation, stress, and duration from written text (Lemmetty, 1999). The intonation tries to estimate how the pitch pattern or fundamental frequency changes in the speech. Whereas variation of pitches is observable in everyone's speech, the variations are not explicitly visible in written text. The term prosody refers to certain properties of the speech signal which are related to audible changes in pitch, loudness, and syllable length (Obius et al., 2003).

2.3.2 Digital Signal Processing (DSP) back-end

The task of the DSP back-end is to convert the linguistic feature representation given by the NLP front-end into synthesized speech sounds. In several TTS systems, this part also includes the computation of the target prosody (pitch, intensity, duration, etc.), which is used for synthesis.

Speech generation: is the production of acoustic output from phonetic and prosodic information**.** The preprocessed and analyzed the phonetic representation of phonemes with correct intonation, duration and stress are used to generate speech.

2.4 Speech synthesis methods

In text-to-speech synthesis, synthesized speech can be produced in several different methods. All of these have some advantages and disadvantages that are discussed in this section. These methods are usually classified into three major groups: Rule-based synthesis (Articulatory synthesis and Formant synthesis), Concatenative based synthesis (Unit Selection Synthesis and Diphone synthesis), and Statistical Parametric synthesis (HMM synthesis and DNN synthesis).

2.4.1 Rule-based synthesis

The rule-based synthesis generates artificial speech through the dynamic modification of several speech parameters, such as fundamental frequency, voicing, and so on. The two

major rule-based speech synthesis techniques are articulatory and formant synthesis (A. A. Aliero et al., 2019).

Articulatory synthesis: Articulatory synthesis models the articulatory behavior of humans by direct simulation of the human speech or voice to generate synthesized speech. It generates the most similar and high-quality synthesized speech, similar to the human voice. Articulatory synthesis generates more natural speech but it is one of the most difficult synthesizers to implement. Some of the mechanisms that regulate the articulation include lip aperture, lip protrusion, tongue tip position, tongue tip height, tongue position, and tongue height (Figueiredo et al., 2006).

Formant synthesis: Unlike articulatory synthesis, Formant synthesis synthesizes speech using some instruments that are governed by a set of rules. A good example of a Formant TTS system is the Klatt formant synthesizer (Klatt, 1987).

The generated speech waveform does not use any natural recorded speech, as it is derived from some parameters (fundamental frequency, the amplitude of voicing, open quotient, nasal pole frequency, etc.) (Figueiredo et al., 2006), the source of sound for vowels is a periodic signal with a fundamental frequency and the unvoiced consonants are generated by a random noise generator. For formant synthesis, estimated frequency is used to synthesize speech and this frequency makes the sound distinct models the pole frequencies of the speech signal.

2.4.2 Concatenative Synthesis

Concatenative synthesis was introduced to eradicate the drawback in rule-based synthesis such as the Articulatory and Formant synthesis, which is the difficulty in finding these parameters from the input specification that was created by the text analysis process for generating speech (Rashad et al., 2010). Concatenative synthesis follows a data-driven approach by concatenating or joining together different units of a recorded human speech made available in an existing speech database to generate speech acoustic waveforms.

Diphone synthesis: Diphone synthesis is the most popular method used for creating a synthetic speech. Diphone refers to speech unit from the middle of one phoneme to the middle of the next phoneme. During the runtime, the target prosody of a sentence is superimposed on these minimal units employing digital signal processing such as linear predictive coding, Pitch Synchronous Overlap Add (PSOLA), or MBROLA (Pärssinen, 2007).

Unit Selection Synthesis: A large speech database is used for the Unit Selection synthesis such as the ATR's CHATR (Black, 1994). In Unit Selection Synthesis, the speech database comes in a variety of forms including phones, half-phones, diphones, syllables, words, morphemes, phrases, and sentences. In unit selection synthesis, digital signal processing is rarely used, and for this reason, the naturalness of the speech is very optimal (Isewon et al., 2014).

2.4.3 Statistical Parametric Speech Synthesis

The SPSS based on the Hidden Markov Model (A. Aliero et al., 2020) and Deep Neural Network (Zen et al., 2013) are the state-of-the-art speech synthesis technique in recent times.

SPSS makes use of statistical parameters of speech (spectral and excitation) in the form of an acoustic model. The speech acoustic model was developed from the training process. During the synthesis, these parameters are extracted from the model and converted to speech signals by a vocoder. SPSS produces a fairly natural synthetic speech and flexible voices. SPSS serves as an alternative to overcome the limitation of previous techniques as it only stored parametric representation of sound in speech generation.

HMM Speech Synthesis: HMM-based synthesis is a type of SPSS that uses symbolic parameters generated from the natural language processing unit to generate speech (A. Aliero et al., 2020). HMM-based synthesis has two major advantages over the unit selection synthesis, which are; it is free from audible bugs, and the size of the footprint is very small, unlike the unit selection synthesis that has a very large storage footprint. The smaller size of the storage footprint makes the HMM-based synthesis implementable in handheld devices. HMM-based synthesis has been developed for many languages such as English, Portuguese, Mandarin, Japanese, Swedish, German, Korean, Slovenian, and so on. (Zen et al., 2009).

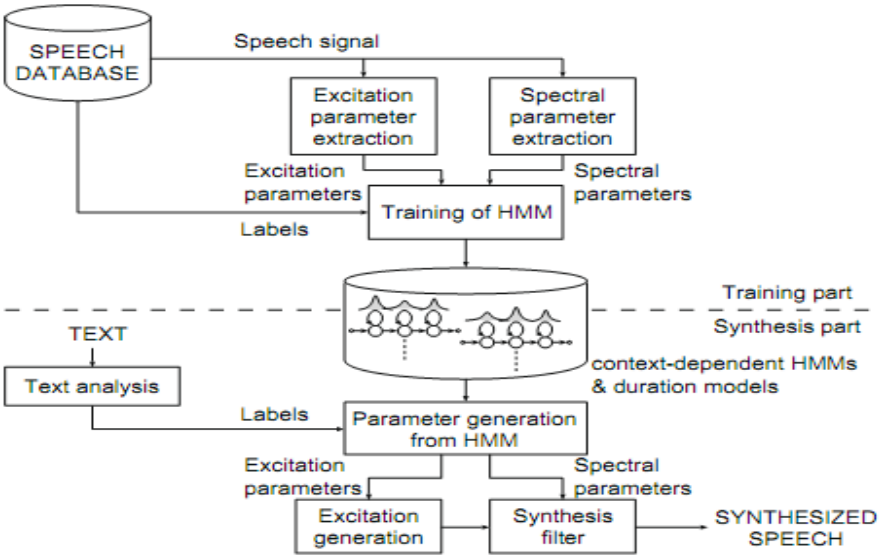

Figure 2.4 The Architecture of HMM-based TTS system (A. Aliero et al., 2020).

As presented in Figure 2.4 HMM-based speech synthesis system comprises two parts, which are training and synthesis. During the training, the recorded human speech parameters (excitation and spectral) together with their phonetic transcription (known as labels) are used to generate the speech acoustic model. During synthesis, the phonetic transcription (labels) of the text to be synthesized was used as the reference to generate the speech waveform from the parameters available in the speech acoustic model.

Deep Neural Network (DNN) based speech Synthesis: DNN-based acoustic models have the potentiality of generating natural-sounding synthesized speech by efficiently offering a distributed representation of complex dependencies between acoustic and linguistic features. DNNs are feed-forward artificial neural networks (ANNs), which have three sets of layers; the input layer, the output layer, and the hidden layer (hidden layer may have more than one set of layers). DNN has achieved remarkable progress in recent years in many machine learning areas including the development of acoustic models for SPSS using speech parameters such as phonetics, syllabic and grammatical ones. DNN-based TTS system produces more natural speech because the training data is presented by the mapping function of the linguistic features (inputs) with the acoustic features (outputs).

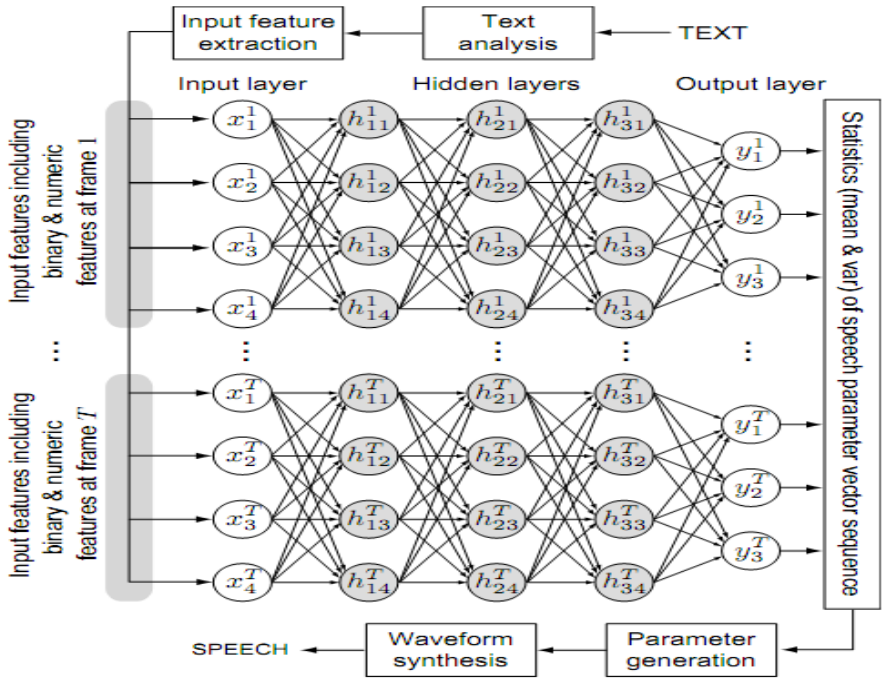

Figure 2.5 a speech synthesis framework based on DNN (Zen et al., 2013).

Figure 2.5 illustrates the DNN-based synthesis framework. The sequence of the input text is first represented by input feature sequence. These input features contain binary answers to questions about linguistic parts such as is-current-phoneme-aa? and the numerical values indicating the number of words in the sentence, the arrangement of the frames in the phoneme, and durations of the phoneme.

The input text feature representations are then associated with output features by a trained DNN model, using forward propagation. The spectral and excitation parameters along with their time derivative features are contained in output features. The input and output pairs of features extracted from training data are used to train the deep neural network model. Similar to the HMM-based approach, it is possible to generate speech parameters by setting the predicted output features from the DNN as mean vectors and pre-computed variances of output features from all training data as covariance matrices. The acoustic feature prediction algorithm can generate smooth orientation of speech acoustic features satisfying the static and dynamic features. Finally, a waveform generation component outputs a synthesized waveform from the given speech parameters.

2.5 Artificial Neural Networks

An Artificial Neural Network is a computing system designed to learn from pairs of data and its associated labels by extracting useful patterns. It is a biologically motivated model that

attempts to simulate the interconnection of neurons that make up a human brain to give the computer the potential to learn things and make decisions in a human-like manner. It extracts patterns from input text data and corresponding speech waveform in the case of TTS systems.

2.5.1 Perceptron

In the context of a neural network, the perceptron is the smallest unit of Artificial Neural Networks that was used in supervised learning to classify the given input data. The perceptron model was designed to incorporate visual inputs, organizing subjects or captions into one of two classes. The perceptron consists of four parts: Input values or one input layer, Weights and Bias, Net sum, and Activation Function (Sagar Sharma, 2017). As shown in Figure 2.6 the inputs $x_i \in \mathbb{R}$, are weighted by the parameters $w_i \in \mathbb{R}$, and summed up. The most frequently used potential v is a weighted sum of inputs plus an additional constant term $w_o \in \mathbb{R}$, called 'bias' and given by:

$$V = wo + \sum_{i=1}^{n} wi * xi \qquad 2.1$$

The potential v finally runs through an activation function f, which is typically non-linear and produces the output y of the perceptron.

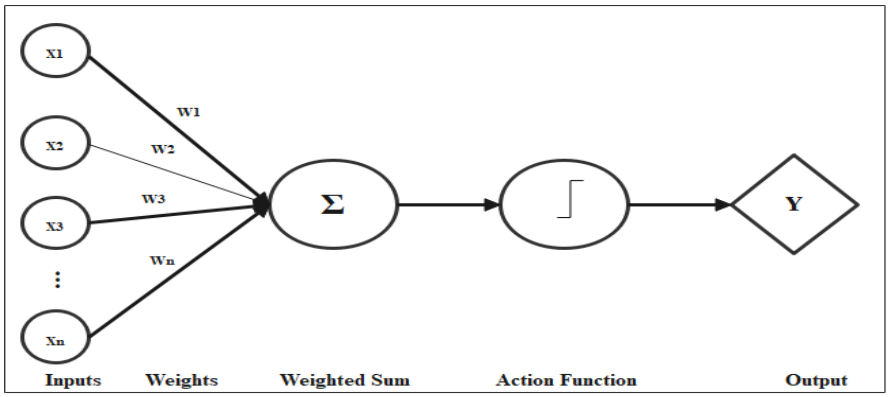

Figure 2.6 Pictorial representation of Single perceptron.

2.5.2 Multi-Layer Perceptron

A perceptron algorithm has a limitation in classification because the architecture is only capable of separating data points with a single line. Unfortunately, XOR inputs are not linearly separable. The solution for this problem is beyond perceptron architecture by using an extra layer of units called the hidden layer. This model is called Multi-layer Perceptron (MLP).

The multi-layer perceptron consists of at least three layers: the input layer, one or more hidden layers, and the output layer. Multi-layer perceptron can have any number of units in its input, hidden, and output layers. The MLP is usually defined through weight matrices $W^{(i)}$ for each layer U_i, which consists of connection weights between layers U_{i-1} and U_i. The computation of the input of any layer U_i can be simplified through:

$$Netui = \Sigma w(i) * Xi \quad 2.2$$

The function *y* of an MLP is defined as follows:

$$y1(x) = f1(w(1) * x)$$
$$yi(x) = fi(w(1) * yi - 1(x)) \quad 2.3$$
$$y(x) = y_r(x),$$

Where:

- *X*i is a pixel vector of the input image of the ith input layer.
- *r* number of layers.
- y_i output of the layer U_i.
- f_i activation function of the layer U_i.

2.5.3 Training of an Artificial Neural Network

To handle the assigned task (can be classification, detection, synthesis, and so on) the ANN should be trained. In closer look training is the algorithmic procedure, which extracts useful patterns or relationships between the input data and given output label. In machine learning, training can be in any of the following three ways: supervised, unsupervised, and reinforcement learning.

Supervised learning: In this method, the model learns an unknown non-linear function that has a finite and known number of numerical values or labels. In such cases, data, which is a set of paired inputs and desired outputs, is given by the supervisor (therefore 'supervised') and the task is to produce the desired output for each input. Classification and regression tasks are the most common example of supervised learning.

Unsupervised learning*:* The aim of unsupervised learning finding previously unknown patterns in a dataset given without labels, thus the network should 'discover' unknown patterns on its own since no 'supervisor' is present. In general, estimation problems such as clustering or self-organization are the most common example of unsupervised learning tasks.

***Reinforcement learning**:* Reinforcement learning is usually performed through interaction with the environment that reacts to actions taken by actors. Reinforcement learning aims to learn the best-possible sequence of actions, whereby the long-term (expected cumulative) cost would be minimized (or reward maximized) in the absence of a training dataset rather through experience.

2.6 Convolutional Neural Network

Convolutional neural networks, or CNNs, are neural networks that are specially designed for processing data that contains a known grid-like topology (Dutoit, 2008). The CNN can process a variety of data such as time-series data, which may be a 1-D grid accepting samples at recurrent time intervals, and the image data, which may be thought of as a 2-D grid of pixels. As its name indicates, it employs a convolutional layer that performs a mathematical operation called a 'convolution'.

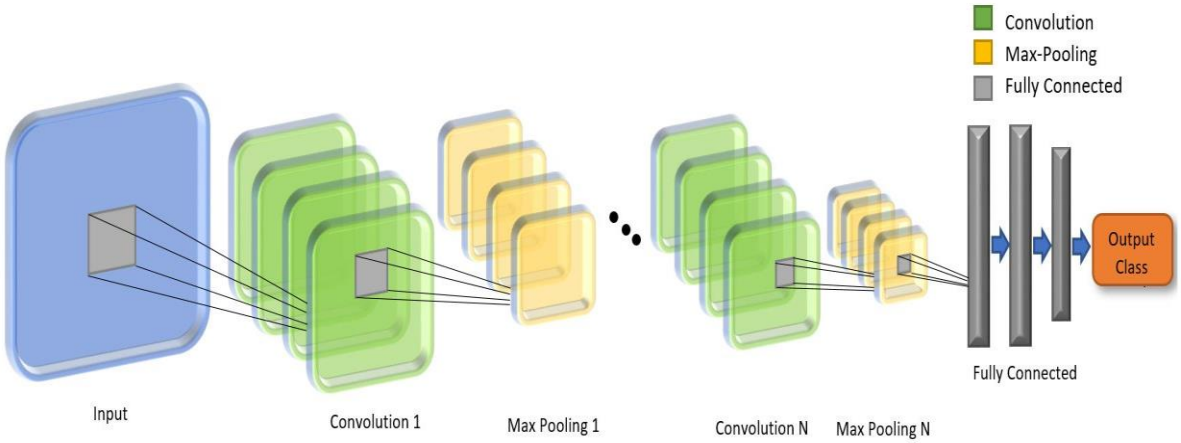

Figure 2.7 General architecture of Convolutional Neural Network (Alam et al., 2020).

The convolution layer: the convolution layer is the main component of a Convolutional Neural Network where a filter/kernel having equal width and height slides over the input with a certain stride (usually 1). This layer produces a feature map resulted from the element-wise matrix multiplication and summing of the result. This result must pass through the ReLU activation function to preserve the non-linearity of the ANN.

Sub-sampling or pooling layer: down sample the output of convolutional layer along both the spatial dimensions of height and width, which lowers both training time and risk of overfitting (Dertat A., 2018). During the pooling process, a pooling window of a specific size slides over the input with a certain stride, operating current values in the window. There are multiple techniques of pooling some of the most common techniques are based on the

operation: max pooling returns the maximum value, average pooling returns the average of all the values in the window.

Fully-connected layers: is a layer that reduces input in a form that is easier to process and that still preserves the original characteristics. The output from convolution and pooling layers are 3D volumes, but a fully connected layer accepts only a 1D vector of numbers. So we convert the 3D output of the final pooling layer to a 1D vector and inputted to the fully connected layer (Dertat A., 2018).

In comparison to 2-D (mainly visual data such as images and videos), it is possible to use CNNs for time-series data such as 1-D strings of characters or words. The key difference is the dimensionality of a kernel and how it slides across the data. The 1-D convolutional filters are widely used in the Natural Language Processing field, especially in the TTS system described in this work.

2.7 Recurrent Neural Network

Recurrent Neural Network is a special type of neural network that is commonly used for processing a piece of sequential information, such as text, audio, and video. In a traditional neural network, we were not considering the dependencies between inputs and outputs. RNNs perform an identical task for every element in the sequence, with the output that is dependent on the previous results (Britz A., 2016). RNNs have a "memory" that captures information about the past calculation.

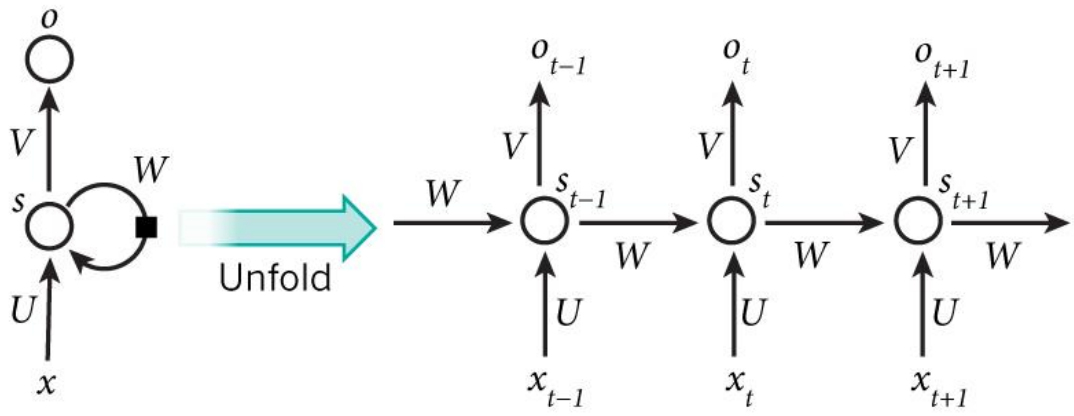

Figure 2.8 A recurrent neural network (Britz A., 2016).

Unidirectional recurrent layer has a memory about the past so at time *t* they have input vector $\boldsymbol{x_t} \in \mathbb{R}^n$ and the memory state at time *t-1* $\boldsymbol{h_{t-1}} \in \mathbb{R}^m$, producing the new memory state $\boldsymbol{h_t}$ also called hidden state, with the following set of operation:

$$\boldsymbol{ht} = f(\boldsymbol{W} * \boldsymbol{xt} + \boldsymbol{U} * \boldsymbol{ht} - 1 + \boldsymbol{b}h) \quad 2.4$$

Where $\boldsymbol{W}$ is the input-to-hidden weights matrix (feed-forward behavior), $\boldsymbol{U}$ is the hidden-to-hidden weights matrix where the feedback is made, $\boldsymbol{b_h}$ is the bias vector and f is a specified element-wise non-linear transformation, such as the hyperbolic tangent or the sigmoid. As we can see, for every input sequence $\mathbf{X = \{x_1, x_2, \ldots, x_T\}}$ we obtain an output sequence $\mathbf{H = \{h_1, h_2, \ldots, h_T\}}$, where each output from the layer keeps track of dynamic changes in time, and this is what makes the recurrent model a really powerful option for sequences (De La Puente, 2016).

2.7.1 Long Short Term Memory (LSTM)

In order to deal with the "long-term dependency" problem (Hochreiter & Schmidhuber, 1997) proposed LSTM that improved the remembering capacity of the standard RNN by introducing a "gate" into the cell. The three different forms of LSTM include: LSTM without a forget gate, LSTM with a forget gate, and LSTM with a peephole connection. By default, the term LSTM cell denotes LSTM with a forget gate. Each LSTM cell has three gates:

The input gate: is responsible for controlling how much of the input from the previous layer is accepted into the memory.

The forget gate: looks at the previous state of the cell and the current input and then regulates how much of the previous memory to keep.

The output gate: controls how much of the current state of the memory is released to the next layer

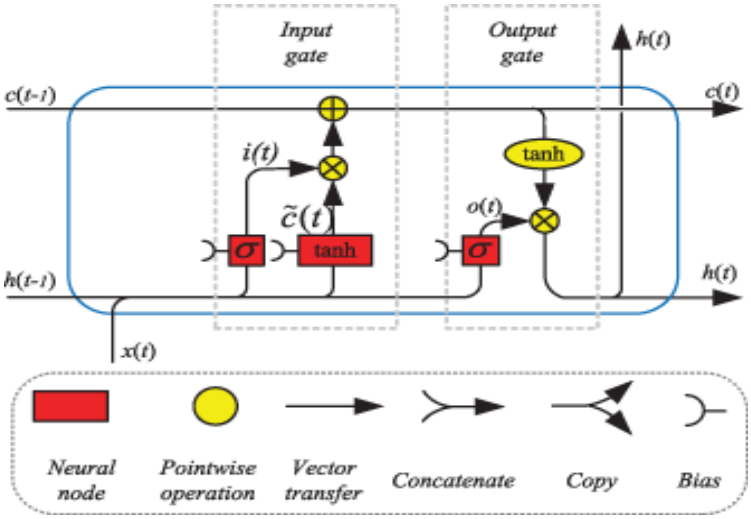

Figure 2.9 Original LSTM architecture (Hochreiter & Schmidhuber, 1997).

2.7.2 Gated Recurrent Unit

LSTM has good learning capacity than that of standard RNN. However, the use of additional parameters increases the computational demands. To deal with this problem (Cho et al., 2014) introduced the gated recurrent unit (GRU) in 2014. Instead of having three gates, the GRU has only two gates these are: the update gate that integrates the forget gate and input gate of the LSTM cell and the reset gate.

2.7.3 Bidirectional LSTMs

Standard RNNs are only able to make use of the previous context for this reason we cannot use them in types of problems that simultaneous access of past and future state is needed. In order to overcome this problem, the Bidirectional RNN (BRNN) that can be trained in both forward and backward time directions simultaneously with separate hidden layers (forward and backward layers) was introduced by (Schuster & Paliwal, 1997).

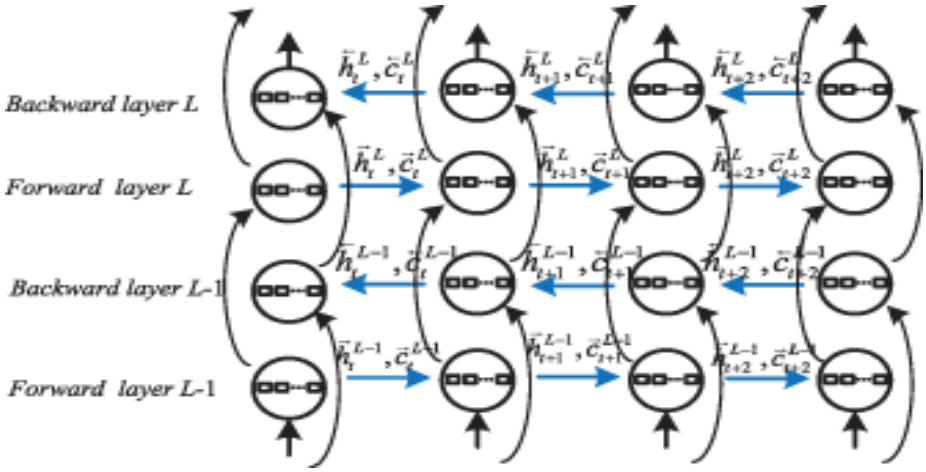

Figure 2.10 Internal connections of bidirectional LSTM (Schuster & Paliwal, 1997).

2.7.4 Sequence-to-Sequence models (Seq2seq)

Sequence to sequence (seq2seq) models is a Recurrent Neural Network architecture that we use to solve complex Language problems such as machine translation, speech recognition, text summarization, and Text-To-Speech Synthesis. Speech synthesis applies the Seq2seq architecture because the number of characters or words in a sentence could be indicative of how long it would take to pronounce it, but it is still difficult to know the exact length. Given an input sequence $\mathbf{(x_1, x_2 \ldots x_S\}}$ of length S a seq2seq model converts into output sequence of $\mathbf{(y_1, y_2, \ldots, y_T)}$ of length T. In most cases, the two sequences are of different lengths ($S \neq T$) (N. Li et al., 2019).

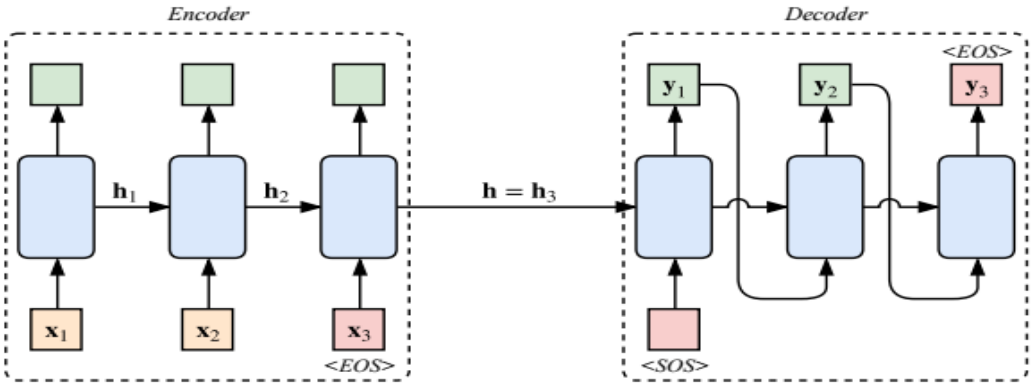

Figure 2.11 a generic sequence-to-sequence architecture (Weweler Y., 2018).

As shown in Figure 2.11, a recurrent encoder compresses a source sequence (x1, x2, x3) into a fixed-dimensional representation h. A recurrent decoder decompresses this representation into a target sequence (y1, y2, y3). Seq2seq models are built from the Encoder-Decoder architecture that contains the encoder and decoder stages.

Encoder-Decoder Architecture

The basic structure of seq2seq model is Encoder-Decoder Architecture where the encoder processes a variable-length input sequence $X = (x_1, x_2 \ldots x_m)$ of m elements and return state representations $Z = (z_1, z_2 \ldots z_m)$, called *encoder vector*. The decoder takes state representation z_m to generate the output sequence $Y = (y_1, y_2 \ldots y_n)$ left to right, one element per time. To compute output y_{i+1}, the decoder calculates a new hidden state h_{i+1} based on the previous state h_i, an embedding of the previous target language word y_i, and the last encoder state z_m (Gehring et al., 2017). Typically, the encoder processes inputs until it encounters an End of sequence (EOS) symbol.

Attention Mechanism

Attention mechanism was introduced to Seq2seq models by (Bahdanau et al., 2015) for machine translation & (Chorowski et al., 2015) for speech recognition allowing higher performance than standard Seq2seq models. The attention mechanism is introduced to address the fixed-length Encoder vector representation bottleneck of encoder-decoder networks. It provides a possibility for a decoder to automatically search parts of the source sequence that are relevant in predicting a target sequence in each step of the output generation (Bahdanau et al., 2015). Models without attention consider only the final encoder state z_m by setting conditional input $c_i = z_m$ for all i or simply initialize the first decoder state

with z_m, in which case c_i is not used (Gehring et al., 2017). Architectures with attention mechanism compute c_i as a weighted sum of $(z_1, z_2 \ldots z_m)$ at each time step.

$$ci = \sum_{j=1}^{m} aij * zj \qquad 2.5$$

Where the α_{ij} (attention score) is interpreted as a measure of how much the output at the time i aligns with the input at time j. It allows the network to focus on different parts of the input sequence as it generates the output sequences. Attention scores are usually computed based on the alignment model a (see Figure 2.12 below), which is parameterized as a simple feed forward neural network and trained with all other components of an encoder-decoder network, and normalized to be a distribution over input elements (Gehring et al., 2017).

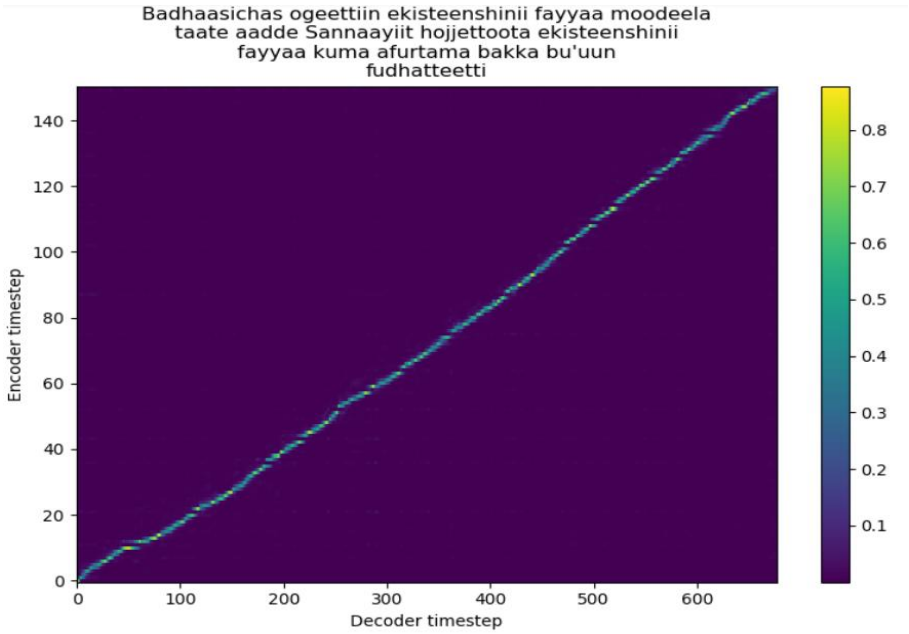

Figure 2.12 Attention alignment plot Example for Text to Speech.

Figure 2.12 shows an example of text-to-speech attention alignment between the encoder steps and the decoder steps for the text at the top. The x-axis shows encoder hidden states, which are attended by the decoder in each decoder time step shown on the y-axis.

2.8 Afaan Oromo Language Phonology

Afaan Oromo (literally Oromo mouth) or the Oromo language, belongs to the Eastern Cushitic branch of the Afro-Asiatic language spoken in Ethiopia, Somali, Sudan, Kenya, and Tanzania (Tilahun Gamta, 1994). Afaan Oromo is probably the third-most widely spoken Afro-Asiatic language in the world, after Arabic and Hausa (Terfa., 1982). According to (Gada Melba, 1988), Afaan Oromo is used as a lingua franca and as a day-to-day means of

communication in whole Ethiopia. Furthermore, it is a language spoken by a large number of people from various ethnonational groups like Harari, Anuak, Berta, Sidama, Afar, and Gurage.

Nowadays, Afaan Oromo is being used as an official language in Oromia Regional State, the biggest region among the current Federal States in Ethiopia. It is used by Oromo people as a mother tongue language, who are the largest ethnic group in Ethiopia, which is close to 50 % of the total population in 2007 (2015 Census statistic of Ethiopia) (Wardbaugh R., 1977).

2.8.1 Afaan Oromoo Writing System

Latin alphabet called Qubee is adopted and used as an official script of Afaan Oromo language replacing the former writing system, Ethiopic Syllabary in 1991 (Tilahun Gamta, 1994). In Afaan Oromo, including the loaned letters (p, ts, v, z, zh) has 34 Qubes (letters). Among these qubes, five of them are vowels (a, e, i, o and u) called ***'Dubbachiiftuu'*** whereas twenty-nine of them are consonants called ***'Dubbifamaa'***. Consonants include b, c, ch, d, dh, f, g, h, j, k, l, m, n, ny, p, ph, q, r, s, sh, t, ts, v, w, x, y, z, zh and ʔ. The last letter (ʔ) is called **"*Hudhaa* (')"** which is called glottal stop in English used to separate more than two consecutive vowels because, in Afaan Oromo, more than two consecutive vowels are not allowed to appear. Example: "*Ta'uu"* which means "become".

Afaan Oromo users will have to be taught the following principles in addition to the above 34 *qubes* (Tilahun Gamta, 1994):

1. Two vowels in succession indicate that the vowel is long, for example. bitaa (left)
2. Gemination of a consonant is phonemic in Afaan Oromo, for example. damee (branch), dammee (sweet potato)
3. h is not geminated in Afaan Oromo.
4. Depending on the context the same words can have more than one meaning, for example. nama baadhu (carry people) namaa baadhu (carry for people).
5. There is no diacritic signs in Afaan Oromo, the combined Latin letters dh, ch, ph, ny, and sh are used instead.

2.8.2 Afaan Oromo Phonemes

A phoneme is an indivisible unit of speech (phone) in a given language. Alternatively, it is an abstraction of physical speech sounds. Latin Qubee used in Afaan Oromo along with their phonemes is shown in Table 2.1.

Table 2.1 The Afaan Oromo Qubes phonemes and their graphemes (Omniglot, 2021)

A a	B b	C c	Ch ch	D d	Dh dh	E e	F f	G g
a	ba	ca	cha	da	dha	e	fa	ga
[ɐ]	[b]	[tʃ']	[tʃ]	[d]	[ɗ]	[ɛ]	[f]	[g]
H h	I i	J j	K k	L l	M m	N n	Ny ny	O o
ha	i	ja	ka	la	ma	na	nya	o
[h]	[ɪ]	[dʒ]	[k]	[l]	[m]	[n]	[ɲ]	[ɔ]
P p	Ph ph	Q q	R r	S s	Sh sh	T t	U u	V v
pa	pha	qa	ra	sa	sha	ta	u	va
[p]	[p']	[k']	[r]	[s]	[ʃ]	[t]	[ʊ]	[v]
W w	X x	Y y	Z z	'				
wa	xa	ya	za					
[w]	[t']	[j]	[z]	[ʔ]				
aa	ee	ii	oo	uu				
[ɑ:]	[e:]	[i:]	[o:]	[u:]				

Lengthened and shortened sounds: The lengthened sounds have the same consecutive vowels in a word. Whereas in the shortened sounds successive appearance of more than one vowel of the same type in a word is not allowed. In Afaan Oromo, the lengthened sound is called *"dheeraa (long)"* whereas the shortened sound is called *"gabaabaa (short)"*. Table 2.2 shows the shortened and lengthened sounds with examples:

Table 2.2 Examples of shortening and lengthening vowels.

Grapheme(vowel)	***Shortened vowel***			***Lengthened vowel***		
	Phonemes	Word	Meaning	Phonemes	Word	Meaning
a	[ɐ]	Hara	Lake	[ɑ:]	Haaraa	New
e	[ɛ]			[e:]		
i	[ɪ]	Bira	Near	[i:]	Biiraa	Beer
o	[ɔ]	Bona		[o:]	Boonaa	
u	[ʊ]			[u:]		

Geminated *or* ***ungeminated:*** Gemination occurs in Afaan Oromo when two consonants appear consecutively in a word. The geminated sound is called *"jabaa (strong)"*. Whereas ungeminated sound is called *"laafaa (weak)"*. According to (Mengasha Riqitu, 1993) gemination of digraphs like ch, dh, ny, ph, sha, ts and zh is marked by doubling the first letter of the components of the digraph. Example "qopphaa'uu" meaning "be prepared". Table 2.3 below shows the germinated and ungeminated consonants with examples.

Table 2.3 Examples of geminated and ungeminated consonants

Grapheme(consonants)	***Ungeminated consonants***			***Geminated consonants***		
	Phonemes	Word	Meaning	Phonemes	Word	Meaning
t	[t]	bitaa	left	[t:]	bittaa	To buy
l	[l]	balaa	accident	[l:]	ballaa	blind
r	[r]	Bira	Near	[r:]	Birraa	autumn

2.8.3 Rules in Afaan Oromo phoneme Distribution

In Afaan Oromo, many restrictions govern the distribution of phonemes known as phonetic rules these are (Kebede, 2014):

1. No more than two consecutive consonants are allowed.
2. In Afaan Oromo, consonant clusters and gemination are not allowed at both the beginning and end of words.
3. Consonant clustering and germination at the middle of words are allowed.
4. Diphthongs of vowels are not allowed in a word.
5. More than two vowels are not allowed in Afaan Oromo unless used with the glottal stop character (hudhaa).

2.9 Morphological process

The modification of basis and affixes in the morphological process are known as morphophonemic change and their study is called morphophonemics (Winfred Philipp Lehmann, 1972).

Assimilation: consonant clusters across morpheme boundaries arise only between the final consonant of a stem and /n/, /t/, or /s/, or between /n/ and a consonant initial stem (Alan S. Kaye and Peter T. Daniels, 1997). Assimilation occurs when consonant clusters appear. Table 2.4 summarizes some of the occurrences of assimilation in Afaan Oromo.

Table 2.4 Assimilation process in Afaan Oromo words

Type	***Description***	***Instance***
S-Labialization	/s/ becomes /f/ when it is followed by a consonant	ajjees + na →ajjeefna- we kill ciis + ta →ciifta- do you sleep
Nasal total assimilation	/n/ completely assimilates to a preceding liquid, and to a following liquid or glide	gal + na → galla- we enter moor(a) + ni → moorri- fat hin + latu → hillatu- not giving
Voice assimilation between stops	/t/ assimilates in voicing to a preceding stop	did + ta → didda- you refuse qab + ta → qabda- do you have
Stop consonant assimilation	Assimilation between a stop and a following **/t/** or **/n/**	fid + na → finna- we bring bit + na → bilma- we buy

Deletion: Sounds can also show a process by which one of the sounds deletes the other. For example in Western Oromo /dh/ is systematically dropped before a consonant (Alan S. Kaye and Peter T. Daniels, 1997).

Example:

fuudh + ta → fuuta – "will you take it"

fedh = na → feena – "we wish"

fuudh + sis + a → fuusisa –" I cause to take"

Epenthesis: Epenthesis is the insertion of a sound in pronunciation (Wardbaugh R., 1977). In Afaan Oromo, the appearance of a sequence of more than two non-syllabic segments is phonetically impossible in the language. In order to deal with this rule, /i/ is inserted between the second and third consonants (Alan S. Kaye and Peter T. Daniels, 1997).

Example:

elm + ta → elmita – "you milk"

fayy + na → fayyina – "we are well"

Metathesis: A phonological process in which, for a variety of reasons, letters, sounds, and even syllables within a word are transposed (Wardbaugh R., 1977). In Afaan Oromo, the process of metathesis takes place when as a consequence of root and suffix combination, an impermissible sequence of segments would occur.

Example:

fr → rf

afur +afa → afuraffaa → afraffa→ arfaffaa - "fourth"

2.10 Afaan Oromo Word Structure and boundaries

2.10.1 Sentence Structure

The sentence structure in Afaan Oromo is not the same as that of English. Afaan Oromo uses the Subject-verb-object (SVO). SVO is a structure where the subject comes first, the verb in between, and the object at the end (Tesema & Tamirat, 2017). For example, in Afaan Oromoo sentence *"Biqilaan bilisa bahe"*. "*Biqilaa*" is a subject, "*bilisa*" is an object and "*bahe*" is a verb. When translated into English, *"Biqila has got freedom"* which is SVO form.

The way adjectives are formed in Afaan Oromo is also different from English. In Afaan Oromo, *adjectives* follow *noun* or *pronoun* and their normal position is close to the noun they modify while in English adjectives usually precede the noun. For example, *"durba bareedduu (beautiful girl)", bareedduu* (adj.) follows *durba* (noun*).*

2.10.2 Word Segmentation

The word (in Afaan Oromo "jecha") is the smallest unit of language (Tesema & Tamirat, 2017). The way we separate words from each other in a sentence might vary from language to language. In some languages, the written text does not have a white space character between words. However, in most languages that use Latin script such as Afaan Oromo, white space characters are used to separate words (Tesema & Tamirat, 2017). Since Afaan Oromo also use the Latin script as alphabet white space character " " is used to separate word from each other. For instance, *"Biqilaan Finfinnee deeme"*. In this sentence, the word "*Biqilaa*", "*Finfinnee*" and "*deeme*" are separated from each other by a white space character.

2.10.3 Punctuation Marks

The punctuation marks used in Afaan Oromo same as that of English and used for the same purpose except for the apostrophe (**'**) which is used to show possession in English and used to represent a Glitch (*hudhaa*) in Afaan Oromo (Tesema & Tamirat, 2017). Like in the English language, the following are some of the most commonly used characters in Afaan Oromo:

- "**Tuqaa**", Full stop (.): is used at the end of a sentence and also in abbreviations.
- "**Mallattoo Gaafii**", Question mark (?): is used in an interrogative or at the end of a direct question.
- "**Rajeffannoo**", Exclamation mark (!): is used at the end of command and exclamatory sentences.
- "**Qoodduu**", Comma (,): is used to separate listing in a sentence or to separate the elements in a series.
- "**Tuqlamee**", Colon (:): is used to separate and introduce lists, clauses, and quotations, along with several conventional uses.

2.11 Related works

In this section different works related to text to speech synthesis, both from local and foreign researchers are discussed. Since most of the foreign researchers work presented in this section are based on deep neural network, they are essential for our work. The local works discussed in this section are based on the conventional TTS methods such as concatenative and Hidden Markov Model (HMM).

2.11.1 Speech synthesis systems for foreign languages using DNN

Recently Deep neural network architectures have proved outstanding performance at learning the inherent features of data. WaveNet (Aaron van den Oord et al., 2016) is a neural-based audio generative model based on the PixelCNN (Aäron van den Oord et al., 2016). It has the capability of producing audio that is very close to a human voice in terms of naturalness. Since it requires linguistic features from another TTS system component that predicts features, WaveNet is not a full end-to-end system and because of the auto-regressive nature of the architecture, it generates speech very slowly.

With the development of end-to-end architectures such as Tacotron (Wang et al., 2017), much of the laborious works to synthesize speech such as feature engineering and human annotation (except the preparation of text, audio pairs for training) are alleviated. Tacotron is an integrated end-to-end generative text-to-speech model that uses a seq-to-seq model with an attention mechanism (Sutskever et al., 2014) to generate speech from the text. It synthesizes speeches frame by frame which makes it faster than the autoregressive method which is sample-level synthesis. Tacotron includes an encoder, content-based attention, decoder (Bengio et al., 2015), and a post-processing network. Tacotron achieves a subjective

mean opinion score (MOS) of 3.82 on US English, hence outperforming a working parametric system in terms of naturalness.

Tacotron 2 (Shen et al., 2018) is a natural advancement of Tacotron that incorporates the best features of Tacotron and WaveNet. It is made from two components these are a recurrent sequence to sequence feature prediction network that converts character embedding to mel scale spectrograms and a modified WaveNet model that acts as a vocoder to generate the synthesized speech signal. Tacotron 2 offers a unified completely neural network technique and removes the non-neural network components employed by Tacotron, such as the Griffin-Lim signal reconstruction algorithm. Tacotron 2 uses hybrid attention (Chorowski et al., 2015) that includes both the location-based attention and the content-based attention and achieves state-of-the-art sound quality close to that of natural human speech. The Tacotron 2 has achieved a mean opinion score (MOS) of 4.53 which is very close to a MOS of 4.58 for professionally recorded speech.

Deep voice 1 (Arik et al., 2017) is another example of deep neural architectures. It has provided high performance, production-level quality, and real-time synthesis. It is made up of five major building blocks these are a grapheme-to-phoneme conversion (G2P) model, a phoneme segmentation model, a phoneme duration prediction, a frequency prediction model, and a signal generation model using a variant of WaveNet using fewer parameters. Deep Voice lays the groundwork for genuinely end-to-end speech synthesis.

Deep Voice 2 (Arık & Miller, 2017) is a modified version of Deep Voice 1 with speaker embedding. It separates the duration model and frequency models where the predicted phoneme durations are provided as inputs to the frequency model. It is a convolutional-recurrent architecture that applies the connectionist temporal classification (CTC) loss for classifying phoneme pairs with the addition of batch normalization and residual connections in the convolutional layers. The initial hidden states of the recurrent neural network (RNN) are produced using speaker embedding.

Deep Voice 3 (Ping et al., 2018) is a fully convolutional attention-based sequence-to-sequence model that is capable of converting textual features such as characters, phonemes, and stresses into vocoder parameters. The audio waveform synthesis model takes the predicted vocoder parameters as an input to generate the audio signals. The Deep Voice 3 architecture is made up of three major components: Encoder which is a fully-convolutional network that converts textual features into an internal learned feature representation,

Decoder which is a fully convolutional that decodes the learned representations in an autoregressive manner and Converter a fully-convolutional post-processing network that predicts the final vocoder parameters. Results are comparable to Tacotron but DeepVoice 3 is faster in training.

(Mishev et al., 2020) presented the first open-source Macedonian language synthesizer that is based on the Deep Learning approach. The dataset of 10,433 short audio clips and approximately 20 hours of speech at a sample rate of 22.5 kHz was created by a native Macedonian female speaker, using a professional microphone that will eliminate most of the noise in the background. For this work, Deep Voice 3 architecture which is fully convolutional that allows a fully parallel computation and training time much faster than at the RNN architectures has been used. The total training using a batch size of 16 was completed after 620K steps after 21 days and 16 h. at a sample rate of 22.5 kHz. The model produced an intelligible, understandable, and partially human-like speech after 50 K steps. The model achieved a MOS of 3.93 which is appropriate for applications that need a text-to-speech synthesis in the Macedonian language.

(Sisamaki Eirini, 2019) implemented an end-to-end neural-based TTS system for the Greek Language using the neural network architecture of Tacotron-2. A dataset created for speech recognition called elenet dataset has been obtained from ILSP (Institute for Language and Speech Processing) and used for this work as well. The dataset contains 3.0 hours of speech, 13 male and female speakers, sampled at 16000Hz, 2154 utterances.

(Fahmy et al., 2020) described the implementation of Tacotron 2 architecture to generate intermediate feature representation from Arabic diacritic text. The researchers used a pre-trained English model with a total of 2.41 hours of recorded speech to apply a transfer learning technique. Modified WaveNet of Tacotron 2 has been replaced by a flow-based implementation of WaveGlow as a vocoder to synthesize speech. It also shows the viability of how to apply transfer learning from English text-to-speech to Arabic text-to-speech successfully even though the two languages are quite different in terms of character level embedding and language phonemes. It also describes how to preprocess audio speech training data to gain a plausible generated speech. The model achieved a MOS of 4.21.

(Näslund, 2018) developed Artificial Neural Networks based speech synthesizer for Swedish. The implementation is based on a modified version of the Tacotron 2 where the WaveNet vocoder is replaced by the simpler Griffin-Lim (Griffin & Lim, 1983) algorithm

used in Tacotron. The dataset used comes from the National Library of Norway and is specifically recorded with speech synthesis creation in mind. A male voice reads 5000 sentences in Swedish. The dataset was recorded between 1997 and 2003 and was intended for use with primarily concatenative synthesis methods.

2.11.2 Speech Synthesis Systems for Afaan Oromo

In this section, the text-to-speech works are carried out by the local researchers for the local language. The summary of these works has been presented in section 2.11.3 below.

Morka Mekonnen (Morka, 2001) developed the first Afaan Oromo TTS system by using the concatenative speech synthesis technique using diphones as the basic concatenation units to synthesize sample Afaan Oromo words. The study mentioned the consideration and inaccessibility of the Hidden Markov Toolkit and Center for Spoken Language Understanding tools for speech synthesis. As indicated in the research paper success in recognizing the utterance of the transcribed phonetic unit was 43.33% for native speakers and 83.33% for listeners who heard the utterance at least three times in a different way. The author finally recommended the incorporation of smoothening techniques like linear predictive coding (LPC) to smooth the transition points of the diphones to improve on the errors that arise from segmentation.

The second author, Samson Tadesse (Samson, 2011) has developed a concatenative-based text-to-speech synthesizer for Afaan Oromo. In the study, the number of rule-based diphone database entries used was too small. According to the finding, 75% and 54% of words in the data set are correctly pronounced as to the diphone and triphone speech units respectively. The concatenation of large units degraded the performance of the system. The author achieved the intelligibility of 3.03 and 2.2 rates for the diphone and triphones respectively and the naturalness of the system was 2.65 and 2.02 for each speech unit respectively. However, due to the use of a rule-based approach, the overall performance of the system becomes poor. The concatenative approach requires a larger database to include all possible utterances in the language.

The most recent study for Afaan Oromo is by Muhidin Kedir (Wosho, 2021). The researcher has developed an HMM-based statistical parametric speech synthesis model for Afaan Oromo. The author obtained the result 4.1 and 4.3 out of 5 scores in terms of naturalness and intelligibility respectively. Non-standard words such as abbreviated words, numbers,

currency, acronyms, and punctuation marks are not considered in this work. Recommended applying the deep learning methods or deep neural network methods (DNN).

2.11.3 Summary of Afaan Oromo Related Works

The simplified summary of the text-to-speech synthesis works that are closely related to our proposed work has been presented in Table 2.5. We have studied the techniques used by prior researchers with the performance evaluation results. In addition, we have identified the gaps that were not incorporated by the previous studies to incorporate in our proposed solution as much as possible.

As described above, there were attempts to develop a speech synthesizer for Afaan Oromo. However, their attempts are based on conventional methods. Even though the requirement of the large database is solved by the use of HMM technique it requires a laborious feature engineering process and deep domain expertise. In addition, in the HMM technique linguistic features need to first be mapped into probability densities, for this reason, it was difficult to incorporate the complex linguistic features. (Wosho, 2021) didn't consider NSWs such as abbreviated words, numbers, currency, acronym, and punctuation marks and used a very small speech dataset.

Table 2.5 Summary of Afaan Oromo Related Works

No	***Author***	***Techniques***	***Evaluation***	***Gaps***
1	(Morka, 2001)	Concatenative (diphone)	*Recognition:* 43.3% for native speakers and 83.3% for listeners who heard three times	- Not used Smoothening techniques like linear predictive coding (LPC). - No smooth transitions at concatenation point.
2	(Samson, 2011)	Concatenative (diphone and triphone)	*Intelligibility:* 3.03 (diphone) and 2.2 (triphone) and *Naturalness:* 2.65 (diphone) and 2.02(triphone) out of five scale.	-Non-standard words were not included. - Glottal stop that we can find in the Afaan Oromo language (called Hudhaa) is not included. - Concatenation of large units degraded the performance of the system. - It requires a larger database to include all possible utterance
3	(Wosho, 2021)	HMM techniques	*Naturalness:* 4.1 and *Intelligibility:* 4.3 out of 5 score.	- Non-standard words were not considered. - Decision tree is inefficient to represent complex dependencies between linguistic & acoustic features. - HMM rely on strong assumptions that may not always be true.

2.12 Summary of the Chapter

The chapter first introduced the introductory concepts about speech production and the modeling of the human speech production system. The structure of anatomical components of the human speech production system with their important contribution has been presented. Apart from this, the application of acoustic speech production theory in modeling the human speech production process has been described in the introductory part of this chapter.

The other concepts covered in this chapter are different speech synthesis methods employed for various languages including rule-based synthesis, concatenative synthesis, and statistical parametric synthesis. The basic features such as types of data used, quality of the synthesized speech, size memory needed to store the data, advantages, and disadvantages they have concerning one another are covered.

The basic concepts about Deep Neural Networks and different DNN based problem-solving techniques with their common application areas have also been presented in this section. CNN, RNN, LSTM, GRU Encoder-Decoder Architectures, and attention mechanisms are presented in detail because they are the basis for this study.

The last section of this chapter presented the review of different related works from both local and international researchers that are closely related to the study. The techniques used by these researchers, the final result of their model as well as their scopes have been presented in this section. Finally, the gaps found in their works have been critically reviewed since these gaps are used as input for the current study. As seen in this section even though there are several works of literature for well-resourced languages like English and Mandarin, the development of text to speech model for Afaan Oromo is still a challenging task. So far three works have been done for Afaan Oromo but their attempts are based on the conventional methods that require a large database, laborious feature engineering, and deep domain expertise. Because of the nature of techniques used for previous works, NSWs such as abbreviated words, numbers, currency, acronyms, and punctuation marks are not yet solved for Afaan Oromo.

CHAPTER THREE

3. RESEARCH METHODOLOGY

In this chapter, the procedures and techniques followed to achieve the research objectives are proposed. The first section of this chapter presents data collection and preparation methods used in this thesis. The second section presents the speech feature representation including the low-level feature that represents the audio signal and how they are used in the proposed model. The third section of this chapter presented the detail of models, their structure, and organization used in this study. From the list of most popular TTS models convolutional neural network (CNN) based model called Deep voice 3 and recurrent neural network (RNN) based model called Tacotron 2 has been presented in detail. The last part of the chapter is concerned with the model evaluation techniques used in this research work.

The detailed methodology workflow of the proposed model is presented in Figure 3.1.

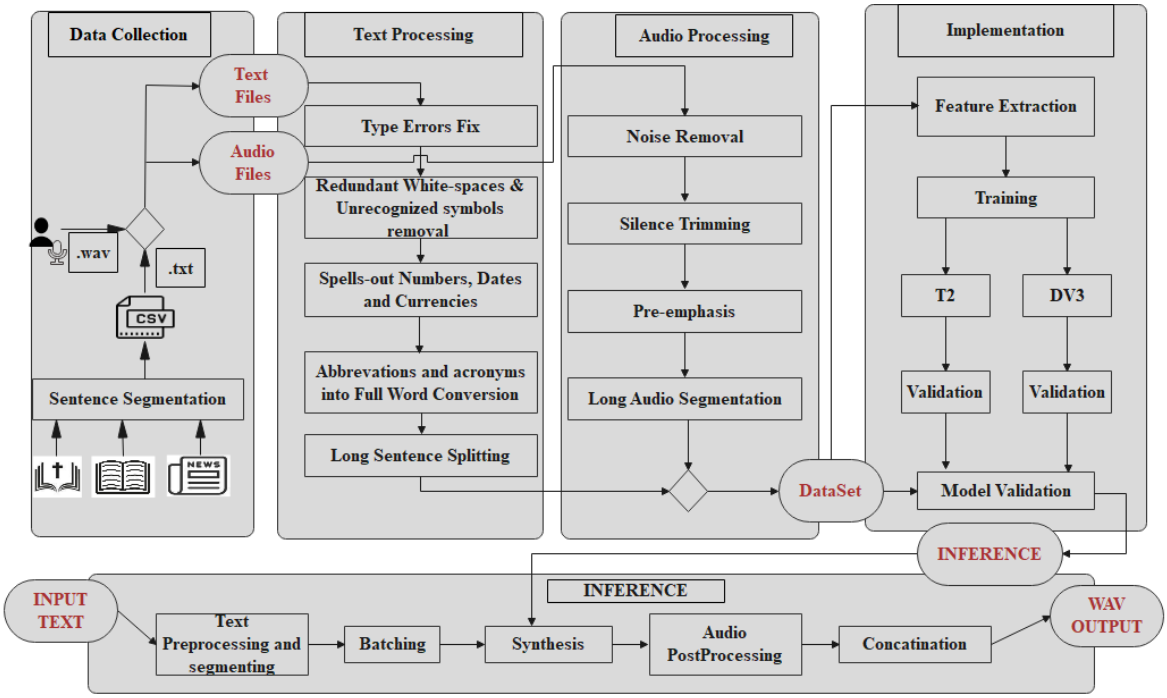

Figure 3.1 Afaan Oromo TTS Methodology

3.1 Data Collection and Preparation

Speech corpus is one of the fundamental requirements for any text-to-speech synthesis research works. Speech corpus is a collection of sentence-level text and corresponding audio records of the text in <text, audio> format. Because of the annotation cost, there is no open-

sourced and large-scale speech synthesis corpus in Afaan Oromo. Because of the unavailability of speech datasets, it is needed to build a speech dataset for this thesis work.

3.1.1 Text Data Collection

For successful training of the TTS system, a large natural human speech dataset is needed. Most researchers follow the common speech dataset collection and preparation guidelines of the LJ Speech Dataset[4], which is considered as a golden standard for testing the accuracy of the DNN-based synthesizers for the English language.

In this work, in order to build a phonetically rich and balanced Afaan Oromo speech corpus a big raw text was crawled from a variety of sources. The Afaan Oromo speech corpus collected for this work is considered as a phonetically rich and balanced corpus, due to its variety in source and its large size. The use of a variety of sentence types, sentence structure, sentence lengths, etc. allows the corpus to incorporate a vast range of domains, hence improve the quality of the TTS system. Therefore, the three major sources used for building a raw text for this work were:

News Media sources: the news agencies that are used as a text source for this study are: Fana Broadcasting Corporate, Ethiopian News Agency, and BBC Afaan Oromo. These organizations were chosen as text sources because of three reasons these are:

- ❖ They have regular programs in Afaan Oromo.
- ❖ They are reliable and easily available sources
- ❖ The texts are free of grammatical errors.

Generally, from these sources, a total of 3,288 sentences are collected and preprocessed for the audio recording process.

Non-fiction Books: these sources are used because of their availability in electronic form as well as they are written by language experts. Two non-fiction books titled: '*Ida'amu'* and *'Coqorsa Abdii'* are used as a text source under this category. For this work, a total of 2,600 sentences are collected from this source.

[4] Recordings by Linda Johnson (https://keithito.com/LJ-Speech-Dataset)

Holy Bible: the third text source used in this work is the *'Afaan Oromo Holy Bible'*. This source provides large and phonetically rich texts in Afaan Oromo. A total of 2,188 sentences are extracted and processed from this source.

Overall a text dataset that contains a total of 8,076 sentences is collected from six sources including three from news services (Fana Afaan Oromo, BBC Afaan Oromo, and ENS Afaan Oromo), two from books (Ida'amu and Coqorsa Abdii), and Afaan Oromo Holy Bible.

3.1.2 Audio Recording Process

Since the proposed model consumes the <text, audio> pair data set, for this reason, the text corpus needs to be converted into corresponding audio form. The audio recording is performed in a professional studio by using a microphone that eliminates background noise. A professional 24 years old male speaker from Oromia Art Institute was invited to read the texts. In order to facilitate the recording processes, a software tool created for this purpose that is presented in Figure 3.2 was used. It automatically serves the speaker with consecutive sentences from the text corpus and allows the one-click creation of audio files.

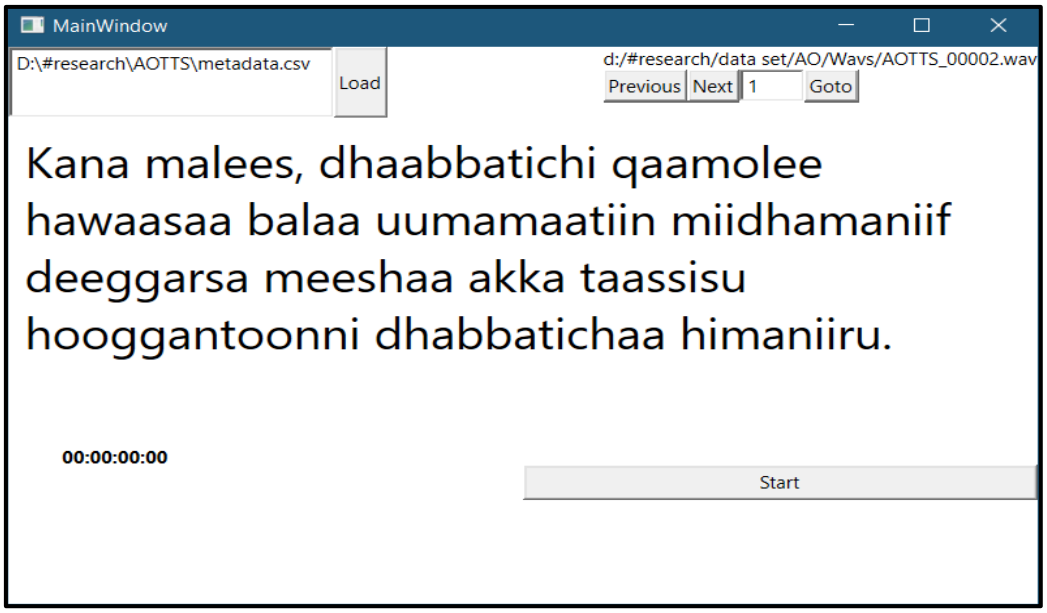

Figure 3.2 Software tool for dataset creation.

As presented in Figure 3.2, the CSV (comma-separated values) file containing the whole text corpus in the format (wav_path | text_sample) is provided to the speaker as an input. The software tool reads the text samples and audio path line by line and the speaker generates a .wav file by clicking the "start" button with the retrieved wave path. As soon as the recording process starts, the timer starts counting and the button is changed to red with a "stop" label. One more click on the button just clicking the "enter" key from the keyboard, and the audio

file is saved in .wav format to the corresponding audio path. Consequently, a new portion of the sample is displayed to be read. To be more user-friendly the software tool provides the manual way of moving backward and forward among the text samples if there is a need to record some samples all over again.

The audio files used in this study have the following characteristics:

- The sampling rate employed is equal to 22050 Hz.
- Bits per sample used is signed 16-bits.
- Saved in WAV file format.
- The audio files are represented in PCM format.
- Each audio signal is recorded using a single audio channel, mono-channel.

After two weeks of the audio recording process, a total of 17 hours of recorded speech data that corresponded to a total of 8076 recorded .wav files was created. The summary of the speech dataset prepared for Afaan Oromo is presented in Figure 3.3. The translated sentence that is displayed in Figure 3.2 is the second sentence from the text corpus whose original form is ***"In addition, the administrators announced that the organization will give material support for the society affected by natural disaster."***

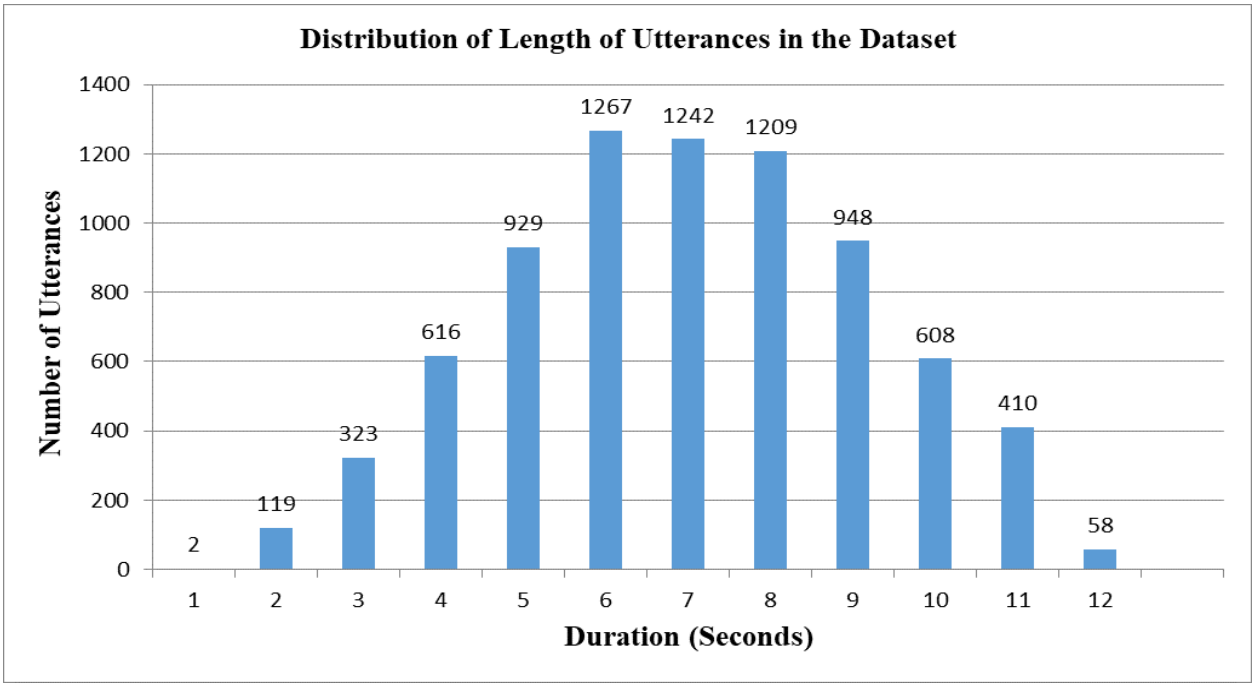

Figure 3.3 Summary of Speech dataset.

3.1.3 Text Pre-processing

Several textual and audio preprocessing steps are applied before the dataset is input into the deep-learning models to improve the quality of the audio and text files. The text processing procedures used in this work are closely related to those used by many other TTS approaches. In this work the following text processing procedures are applied to the text data:

Text Cleaning: the collected text needs to be cleaned before performing the audio recording process. The main cleaning activities performed on our text data are:

- ✓ All kinds of typo errors, unnecessary white spaces, and repeated words are fixed.
- ✓ All characters are converted to lowercase.
- ✓ Period (.) has been added at the end of the sentences that do not have end of sentence markers such as !,.:;?.
- ✓ All characters other than a-z, ␣, and " ’ - , ; : ( ) ! ? are stripped.
- ✓ Very long sentences were segmented.

Normalization: text normalization is a process of generating the normalized word for the text that contains non-standard words. Obviously, Afaan Oromo text contains non-standard words that cannot be directly transcribed to phonemes. In this study, we have performed the normalization processes on the following texts:

- ✓ Sentences containing numbers, years, and currencies were replaced with a full textual representation matching the recordings.[5]
- ✓ Abbreviations and acronyms are written out.[6]

Segmentation: after normalization, the length of the sentence has increased, for this reason, very long sentences which need greater than 25 words to be uttered and short sentences that are less than 3 words were discarded as such sentences can make the training process inefficient.

3.1.4 Audio Pre-processing

Before extracting the acoustic features and starting the model training, the audio signals underwent the following audio processing procedures:

[5] Example 14 replaced by ***“Kudha-Afur”***
[6] WD is expanded as ***“Waaree Dura”***

Normalization: while recording the audio files the volume of the speaker may differ from word to word and as a result, the file can end up unable to be processed clearly. In order to deal with the above problem, the volume of audio files has to been adjusted to a standard set level.

Pre-emphasis: before starting acoustic features are extracted, the audio signals are pre-emphasized. We do this by amlifying the high-frequency components of the audio signal. Pre-emphasis is done to compensate for frequency variation while recording the audio signal since there is more energy at lower frequencies than at the higher frequencies.

Silence trimming: the audio signal has silence at the beginning and the end. The silence at the beginning as well as at the end of the audio samples in the dataset has been trimmed. It proved that trimming facilitates the alignment between the text utterances and audio samples, which decreases the time requirements for training.

3.2 Linguistic Features Representation

The text-to-speech consumes the linguistic features as an input. These features can be in either phoneme-level or character-level. In the phoneme-level feature the conversion of characters to their corresponding phoneme, grapheme to phoneme conversion, is performed. Though it outperforms the character-level embedding, the process is laborious and requires deep linguistic expertise as a result the character-level feature was widely used by TTS synthesis systems for under-resourced languages. For this study, the character-level embedding was used to feed the input text to the acoustic models.

Character embedding: refers to the process of converting each character of a sentence into its numeric representations. Character embedding is usually a high-dimensional vector used to capture the content of a sentence or a single element such as word and character as shown in Figure 3.4. In this study, we have used character embedding with 512 dimensions to represent each character before feeding them into an acoustic model for parameter prediction. The reason why we used character embedding than word2vec embedding is that it can handle infrequent words better and reduces model complexity by using a small number of vectors (Edward M., 2019).

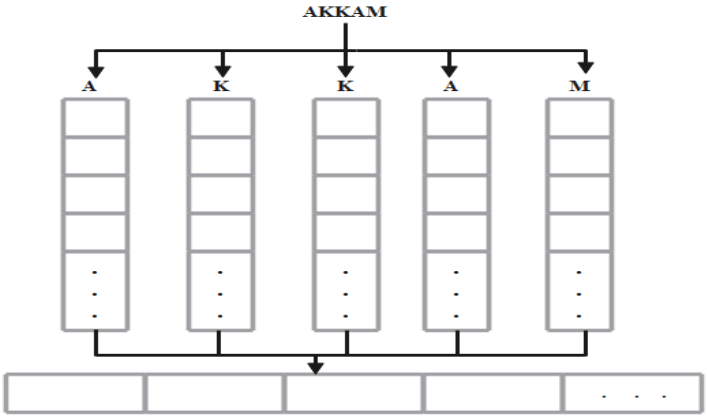

Figure 3.4 Character Embedding

3.3 Acoustic Feature Representation

Acoustic features are the acoustic properties of speech signals for speech analysis. Based on their semantic interpretation audio features are classified into physical and perceptual features. Perceptual features are basic features that are perceived by human listeners such as loudness, pitch, rhythm, and timber. In contrast, physical features are properties that present mathematical, statistical, and physical properties of audio signals.

The most commonly used acoustic features in speech synthesis are the Mel Frequency Cepstral Coefficients (MFCC), Mel-frequency spectra, and Linear Predictive Coefficients (LPC). Acoustic features are often used as a low-level audio representation to bridge the synthesizer and the vocoder in the back-end of a TTS system. In this study, we used Mel and linear scale frequency spectrograms as an intermediate acoustic feature representation. To be specific, the output of the feature predictor and the conditional input of the vocoder are sequences of mel and linear frequency spectrograms.

3.3.1 Linear scale Spectrogram

Many TTS systems have to model speech with some kinds of speech parameters that capture information about the fundamental frequency, formants, and periodicities of speech. These characteristics are found in a short-time Fourier transform (STFT) of a speech signal. An STFT converts the audio signal from its time-domain into its frequency domain and displays the amplitude of each frequency with respect to time function using a Linear scale spectrogram. Linear scale spectrogram is a two-dimensional representation of speech that shows the magnitudes of frequency components in the y-axis and the function of time in the x-axis.

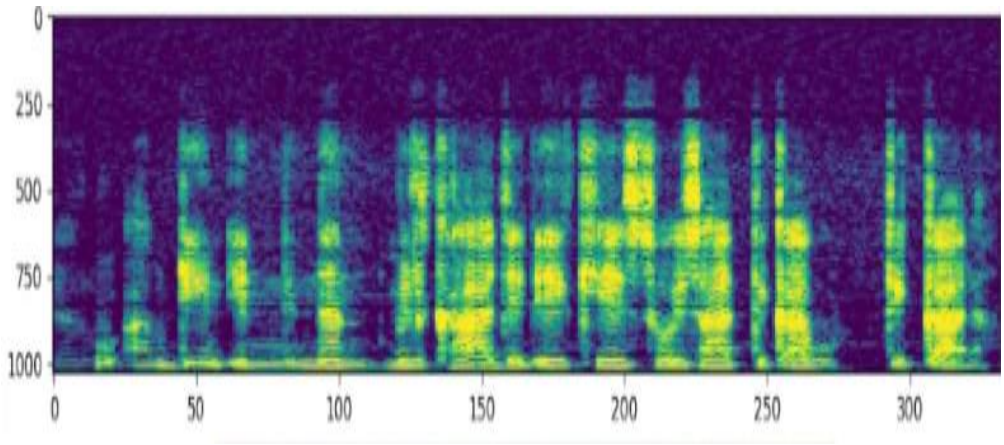

Figure 3.5 Example of a linear scale spectrogram

Figure 3.5 shows an example of a spectrogram where the logarithm of the magnitude of each Fourier transformation is visualized. The meaningful information regarding speech in spectrograms that are needed during decoding tends to be on the lower frequencies and highly redundant. This redundancy makes the model more complex and restricts the process of learning an alignment between the written text and the speech signal. A useful tool that has been shown to improve TTS systems that use spectrograms as speech features is to convert the frequencies into a mel-scale spectrogram (Shen et al., 2018).

3.3.2 Mel scale spectrogram

Mel scale is a non-linear transformation of the frequency scale based on how pitches are perceived by listeners. It is a spectrogram that contains mel scale frequencies that are computed from the linear one by replacing the frequencies on the y-axis with mel scale. Mel scale spectrograms are computing by a short-fast Fourier Transformation (STFT) using a Hann window function and slicing a waveform signal into a window of 50ms frame size and 12.5ms frame hop size. Unlike the linear spectrogram, mel scale spectrogram uses the decibel scale to indicate the frequencies.

Distances between sounds on a mel-scale correspond to how different the sounds are perceived. Figure 3.6 shows the mel-scale spectrogram computed from the linear spectrogram of Figure 3.5. The figure shows how the distances between lower frequencies are greater, and differences between higher frequencies are smaller. When comparing a mel-scale spectrogram to a regular spectrogram, the resolution between the frequencies that matter for speech is higher and there is less redundant information. It emphasizes details in lower frequencies that are critical for speech modeling and de-emphasizes the high frequencies.

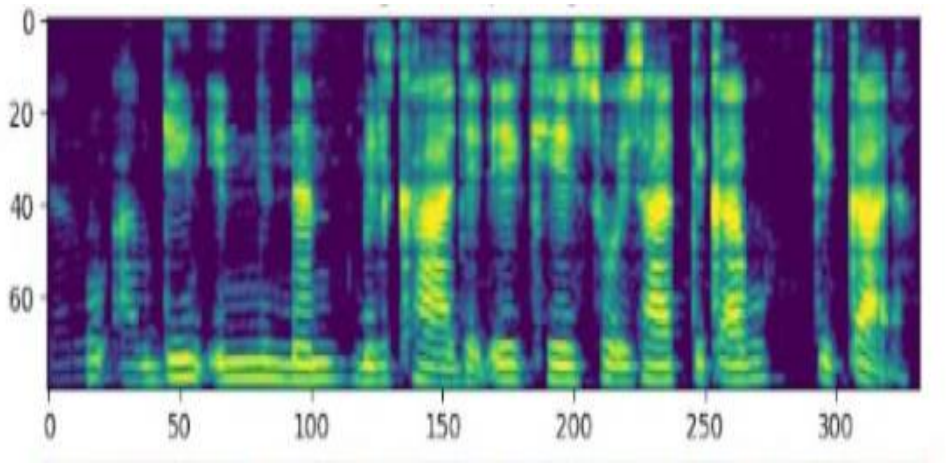

Figure 3.6 Example of a mel-scale spectrogram

3.4 Acoustic Model Architecture

Different neural network-based acoustic modeling techniques have been used for the systems designed in the text-to-speech synthesis domain. Each neural network techniques have both strengths and weaknesses. For this reason, two of the most popular neural-based text-to-speech synthesis models have been chosen to train on our dataset: the RNN-based model called Tacotron 2 and the CNN-based model called Deep Voice 3. Both Tacotron 2 and Deep Voice 3 models have different architecture and designs. Therefore, the initial study of these systems allows a better understanding of their structure, their components, and their compatibility with the way our speech dataset is prepared.

3.4.1 Convolutional Neural Network Model

The first TTS model used in this work is a Convolutional Neural Network based model proposed by (Ping et al., 2018) which is called Deep Voice 3. Deep Voice 3 is a fully convolutional neural network that uses the sequence-to-sequence architecture to maps text into spectrograms or other acoustic parameters to be used with an audio waveform generation method. It outperforms other models in terms of the trade-off between training speed and generated quality speech. It supports the input of the joint representation of characters and phonemes for training.

The Deep Voice 3 architecture consists of three components. The first one is a fully convolutional component called an encoder, which converts linguistic features into an internal learned representation that is fed into the attention-based decoder. The second component is called decoder that is a fully convolutional causal component, which maps the internal learned representation into a low-dimensional audio representation, a spectrogram,

in an autoregressive manner. The third component is a fully convolutional post-processing network called a converter, which computes the final vocoder parameters from the decoder hidden states depending on the target vocoder used.

3.4.2 Recurrent Neural Network Model

The second TTS model used in this work is a Recurrent Neural Network based model proposed by (Shen et al., 2018) which is called Tacotron 2. Tacotron 2 is a recurrent neural network model for synthesizing a speech directly from the text. The Tacotron 2 model contains two major components. The first one is an attention-based recurrent sequence-to-sequence network that maps character embedding to mel-scale spectrograms. The second component is a modified WaveNet, a neural model that synthesizes time-domain waveforms from mel spectrograms. It functions based on the combination of a convolutional neural network and a recurrent neural network.

3.4.3 Griffin-Lim Algorithm

Waveform synthesis is the last stage of the text-to-speech synthesis systems where the predicted acoustic features represented by the intermediate representation, the spectrogram, are converted into time-domain waveforms. The spectrograms are predicted by the acoustic modeling networks discussed in sections 3.4.1 and 3.4.2. In our study, since neural network-based vocoders need a long training time, we used Griffin-Lim Algorithm (GLA) for real-time inference. It does not have trainable weights, simple and faster in convergence while generating the waveforms. Therefore, it is the initial choice for recent text-to-speech synthesis systems.

3.5 Model Evaluation

Once the models are trained properly they need to be evaluated to assess their performance. In this study, the performance of the TTS models trained on our dataset is assessed using both subjective and objective evaluation techniques.

3.5.1 Objective Evaluation

As stated in (Ping et al., 2018) attention-based neural TTS systems may run into three types of errors that can degrade the synthesis quality these are repeated words, mispronunciations, and skipped words. In this study, an objective evaluation has been performed to measure the performance of our models against these problems.

3.5.2 Subjective Evaluation

The subjective evaluation is performed to assess the performance of the model in terms of intelligibility and naturalness. For subjective valuation of the text to speech synthesis model, a mean opinion scale (MOS) listening test was conducted. MOS has been the recommended subjective performance assessment of the synthesized speech quality since 1994 (ITU-T, 1994). MOS is the arithmetic mean of the human raters' opinions on the synthesized speeches that are used to measure intelligibility and naturalness. It contains eleven items five for naturalness and six for intelligibility that were rated on a five-point scale. Participants of the test listened to generated speech utterances and rated them. The points of the scale were *Bad*, *Poor*, *Fair*, *Good,* and *Excellent* which are mapped to numbers 1-5 with 1 increment.

CHAPTER FOUR

4. THE PROPOSED SOLUTION AND LANGUAGE PHONOLOGY

This chapter presents the proposed solution for Afaan Oromo text-to-speech synthesis using a deep neural network (AOTTS). The proposed solution is divide into two phases these are the training and synthesis phase. The details of each of the components that have been used in both training and synthesis phases are discussed in this section. The Afaan Oromo language phonology has also been presented in this section to give relevant details of the language. The Afaan Oromo writing system, Morphological structure, Phoneme formation rules, and usage, and Afaan Oromo word structure and boundaries are discussed in this chapter.

4.1 The Proposed Solution

In this section, the proposed architecture of Afaan Oromo text to speech synthesis using a deep neural network (AOTTS) has been described. Our architecture has a training and synthesis or inference phase. The overall architectural design of the proposed solution is depicted in Figure 4.1 below.

The proposed solution is divided into two phases: the training phase and the synthesis phase. In the training phase, linguistic features parameters are extracted from the input texts and acoustic features parameters are extracted from the input audio signals, and then the DNN based acoustic model was trained based on linguistic and acoustic features. In the synthesis phase, the trained acoustic model predicts acoustic features parameters from the given linguistic feature (the given unseen text), and then the vocoder produces the synthesized audio signal from the predicted acoustic feature parameters.

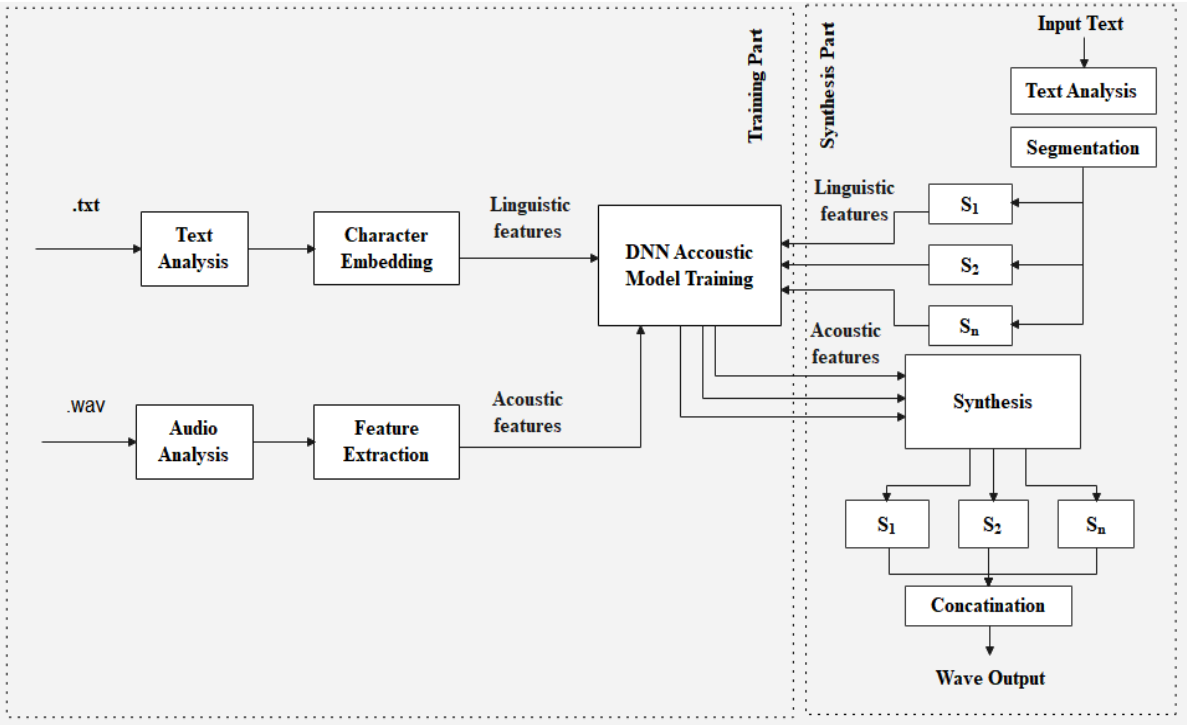

Figure 4.1 The block diagram of Afaan Oromo TTS.

4.2 Model Training Phase

In the training phase, we trained both RNN based Tacotron 2 and CNN-based Deep Voice 3 parameter prediction models in advance. Before feed into the acoustic model for training, the input text and audio signals need to be analyzed in order to get the best out of it.

Text Analysis: the proposed model has a text analysis module used at inference time that is specifically created for this work since the text analysis frontend is a language-dependent component. The text input has been analyzed before using it for training. The detailed text pre-processing procedures have been presented in section 3.1.3.

Character embedding: refers to the process of converting each character of a sentence into its numeric representations. Character embedding is usually a high-dimensional vector used to capture the content of a sentence or a single element such as word and character. In this study, we have used character embedding to represent each character before feeding them into an acoustic model for parameter prediction. The reason why we used character embedding than word2vec embedding is that it can handle infrequent words better and reduces model complexity by using a small number of vectors (Edward M., 2019).

Audio Analysis: as explained in section 3.1.4, the audio files are passed through audio pre-processing and feature extraction procedure. In our proposed model, the acoustic feature

parameters of the audio signals have been extracted and represented by the intermediate feature representation. We used linear spectrogram and Mel spectrogram as an intermediate acoustic feature representation.

4.3 Acoustic Models

This section presents the architecture of the DNN-based TTS models used in this study. As explained in section 3.4, we have used Tacotron 2 and Deep Voice 3 acoustic parameter prediction networks in our proposed system.

4.3.1 CNN based Model

The CNN-based acoustic feature prediction model used in this work is called Deep Voice 3. Deep Voice 3 proposed by Baidu's Silicon Artificial Intelligence Lab is an attention-based sequence to sequence model. It is a fully convolutional model that can convert linguistic features into acoustic features. The functional block diagram of Deep Voice 3 is presented in Figure 4.2.

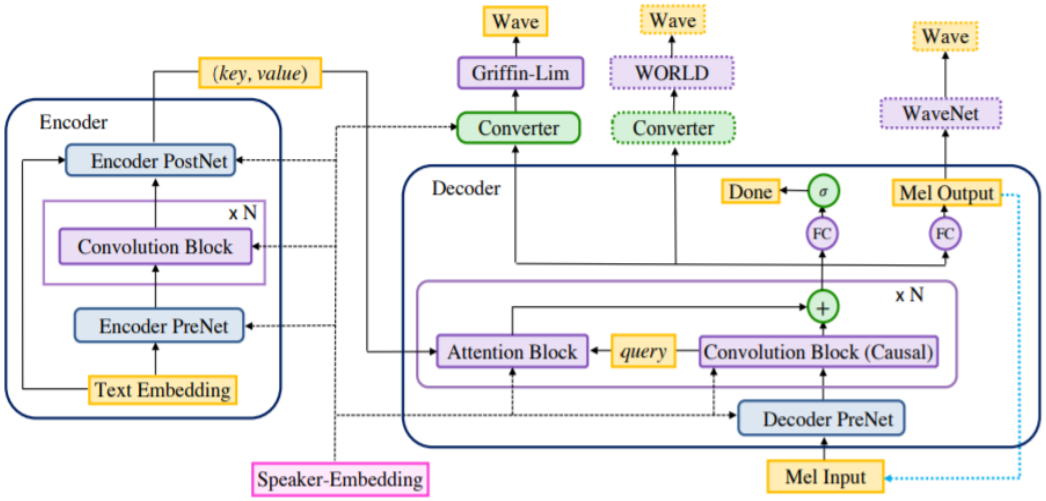

Figure 4.2 Deep Voice 3 architecture adopted from (Ping et al., 2018)

Encoder network: starts by first converting the textual input (i.e. characters or phonemes) into trainable vector representations, h_e, through embedding layer. In order to extract time-dependent textual information, the embedding dimensions that are first projected via a fully connected layer to a target dimensionality are again going through a stack of convolution blocks. Finally, they are projected back to the embedding dimension to create the attention key vectors h_k are created. The attention value vectors are created from attention key vectors and text embeddings, $h_v = \sqrt{0.5}(h_k + h_e)$. Each attention block uses the key vectors h_k to

compute attention weights, whereas the final context vector is computed as a weighted average over the value vectors h_v.

Decoder network: the decoder is naturally autoregressive. It produces audio by predicting a group of *r* futures audio frames conditioned on the past audio frames in an autoregressive way. Mel spectrogram is chosen as a compact low-dimensional audio frame representation.

The decoder network starts with preprocessing the input mel-spectrograms using multiple fully connected layers with rectified linear unit (ReLU) nonlinearities. The convolution blocks generate the queries used to attend the encoder's hidden states over the attention block. A fully connected layer produces the next group of *r* audio frames and also a binary "final frame" prediction that indicates whether the last frame of the utterance has been synthesized. Dropout is applied to each fully connected layer that precedes the attention blocks, except for the first one. An L1 loss and a binary cross-entropy loss are computed using the output mel-spectrograms and the final-frame prediction respectively.

Attention Block: Deep voice 3 used the dot-product attention mechanism. The dot-product attention uses the encoder key-value (key, value) vectors and the decoder hidden states (a query) as input to compute attention weights and then outputs a context vector calculated as the weighted average of the value vectors.

Converter network: first takes the decoder activations from the last hidden layer as inputs to predicts vocoder parameters to be used by vocoders applying several non-autoregressive and non-causal convolution layers. Several vocoders can be used with described architecture such as Griffin-Lim (Griffin & Lim, 1983), WORLD (Morise et al., 2016), or WaveNet vocoders.

4.3.2 RNN based Model

The RNN based model used in our study is called Tacotron 2. Tacotron 2 is proposed by researchers from the Google research center. Its architecture contains an attention-based recurrent sequence-to-sequence network that predicts the mel-scale spectrograms from character embeddings, and a vocoder to generate waveforms from predicted mel-scale spectrograms. The functional block diagram of the Tacotron 2 model is shown in Figure 4.3.

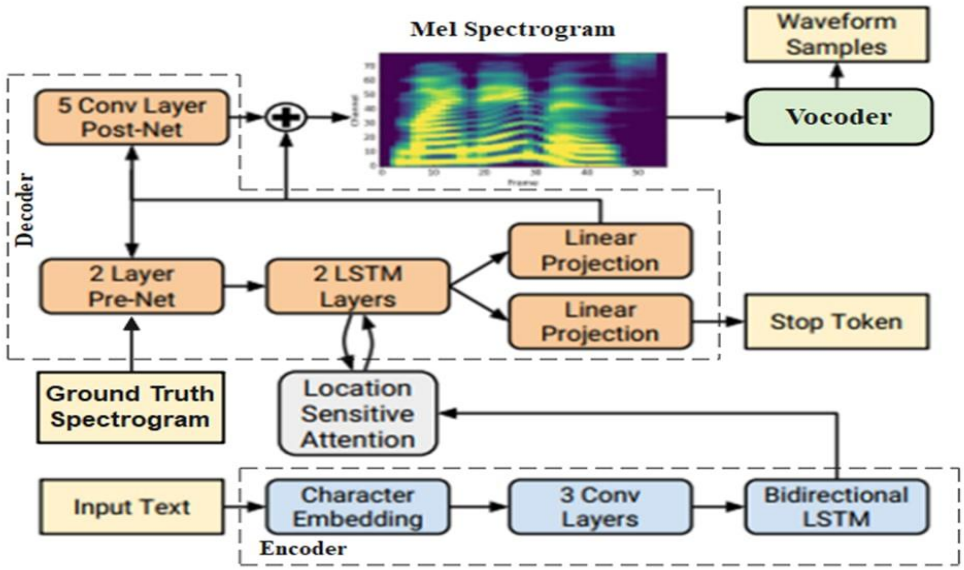

Figure 4.3 Block diagram of the Tacotron 2 Model.

Spectrogram Prediction Network: as presented in Figure 4.3, the spectrogram prediction network is a sequence to sequence architecture that contains an encoder and decoder components with attention. All audio files are converted to 80 channel mel scale spectrograms through a short-time Fourier Transform (STFT) using a 50 ms frame size 12 5 ms frame hop and a Hann window function. The encoder is visualized with blue blocks and the decoder with orange blocks. The encoder is made up of three parts: character embedding, three one-dimensional convolution layers, and a bidirectional LSTM. As an input, the encoder takes a text sequence and generates a hidden feature vector representation that acts as a multi-dimensional representation of the character. The coordinates of the embedding vectors are among the trainable parameters of the model, and during training, the randomly initialized embedding vectors start adjusting themselves.

The following block in the architecture is an embedding layer (512- dimensional vector) which represents each character symbol numerically. The output from the embedding layer is inputted to a stack of three convolutional layers, each contains 512 filters with a 5x1 dimension to span five characters and model long-term contexts (N-gram). Following each filter, there are batch normalization and a ReLU (J. Brownlee, 2021). Tensors produced by the convolutional blocks are fed to a bi-directional LSTM of 512 units (256 in each direction). Forward and backward results are concatenated to generate encoded features to be supplied to the decoder.

The decoder architecture mostly follows a recurrent neural network's structure, it autoregressively creates one timeframe of the mel-spectrogram at a time and feeds the frame

back as an input for the next time step. The decoder consists of a stack of two LSTM layers, two-layer pre-net, post-net of five convolutional layers, and linear progression. It consumes encoder hidden representation of feature vector and predicts mel-spectrograms of the given input sequence of characters.

The predicted mel-spectrogram is then passed through a pre-net that contains 2 layers of 256 hidden ReLU units. The mel-spectrogram from the pre- net layer is concatenated with the context vector of the attention block and passed through a block of 2 uni-directional LSTM layers of 1024 units. The output of the LSTM layer and the context vector of the attention block are concatenated and projected through a linear transform to estimate the target spectrogram frame. Finally, the target mel spectrogram went through a post-net of 5 convolutional layers which predicts a residual to add to the prediction to improve the overall reconstruction. The post-net is a component that is made up of 512 filters with shape 5×1 with batch normalization, followed by tanh activations on all but the final layer.

The spectrogram prediction network uses location-sensitive attention (Chorowski et al., 2015) which extends the additive attention mechanism from (Bahdanau et al., 2015) by using cumulative attention weights from all previous decoder time steps. The attention layer ensures that the decoder focuses on the right parts of the decoder output. More specifically, the attention layer summarizes the entire encoder output, calculates attention weights for each decoder time step, and creates a context vector, which is used as input for the LSTM layers.

4.4 Synthesis Phase

The second part of any text-to-speech synthesizer is the synthesis or inference phase. In the synthesis phase, our input text underwent text analysis and segmentation before being sent to the acoustic feature prediction model. In the text analysis phase, our new sentence was sent to the text preprocessing frontend which is written for handling Afaan Oromo texts. The frontend module applies preprocessing activities such as text cleaning, text normalization, tokenization, and number to word conversion. In the segmentation phase, the long sentences are segmented to prevent inferences with a duration of over 10 seconds. Each segment is batched to preserve the order. The acoustic feature prediction model predicts each of them and the vocoder converts them to a waveform. The segments are concatenated together and presented as a single .wav file during the concatenation phase.

The predicted acoustic features of the input text need to be converted into an audio signal. This conversion is carried out by a device called a vocoder, a contraction of voice and encoder. A vocoder is an algorithm that estimates raw audio waveforms from its representation by predicting the vocoder parameters. The representation is usually the mel-spectrogram or spectrogram derived from modified Short Time Fourier Transform (STFT). To be specific in our work, we have used Griffin-Lim and WaveNet as vocoders.

4.4.1 Griffin-Lim Algorithm

For the experimental purpose, an algorithm that provides results efficiently and quickly is required. The Griffin-Lim algorithm is a waveform generation method based on the use redundancy of the short-time Fourier transform. The goal is, to convert spectrograms to time-domain audio waveforms by estimating the unknown phases iteratively.

The algorithm is iterative, so depending on the size of the spectrogram size of iteration is set. A large number of Griffin-Lim iteration slow down the inference. In our work, we have used a GLA to predict the audio waveform from the spectrogram predicted by the acoustic feature prediction models. The GLA pseudo code is presented in Figure 4.4.

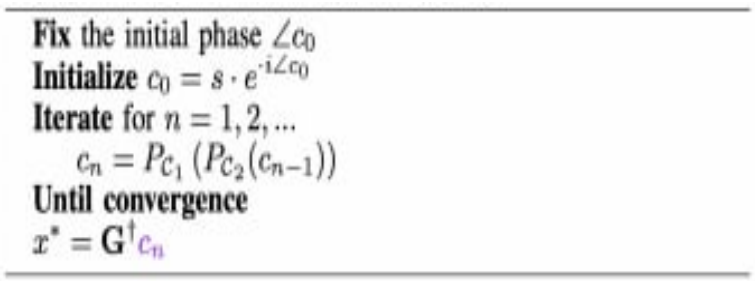

Figure 4.4 GLA pseudo code

In this study, we have raised the predicted weights of the spectrogram by a power of 1.5 before feeding to the Griffin-Lim algorithm to reduce the artifacts in the constructed signals (Wang et al., 2017). Though 30 iterations are typically enough, we have implemented the Griffin-Lim algorithm for 60 iterations to ensure convergence.

CHAPTER FIVE

5. Implementation of Afaan Oromo Text-to-Speech Model

This section describes the implementation details including the tools used for implementation, data preparation, data preprocessing, feature extraction, and model training. We have used different tools both software and hardware tools are used to facilitate our work. These tools are categorized into data collection and preprocessing tools, package and environment management tools, modeling tools, and hardware tools. The implementation of data preparation, preprocessing, and feature extraction have been carried out on our dataset before feeding to the model to be trained. At the end of this chapter, the training of the acoustic feature prediction models has been conducted and the training progress of each model has been presented.

5.1 Tools used

5.1.1 Data Collection and Pre-processing tools

Audacity[7] ***2.4.2:*** audacity is an audio editor and recording software tool that is free and open-source. It is a cross-platform and easy-to-use software. We used audacity in the data collection process for removing long silences and noise in the audio signal.

Speaker App: speaker App is a window-based software tool developed to facilitate the audio recording process by providing the text data that is read by the speaker. It provides a single sentence at a time and saves the audio file automatically with the specified file name. This tool also contains a start and end button for starting and stopping the recording process.

Microsoft Excel 2013: Microsoft Excel is used in collecting tasks such as normalization of the non-standard word, segmentation of long sentences, removing duplicates, fixing typo errors, aligning the sentences with audio files in <*wav file* |*un-normalized text*| *normalized text*> format, and saving our dataset in CSV file format.

SciPy[8]***:*** SciPy is a Python-based library that the scientific and technical computing tasks. It is free and open-source. It contains packages that are used in optimization, integration, interpolation, linear algebra, Fast Fourier transform (FFT), signal and image processing, and other tasks common in science and engineering. We used two of SciPy modules in this study, the first one is the signal processing module for synthesizing and modifying the signal before

[7] www.audacityteam.org
[8] https://www.scipy.org

processing and the second one is the Fast Fourier transform module for converting a signal from its original time domain to a representation in the mel spectrogram and vice versa.

NumPy[9] ***1.16.0:*** NumPy is a python library used for processing large, multi-dimensional arrays and matrices by providing a large collection of mathematical functions to manipulate on these arrays. In this study, the NumPy library has been used for handling the audio files with its corresponding feature representations; linear spectrogram, and mel spectrogram.

Librosa[10] ***0.7.2:*** Librosa is a powerful python package dedicated to music and audio analysis. We used librosa for audio processing tasks such as retrieving the audio, visualizing the audio signals, and extracting audio features using different signal processing techniques.

Matplotlib[11] ***2.0.2:*** Matplotlib is a comprehensive plotting Python library used to visualize the different statistical aspects. In this study, we used matplotlib for visualization such as plotting the encoder-decoder alignment, presenting the training history, and drawing the target and predicted mel spectrogram of the audio file.

Natural Language Tool Kit[12] ***(NLTK):*** The NLTK is a collection that contains python modules, datasets, and tutorials. Though NLTK is written for English some modules are shared with the Afaan Oromo language such as Punkt word tokenizer for marking sentence boundary. Punkt tokenizer knows that the period in Mr. Smith and Johann S. don does not mark sentence boundaries. It also learns parameters such as a list of abbreviations from the portion of text.

CUDA: CUDA is a parallel computing platform and application programming interface (API) model created by Nvidia. It increases the computing performance by allowing software developers and software engineers to use a CUDA-enabled graphical processing unit for general purpose processing.

5.1.2 Package Manager and Environments

Colab: Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody having a google account to code and execute any python code using a browser. It is a well-known programming environment in data analysis and machine learning. It is a

[9] https://numpy.org/
[10] https://librosa.org/doc/latest/index.html
[11] https://matplotlib.org
[12] https://www.nltk.org

cloud Jupyter notebook that provides a virtual programming environment without a need for any setup, providing free access to high computing resources such as GPUs and TPUs.

Anaconda Navigator: Anaconda is a distribution of the Python programming language that aims to remove complexities from package and environment management. The distribution also includes packages for data-science tasks suitable for all platforms. It provides GUI based interface for the management of packages and environments

5.1.3 Modeling Tools

Python 3.6: Python is a widely used high-level general-purpose programming language that is concise and easy to read. The endless packages and libraries it provides make it a popular programming language in the area of data science.

Tensor flow 1.15.2: It is a widely-used open source library for the development and training of advanced machine learning models such as a deep neural network. It is based on Tensors which are multidimensional arrays and provides high performance and computational abilities. TensorFlow can run on both CPUs and GPUs and has recently emerged on more powerful TPU platforms.

Keras 2.2.4: Keras is a high-level neural network API that offers simple and consistent APIs minimizing the number of user actions required. We can use Keras alongside TensorFlow and Theano. It is a powerful and free open-source library written by python for the development and evaluation of deep neural networks.

Pytorch 0.22: PyTorch is a Torch-based, Python machine learning library. The torch is a Lua-based computing framework, scripting language, and machine learning library.

The NVIDIA CUDA Deep Neural Network library (cuDNN): cuDNN is a library for deep neural networks that rely on high-performance GPU acceleration. It allows deep learning researchers to focus on training and developing software applications by providing them a high-performing computational resource.

5.1.4 Hardware Tools

Condenser Microphone: for successful training of the deep neural network, high-quality data input is mandatory. Similarly, for the success of our model, we need high-quality audio signals. In this study, a high-quality condenser microphone with USB output has been used for audio recording purposes in a professional studio.

Personal Computer: we have used a personal computer with Intel® processor Core™ i5-6200U CPU @2.40GHz, 8 Gigabytes of physical memory, 1 Terabyte hard disk storage capacities, GPU: NVIDIA GeForce GTX 950M with dedicated 4GB of RAM, and 64 bits Windows 10 operating system.

The graphics processing unit (GPU): GPU is a specialized processor with dedicated memory and processor that can dramatically speed up computational processes for deep learning. GPU can be faster at completing tasks than CPU but not for every case. In this study, because of the size of the dataset used it was difficult to carry out the training process only on CPU that is the point where the need for GPU arises. We have used NVIDIA Tesla T4 GPU that supports up to 16GB of memory and 2560 CUDA cores to accelerate a wide range of modern applications.

5.2 Data Preparation

5.2.1 Loading the dataset

Before carrying out any processing on our dataset, the dataset should be loaded from disk to the python environment. The dataset which contain *(wav-path | un-normalized-sentence | normalized-sentence*) format is loaded as a CSV file from *input_dir*. For preprocessing purpose we need the *wav-path* and *normalized-sentence* columns which represents the location of the recorded audio file and the corresponding normalized sentence respectively.

```
executor = ProcessPoolExecutor(max_workers=n_jobs)
futures = []
index = 1
for input_dir in input_dirs:
  with open(os.path.join(input_dir, 'metadata.csv'), encoding='cp1252') as f:
    for line in f:
      parts = line.strip().split('|')
      basename = parts[0]
      wav_path = os.path.join(input_dir, 'wavs', '{}.wav'.format(basename))
      text = parts[2]
      futures.append(executor.submit(partial(_process_utterance, mel_dir, linear_dir, wav_dir, basename, wav_path, text, hparams)))
      index += 1

return [future.result() for future in tqdm(futures) if future.result() is not None]
```

Figure 5.1 Implementation of dataset loader

5.2.2 Loading the Audio Files

The loaded audio files need to be preprocessed before feeding to the model for training. We have preprocessed each instance in audio/text pair by loading the audio files as NumPy array from disk storage. The following code snippet has been used to load the .wav files from the

storage location, *wav_path* by using librosa library applying the *sample_rate* on every audio signal.

```
import librosa
import librosa.filters
import numpy as np
import tensorflow as tf
from scipy import signal
from scipy.io import wavfile


def load_wav(path, sr):
  return librosa.core.load(path, sr=sr)[0]
```

```
import numpy as np
from datasets import audio

  try:
    wav = audio.load_wav(wav_path, sr=hparams.sample_rate)
  except FileNotFoundError: #catch missing wav exception
    print('file {} present in csv metadata is not present in wav folder. skipping!'.format(
      wav_path))
    return None
```

Figure 5.2 Implementation of audio file loader

5.3 Text Preprocess

5.3.1 Text cleaning

For training purposes, the text data has been cleaned and inputted into the model. At the inference time or synthesis time, we use new texts that are not yet preprocessed and contains unnecessary elements. For this reason, we have to repeat the preprocessing tasks performed before training. We have performed preprocessing activities such as: removing unnecessary white spaces, converting into lowercase, adding virtual end of sentence character if not set, expanding numbers and abbreviations. The code which performs text cleaning is depicted in Figure 5.3 below.

```
def ao_cleaners(text):
    #Pipeline for Afaan Oromo text, including number and abbreviation expansion.
    text = add_punctuation(text)
    text = lowercase(text)
    text = expand_numbers(text)
    text = expand_abbreviations(text)
    text = collapse_whitespace(text)
    return text
```

Figure 5.3 Afaan Oromo Cleaner

5.3.2 Expanding Abbreviations

In this work, since abbreviations and acronyms are not pronounced as they are written, we have written a code that replaces the abbreviations and acronyms in their expanded form. The python code that performs this normalization with sample Afaan Oromo abbreviations is presented in Figure 5.4 below.

```
_abbreviations = [(re.compile('\\b%s\\.' % x[0], re.IGNORECASE), x[1]) for x in
[
    ('dr', 'dooktar'),
    ('Inj', 'Injiiner'),
    ('jen', 'jeneraal'),
    ('prof', 'proofeesar'),
    ('Aadd', 'Aadde'),
    ('kkf', 'Kan kana fakkaatan'),
    ('wkf', 'Waan kana fakkaatan'),
    ('Fkn', 'Fakkeenyaaf'),
    ('Hub', 'Hubachiisaa'),
    ('Obb', 'Obbo'),
    ('A.L.I', 'Akka Lakkoofsa Itiyoophiyaa'),
    ('A.L.A', 'Akka Lakkoofsa Awurooppaa'),
    ('M/Murti', 'Mana Murtii'),
    ('WD', 'Waaree Dura'),
    ('WB', 'Waaree Booda'),
]]

def expand_abbreviations(text):
    for regex, replacement in _abbreviations:
        text = re.sub(regex, replacement, text)
    return text
```

Figure 5.4 Common Afaan Oromo abbreviations

5.3.3 Normalization

Another preprocessing activity is the normalization of numbers, currencies, ordinals, decimals, and comma-separated numbers. We have performed this task using the following function which substitutes each attribute with its normalized form.

```
def normalize_numbers(text):
    text = re.sub(_comma_number_re, _remove_commas, text)
    text = re.sub(_pounds_re, _expand_pounds, text)
    text = re.sub(_euros_re, _expand_euros, text)
    text = re.sub(_dollars_re, _expand_dollars, text)
    text = re.sub(_decimal_number_re, _expand_decimal_point, text)
    text = re.sub(_ordinal_re, _expand_ordinal, text)
    text = re.sub(_number_re, _expand_number, text)
    #text = re.sub(_phone_re, _expand_phone_number, text)
    return text
```

Figure 5.5 Afaan Oromo Text Normalization

5.3.4 Num2Word

The following three figures show the num2word conversion of Afaan Oromo texts during synthesis time. This section presents the implementation of some of the num2word conversion used in this study for Afaan Oromo numbers, ordinals, decimals, and currencies.

```
def _expand_dollars(m):
  match = m.group(1)
  parts = match.split('.')
  dollars = int(parts[0]) if parts[0] else 0
  cents = int(parts[1]) if len(parts) > 1 and parts[1] else 0
  if dollars and cents:
    dollar_unit = 'doolaara'
    cent_unit = 'saantima'
    return '%s %s, %s %s' % (dollar_unit, dollars, cent_unit, cents)
  elif dollars:
    dollar_unit = 'doolaara'
    return '%s %s' % (dollar_unit, dollars)
  elif cents:
    cent_unit = 'saantima'
    return '%s %s' % ( cent_unit, cents)
  else:
    return 'doolaara duwwaa'
```

Figure 5.6 Implementation of Currency Units to word

```
def to_ordinal(self, value):
    self.verify_ordinal(value)
    outwords = self.to_cardinal(value).split(" ")
    lastwords = outwords[-1].split("-")
    lastword = lastwords[-1].lower()
    try:
        lastword = self.ords[lastword]
    except KeyError:
        if lastword[-1:] == "i":
            lastword = lastword[:-1] + "affaa"
        elif lastword[-1:] == "n":
            lastword = lastword + "affaa"
        elif lastword[-1:] == "t":
            lastword = lastword + "taffaa"
        elif lastword[-1:] == "t":
            lastword = lastword + "taffaa"
        else:
            lastword += "ffaa"
    lastwords[-1] = self.title(lastword)
    outwords[-1] = "".join(lastwords)
    return " ".join(outwords)
```

Figure 5.7 Implementation of Ordinals to word.

```
def setup(self):
    self.negword = "negetiva "
    self.pointword = "tuqaa"
    lows = ["non", "okt", "sept", "sext", "quint", "quadr", "tr", "b", "m"]
    units = ["","tokko","lama","sadi","afur","shan","ja'a",
             "toorba","saddeet","sagal"]
    tens = ["kudha", "digdamii", "soddomii", "afurtamii", "shantamii",
            "jaatamii", "toorbaatamii", "saddeettamii", "sagaltamii"]
    self.high_numwords = self.gen_high_numwords(units,tens,lows)
    self.exclude_title = ["f", "tuqaa", "negetiiva"]

    self.mid_numwords = [ (90, "sagaltama"), (80, "saddeettama"), (70, "toorbaatama"),
                          (60, "jaatama"), (50, "shantama"), (40, "afurtama"),
                          (30, "soddoma"),(20, "digdama"),(10, "kudhan")]
    self.low_numwords = [ "sagal", "saddeet","toorba", "ja'a", "shan", "afur", "sadi", "lama",
                          "tokko", "duwwaa"]
    self.ords = {"tokko": "tokkoffaa",
                 "lama": "lammaffaa",
                 "sadi": "sadaffaa",
                 "afur": "afuraffaa",
                 "shan": "shanaffaa",
                 "ja'a": "ja'affaa",
                 "toorba": "toorbaffaa",
                 "saddeet": "saddeettaffaa",
                 "sagal": "sagalaffaa",
                 "kudhan": "kudhanaffaa",
                 }
```

Figure 5.8 Implementation of Number to word conversion

5.3.5 Tokenization

The entire input sequence is tokenized into characters and mapped to IDs, resulting in a sequence of integers called character embeddings.

5.4 Audio Preprocessing

5.4.1 Silence Trimming

Silences in the beginning and end of the audio signal need to be trimmed because they may affect the training speed. As shown in Figure 5.9, we used a threshold of 40 decibels for silence trimming.

```
import librosa
import librosa.filters
import math
import numpy as np
from scipy import signal
from hparams import hparams
from scipy.io import wavfile

def trim_silence(wav, hparams):
  #Trim leading and trailing silence

  return librosa.effects.trim(wav, top_db= 40, frame_length=2048, hop_length=512)[0]
```

Figure 5.9 Audio Silence Trimming

5.4.2 Pre-emphasizing

In our work, to improve the signal-to-noise ratio of the high-frequency portion, we pre-emphasized the audio signal using the filter coefficient of 0.97.

```
import librosa
import librosa.filters
import numpy as np
import tensorflow as tf
from scipy import signal
from scipy.io import wavfile

def preemphasis(wav, k, preemphasize=True):
  if preemphasize:
    return signal.lfilter([1, -k], [1], wav)
  return wav
```

Figure 5.10 Pre-emphasizing

5.4.3 Rescaling

In order to ensure that the audio files are in the range of [1,-1], we applied the rescaling method on each of the audio files with rescaling_max of 0.99. This process is almost equivalent to normalization.

```
if hparams.rescale:
  wav = wav / np.abs(wav).max() * hparams.rescaling_max
  preem_wav = preem_wav / np.abs(preem_wav).max() * hparams.rescaling_max

  #Assert all audio is in [-1, 1]
  if (wav > 1.).any() or (wav < -1.).any():
    raise RuntimeError('wav has invalid value: {}'.format(wav_path))
  if (preem_wav > 1.).any() or (preem_wav < -1.).any():
    raise RuntimeError('wav has invalid value: {}'.format(wav_path))
```

Figure 5.11 Rescaling

5.5 Feature Extraction

After preprocessing the dataset, the next important audio analysis stage is the audio feature extraction. It is used to extract a set of acoustic features that can provide a representative characteristic of the audio frame. For our study, linear scale spectrogram and mel scale spectrogram are used as an intermediate future representation. The implementation of linear spectrogram and mel spectrogram using librosa python library is presented in Figure 5.12.

```
def linearspectrogram(wav, hparams):
  D = _stft(wav, hparams)
  S = _amp_to_db(np.abs(D)**hparams.magnitude_power, hparams) - hparams.ref_level_db

  if hparams.signal_normalization:
    return _normalize(S, hparams)
  return S

def melspectrogram(wav, hparams):
  D = _stft(wav, hparams)
  S = _amp_to_db(_linear_to_mel(np.abs(D)**hparams.magnitude_power, hparams), hparams) - hparams.ref_level_db

  if hparams.signal_normalization:
    return _normalize(S, hparams)
  return S
```

Figure 5.12 Implementation of linear spectrogram and mel spectrogram

5.5.1 Feature Normalization

Before starting training of the model, the minimal and the maximal linear and mel scale spectrogram magnitudes are calculated for each data element. Before inputting the spectrograms for training or evaluation they are converted into an absolute decibel representation with the reference being a magnitude of 1. Towards zero the magnitudes are capped to 10−5, hence −100 dB. Afterward, they are linearly scaled down to the range [−100; 0] dB using the minimum and maximum decibel magnitudes calculated over the datasets. Subsequently, this range is scaled linearly such that the numeric values fit the range [0; 1].

```
def _normalize(S, hparams):
  if hparams.allow_clipping_in_normalization:
    if hparams.symmetric_mels:
      return np.clip((2 * hparams.max_abs_value) * ((S - hparams.min_level_db) / (-hparams.min_level_db)) - hparams.max_abs_value,
       -hparams.max_abs_value, hparams.max_abs_value)
    else:
      return np.clip(hparams.max_abs_value * ((S - hparams.min_level_db) / (-hparams.min_level_db)), 0, hparams.max_abs_value)

  assert S.max() <= 0 and S.min() - hparams.min_level_db >= 0
  if hparams.symmetric_mels:
    return (2 * hparams.max_abs_value) * ((S - hparams.min_level_db) / (-hparams.min_level_db)) - hparams.max_abs_value
  else:
    return hparams.max_abs_value * ((S - hparams.min_level_db) / (-hparams.min_level_db))
```

Figure 5.13 Feature Normalization

5.6 Model Implementation

Section 4.3 describes the architectures of the models in more detail and how they were presented in their original publications. In this section, we present the implementations of the models and the details which deviate from the original models. Our dataset is already preprocessed and ready for the training phase. To select the best-performing model for our work, we compare the models Deep Voice 3 and Tacotron 2 TTS using the open-source model implementations. The model hyper-parameters are used with slight modification to get reasonable results on our dataset with considering our computational resources. Since we are training for a new dataset the training process for both models started from scratch.

Even though both DV3 and T2 TTS system implementation were capable of training the Afaan Oromo speech dataset, some of the frontend components need to be modified. All required operations of an input text cleaning such as text to sequence conversion, removal of redundant white spaces, the addition of punctuation to indicate the end of the sentence, and converting into lower-case can be handled by the existing frontend. Since both DV3 and T2 models are proposed by considering the English language, their frontend is not capable of handling Afaan Oromo non-standard words such as years, currencies, numbers, abbreviations, and acronyms. For this reason, a new normalization frontend was written specifically for Afaan Oromo. The ARPAbet from the existing frontend was completely omitted since it does not work for Afaan Oromo texts. Omitting the ARPAbet, required making the pronunciation probability parameter equal to zero, which is highly related to retrieving the phonemes representations of the words from the phonemes dictionary.

5.6.1 Deep voice 3 Based Implementation

The Deep Voice 3 architecture that is shown in section 4.3.1 was trained using our dataset from scratch with hyper-parameters listed in Appendix D. The hyper-parameters suggested by DV3 worked for our model, thus we used the same hyper-parameters without increasing the demand due to our resource limitations. For the implementation of DV3, we have used the Pytorch implementation from Ryuichi Y[13]. The implementation contains two phases these are the training phase where we conduct the training of DV3 using our speech dataset and the synthesis phase where we input new text and perform the synthesis using the trained model with a Griffin-Lim algorithm.

[13] https://github.com/r9y9/deepvoice3_pytorch

Training phase:

The training of DV3 was performed by using NVIDIA Tesla T4 GPU with 16GB RAM. The total training time by using a batch size of 8 took 12 days and 16 hours. The training was completed after 1M steps. The attention changing during the training process is presented in Figure 5.14 below. As we can observe from the figure, the model started to produce an intelligible, understandable, and partially human-like speech after 40 K steps. Subsequently, the model started to improve itself by losing the robotic nature in the synthesized speech.

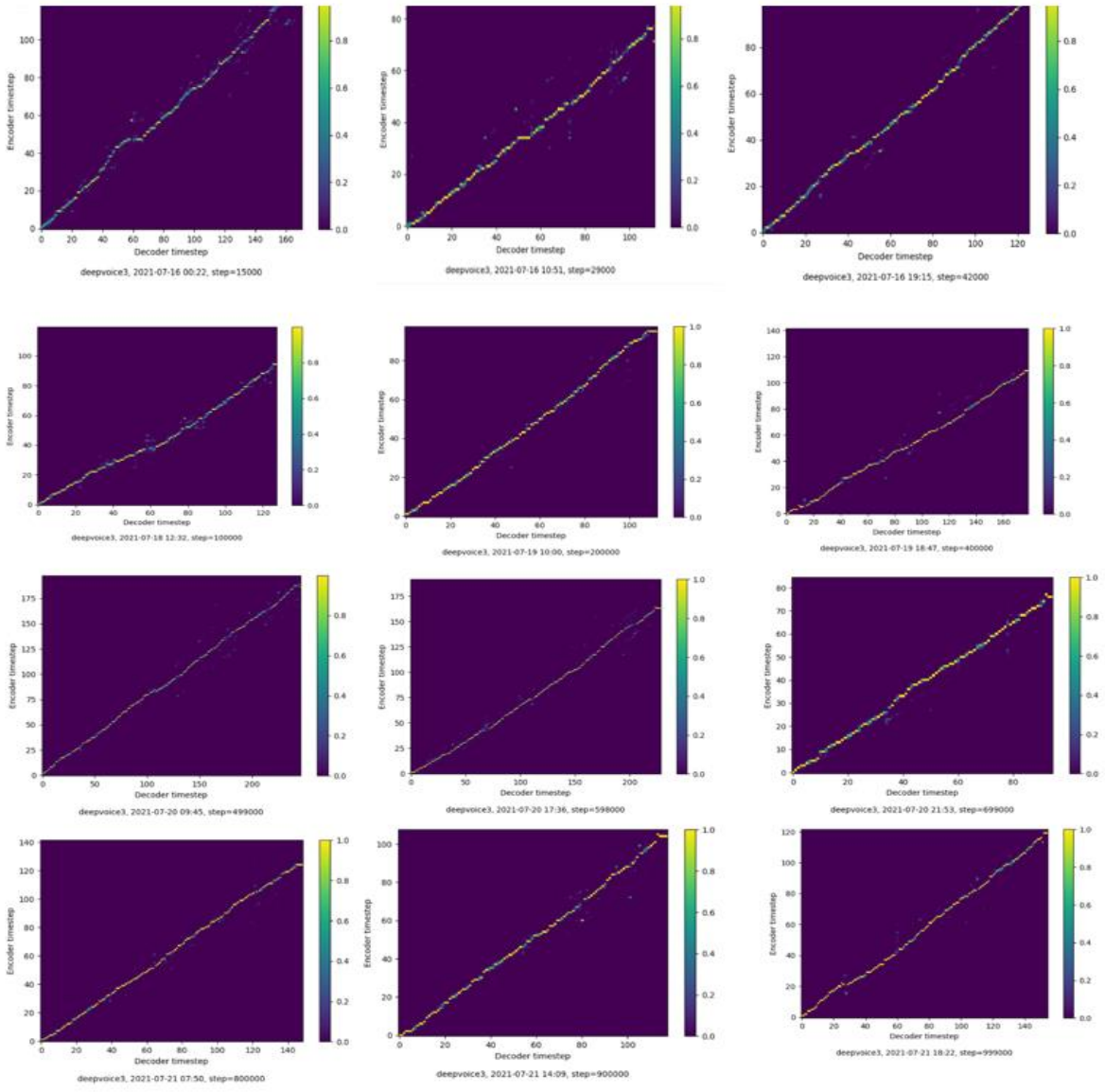

Figure 5.14 Attention changing for DV3 model during the training process.

The training of the model was evaluated every 10K steps before the training process proceeds. The evaluation was performed using six unseen sentences that are carefully chosen to be decisive to test the ability of the models. The sentence examples cover special cases in

the Afaan Oromo language, such as long words; compound words; the glottal stop *(Hudhaa)* characters, and comma somewhere in the sentence to check whether the model makes the appropriate pause when synthesizes the speech; sentences ending with a full stop, question mark, and exclamation mark to check whether the model can change the intonation in the synthesized speech. The attention change for a sentence: *"Kunoo, Waaqayyo gooftaan keenya ulfina isaa fi guddina isaa nu argisiiseera, nuyis ibiddicha gidduudhaa sagalee isaa dhageenyeerra;"* could be seen in Figure 5.15.

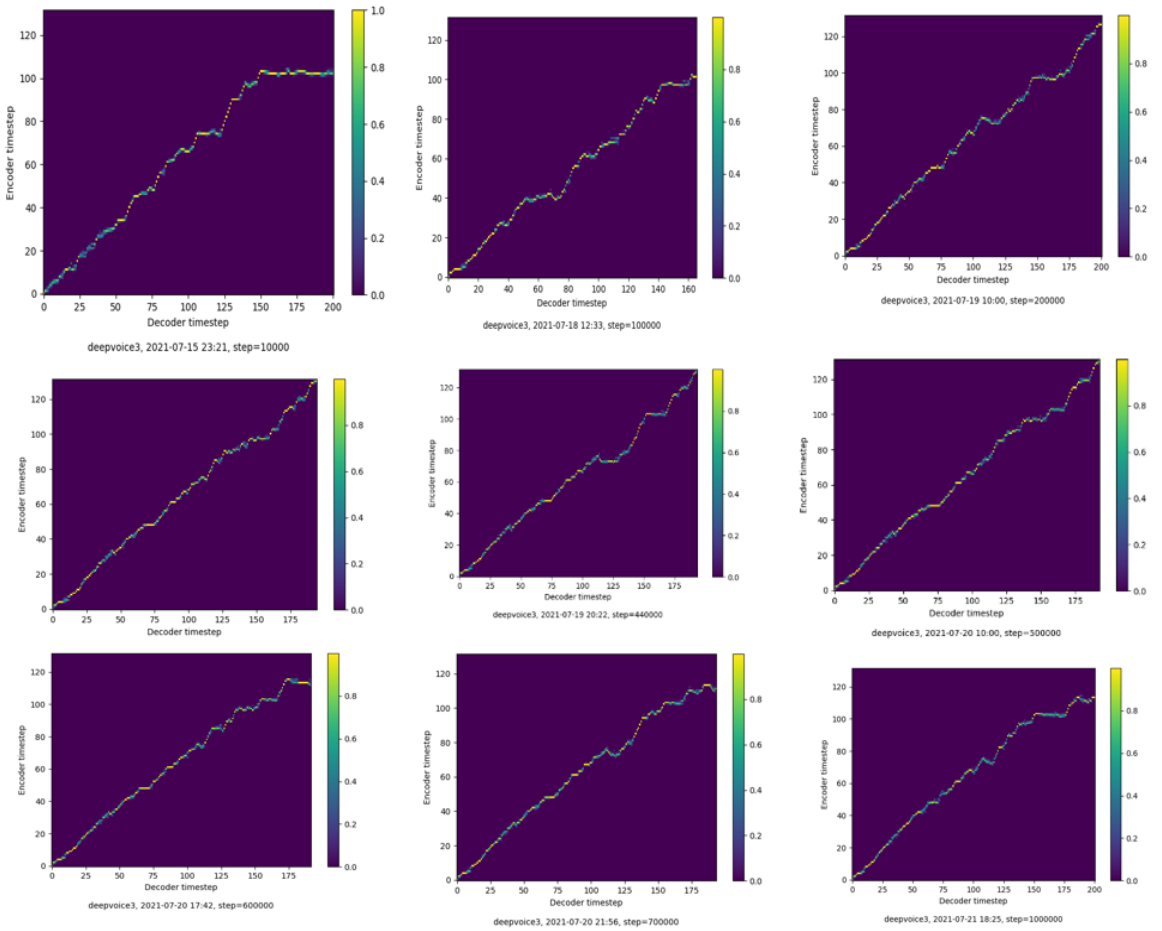

Figure 5.15 Attention Alignment plot for the above sentence

The evaluation metrics that explain the performance of the training models are presented in Figure 5.16. The loss function is a metric that measures how far an estimated value is from its true value. Since the main objective is to minimize the loss function, the Deep learning algorithm will repeat as many times as needed for the loss to reach as flatter shape as possible. In our case, the loss functions behave in a desired manner, it gradually decreases, converging to a value of 0.202 after 88 K steps in three days and 10 h of training.

Learning rate plays a vital role in minimizing the loss function. It dictates the speed at which we want our model to learn. This value must be set properly, since if not, for example, setting it too high, the model will not have time to learn anything and, thus, the results will be poor. According to DV3 architecture, in our case, the initial learning rate was set to 0.0005. After four days and 10 h of training, or 94 K steps, it decreases to a value of 0.00010304.

The gradient norm that is presented in the same figure calculates the L2 norm of the gradients of the last layer of the Deep learning network. It automatically balances training in deep learning models by dynamically tuning gradient weights. If its value is too small, it might indicate a vanishing gradient. This problem affects the upper layers of the Deep learning network, making it hard for the network to learn and tune the parameters. On the contrary, if its value is too high, it may indicate exploding a gradient phenomenon. In such a case, the model will be unstable and it will not able to learn from data since the accumulation of large error gradients during the training process results in very large updates in the Deep learning model weights.

The last performance metric presented in Figure 5.16 is L1 norm that refers to the ability of the model to predict the mel-spectrograms. The L1 norm of the metric is decreasing across the iteration steps, reaching 0.04104 after 100 K steps, or five days and 8 h of training.

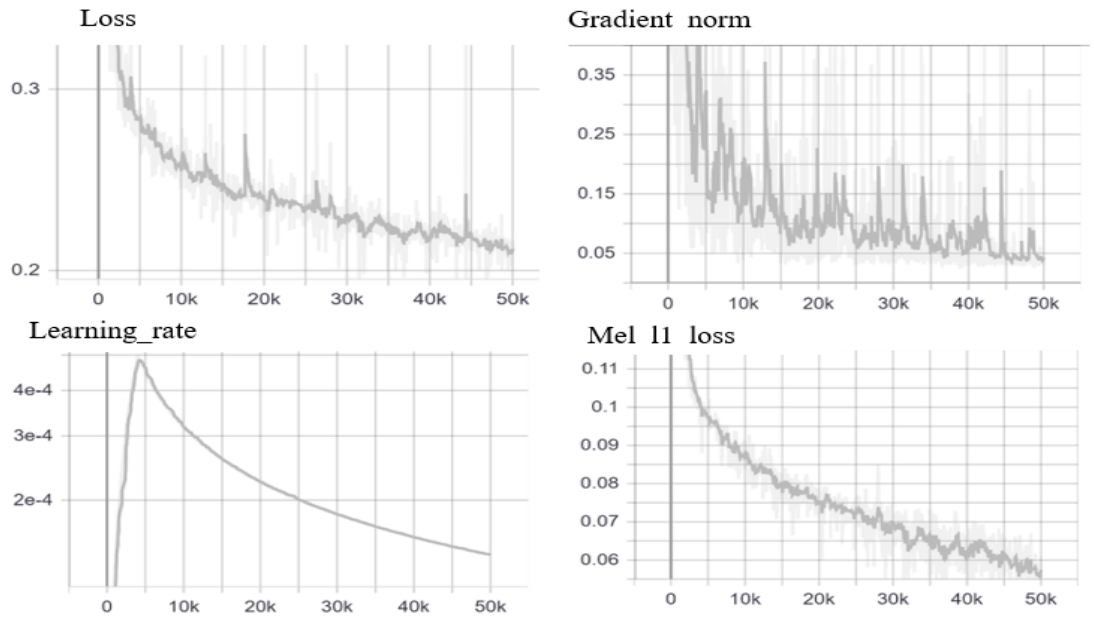

Figure 5.16 Tensor board scalars of the learning models

Synthesis phase:

Following the training and validation stages, the inference stage was performed. This stage itself is independent of the previous stage and uses the leverage of the already created checkpoints. When new text is feed into the model, it needs to be passed through the language-specific frontend that normalizes non-standard words such as numbers, abbreviations, currencies, and acronyms as explained in the text and audio preprocessing stage. Large input sentences are also segmented and batched to preserve the order. Each of them is predicted by the last checkpoint upon which an intelligible speech is synthesized. During the audio post-processing, the segments are put together and concatenated to be outputted as a single .wav file. Finally, we have synthesized ten new sentences using the last checkpoint for User Evaluation of the model performance.

5.6.2 Tacotron 2 Based Implementation

In this study, the Tacotron 2 model presented in section 4.3.2 has been trained from scratch using our dataset. We used an open-source Tensorflow implementation of Tacotron 2 from Rayhane M[14] where the WaveNet part has been striped with a slight modification of default hyperparameters to deal with our time and computing capabilities. The implementation contains two phases these are the training phase where we train the feature prediction network and the synthesis phase where we feed new text to the model and generate a synthesized speech using the trained model with a Griffin-Lim algorithm.

Training the Feature Prediction Network

We trained the feature prediction model for 100k steps, with a batch size of 16 using a single GPU over a total computation duration of 16 days. The model is trained on both the linear and mel representation of the raw audio signal features representations. The attention changing during the training process is presented in Figure 5.17 below.

[14] https://github.com/Rayhane-mamah/Tacotron-2

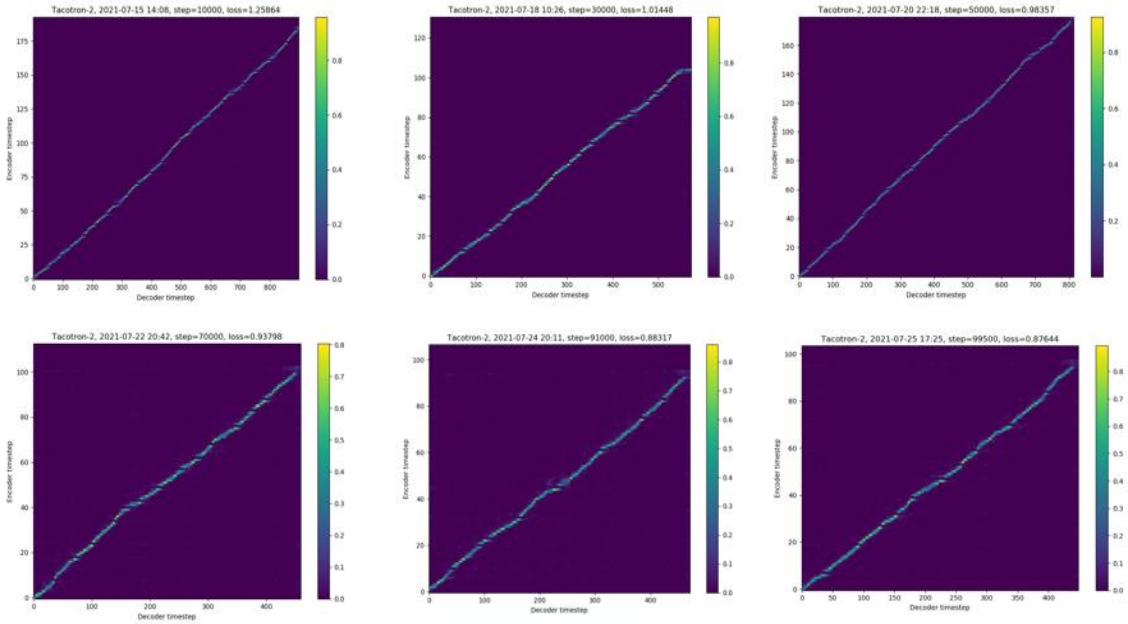

Figure 5.17 The attention changing during the training

The number of frames to generate at each decoding step is set to 1, as is done in the open-source implementation. During training, the model is set in Ground Truth Aligned (GTA) mode, where the input to the pre-net is the previous frame of the ground truth spectrogram instead of the predicted one. With GTA, the pitch and prosody of the generated spectrogram are aligned with the ground truth, allowing for a shared context between the prediction and the ground truth as well as faster convergence. Without GTA, the synthesizer would generate different variations of the same utterance given fixed text input.

In our experiments, it was difficult to provide any quantitative assessment of the performance of the model. It can be seen that the model is producing correct results by informal listening tests because the formal evaluation would require us to set up subjective evaluations such as the MOS listening test by user's participation. As an alternate assessment of the results, we use the alignments generated by the attention module. The example is shown in Figure 5.18, it presents the alignment between the encoder and the decoder steps (left) and a comparison between the GTA predicted spectrogram and the ground truth spectrogram (right).

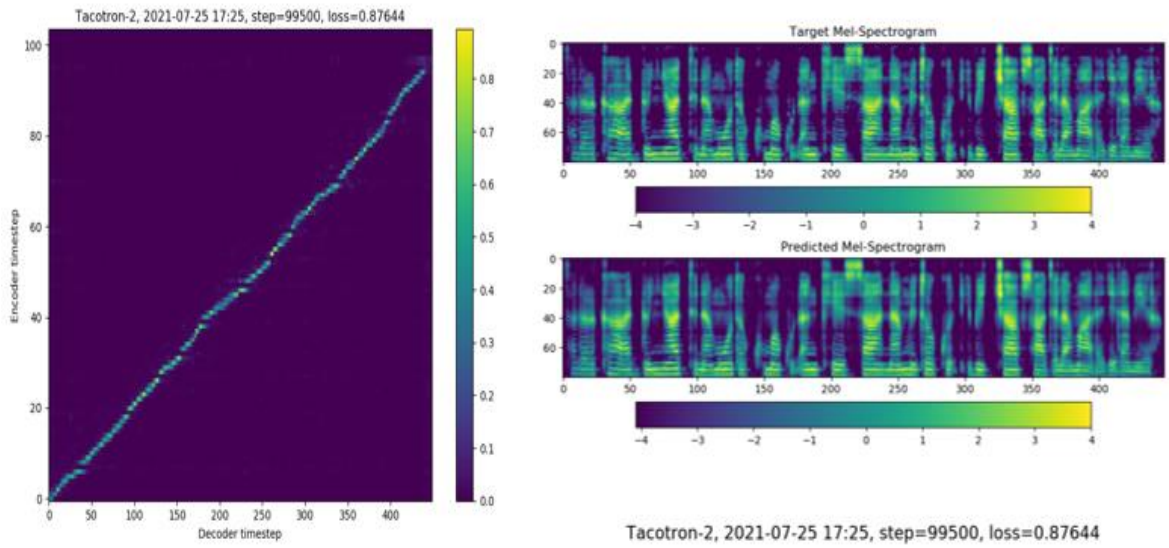

Figure 5.18 Example of training alignment.

The model was evaluated every 2500 steps on the validation portion of the dataset. The progression of the predicted attention alignments is illustrated in Figure 5.19. The alignments are predicted during synthesis for different amounts of training steps.

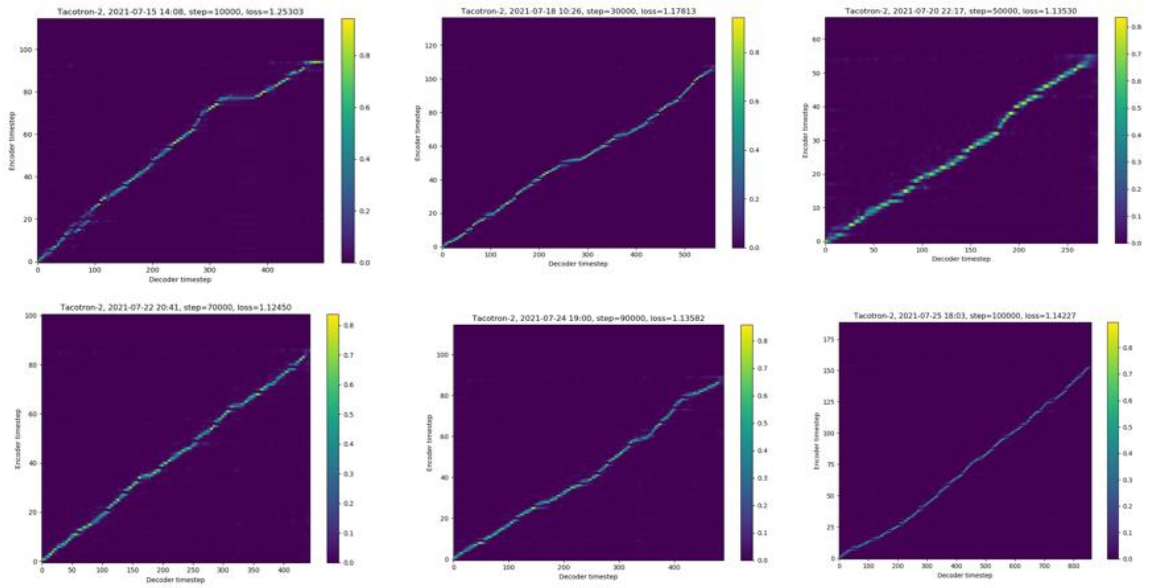

Figure 5.19 Attention changing during the Evaluation phase

The evaluation metrics that explain the performance of the training models are presented in Figure 5.20. The loss function behaves in the desired manner, it gradually decreases converging to a value of 0.889 after 100k steps. The learning rate was initially set to 0.001, it decreases to a value of 0.0001 after 100k steps. The gradient norm is also behaving in the desired manner it was not too high and too low throughout the training process. Another

metrics is the evaluation loss that is substantially larger than the training loss due to the feeding of ground-truth frames during training. The evaluation loss was converging throughout the process this indicates that the model is not overfitting.

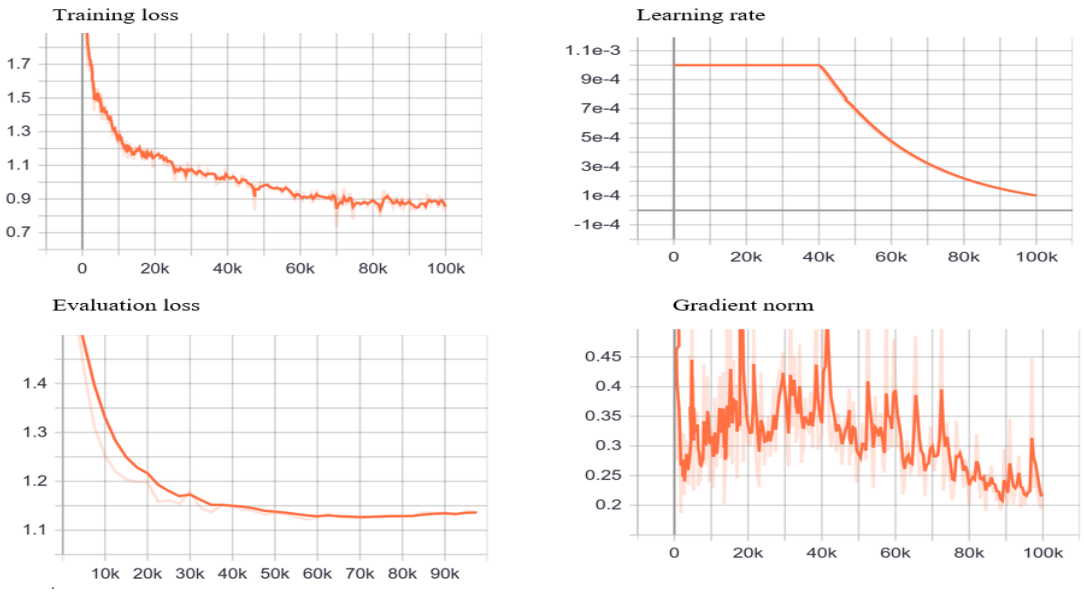

Figure 5.20 The performance of the Tacotron 2 model

Synthesis Phase:

In this work, since we trained Tacotron 2 model without WaveNet we shall use the Griffin-Lim algorithm presented in section 4.4.1 as a vocoder to evaluate the progress of the trained feature prediction model. The audio generated by the Tacotron 2 matches correctly the text, even in the presence of complex and non-standard words. However, the prosody is sometimes unnatural, with pauses at unexpected locations in the sentence, or the lack of pauses where they are expected.

The Tacotron 2 model has been trained with both linear and mel spectrogram representations of the audio signals. Therefore, the evaluation of this model was conducted using both linear and mel spectrograms. For the user evaluation, we have synthesized ten new sentences using this model in both forms.

CHAPTER SIX

6. EVALUATION AND DISCUSSION

This chapter presents the performance evaluation process and discussion about the result. In this section, the subjective and objective evaluation of proposed models has been conducted and the result is also presented. In the end, there is a discussion section where the experiment results are discussed.

6.1 Evaluation

Currently, the evaluation of Text-to-speech synthesis systems is performed in different ways. The evaluation of our model performance is conducted using objective and subjective evaluations by participating volunteers native speakers. For this purpose, a total of ten new sentences that are unseen during training time has been prepared. The utterances were prepared by taking into consideration the non-standard words such as ordinals, numbers, abbreviated words, acronyms, currency, and punctuation marks of the language. The utterances also contain very long sentences, complex sentences; the glottal stop *(Hudhaa)* characters, and comma somewhere in the sentence to check whether the model makes the appropriate pause when synthesizes the speech. The participants were volunteers from Adama Science and Technology University.

6.1.1 Preparing Questionnaire

In this study, before conducting the evaluation we have prepared a questionnaire that focuses on the intelligibility and naturalness of the synthesized speech by the proposed model. The questionnaire is given in Appendix A. The questionnaire contains eleven items five for measuring the naturalness and six for measuring the intelligibility of the synthesized speech.

6.1.2 Objective Evaluation

We have conducted the objective evaluation to evaluate the performance of the two models. As stated by (Ping et al., 2018), the attention-based TTS systems run into attention errors such as repeated words, mispronunciations, and skipped words. In order to measure the performance of our models against these errors, we have carried out the objective evaluation of the models. The evaluation speech along with its normalized sentence is given to the participants. The participants listen to the evaluation speeches and manually calculate the occurrence of the above errors.

The statistics of the occurrence of attention errors in the evaluation sentences for both models are given in Table 6.1. The participants were asked to count occurrences of repeated words, skipped words, and mispronunciations in each utterance. Occurrence one or more mispronunciation skips and repeats count as a single error per utterance. Our evaluation sentences list contains ten unnormalized sentences yielding 148 words after normalization. The detailed report is included in Appendix C. 1.

Table 6.1 The scores of objective evaluations.

Model	***Repeated words***	***Mispronunciations***	***Skipped words***
Deep Voice 3	6	8	2
Tacotron 2	0	2	0

6.1.3 Subjective Evaluation

Subjective evaluation has been conducted to measure the naturalness and intelligibility of the trained models. For subjective valuation of the text to speech synthesis model, a mean opinion scale (MOS) listening test was carried out. It has been the recommended subjective performance assessment of the synthesized speech quality since 1994 (ITU-T, 1994).

The subjective evaluation in our experiments required the participants to be fluent in Afaan Oromo. Commercial crowdsourcing platforms such as MTurk[15], Microworkers [16], and CrowdFlower[17] don't usually offer Afaan Oromo workers. For this reason, we had to prepare our questionnaire, and the participants were volunteers from native speakers. The respondents were given a printed copy of the questionnaire as shown in Appendix A before they took the listening test. The questionnaire contains eleven items with the numbers 5-1 alongside phrase anchors where 5 refers to the more positive scores on the item.

At the beginning of the test, participants were asked to find a quiet listening environment and use headphones for listening. They were also instructed to adjust the volume to a comfortable level and requested not to change the volume during the test. Changing the audio volume during the test might affect the results. In the MOS test, seven responses were collected from volunteer participants. The overall mean opinion score result has been shown in Table 6.2 and the detailed mean opinion score report was presented in Appendix C. 2.

[15] https://microworkers.com
[16] https://www.crowdflower.com
[17] https://crowdee.de

Table 6.2 The Mean Opinion Score result

Experiment	***Intelligibility***	***Naturalness***
Deep Voice 3	3.28	3.02
Tacotron 2 (Linear spectrogram)	4.32	4.14
Tacotron 2(Mel spectrogram)	4.32	4.21

6.2 Result and Discussion

To our knowledge, this study is the first attempt to develop a text-to-speech synthesizer model for Afaan Oromo using the deep neural network method. We conducted several experiments on our newly prepared Afaan Oromo speech dataset and got promising results.

The first experiment is training Tacotron 2 feature prediction model that is based on a recurrent neural network by tuning the hyperparameters to train with our dataset. Because of the use of the recurrent neural network, the training was very slow. The training is carried out using both linear and mel spectrogram representations of the raw audio signals. This experiment started to produce an acceptable result after 50k steps overall the model was trained for 100k steps. In this experiment, we have used the Griffin-Lim algorithm to convert the predicted acoustic features into time-domain waveforms of the audio. This experiment was successful in both objective and subjective evaluations.

The second experiment is the training of Deep Voice 3 that is based on a convolutional neural network. This experiment was faster than the first experiment since it employed the convolutional layers on all of its sub-components. The experiment was conducted for 1M steps in order to find an acceptable result. The model is trained on mel spectrogram representation of the raw audio signals. The Griffin-Lim algorithm has been used for generating audio files from the predicted acoustic features. The attention alignment curves for sentences from a test list depicted in Figure 5.14 were not as good as that of Tacotron 2, and this results in a lower MOS score. The speeches produced by the Deep Voice 3 model contain artifacts and the voice did not sound smooth and continuous.

The objective and subjective evaluation tests have been performed on both experiments in order to assess their performance. The objective evaluation shows that the Tacotron 2 feature prediction model outperforms in terms of attention error since very few attention errors are made by this model. Quantitatively 2 errors out of 148 words are made by Tacotron 2

whereas the Deep Voice 3 made 16 attention errors out of 148 words. This results from two reasons these are the unavailability of Afaan Oromo phoneme dictionary that contains words with their pronunciation to use explicit grapheme-to-phoneme model and attention mechanism employed since Deep voice 3 uses dot-product attention mechanism replacing it with that Tacotron 2, location-sensitive attention might improve the attention errors.

In the subjective evaluation, the MOS listening test has been carried out for assessing the naturalness and intelligibility of the synthesized speech samples. The MOS test is conducted on Deep voice 3 trained with a mel spectrogram, Tacotron 2 trained with a mel spectrogram, and Tacotron 2 trained with a linear spectrogram. The Deep voice 3 model scored 3.02 and 3.28 out of five in naturalness and intelligibility respectively. The Tacotron 2 trained on linear spectrogram scored 4.32 out of five in intelligibility and 4.14 out of five in naturalness. The Tacotron 2 trained with mel spectrogram on the other hand scored 4.32 and 4.21 out of five in intelligibility and naturalness respectively.

As summarized in Table 6.1 and Table 6.2, the evaluation result indicates that the Tacotron 2 model trained on mel spectrogram representation outperforms the remaining models in overall performance. The Tacotron 2 trained on linear spectrogram also provides a very close result to the first one. The deep voice 3 model needs considerable effort to improve its performance to be accepted as a working text-to-speech model.

Besides this, we have conducted another experiment by replacing the Griffin-Lim algorithm with WaveNet, a trainable neural network as a vocoder. We have trained the open-source implementation of WaveNet from (Shen et al., 2018) and (Ryuinchi Yamamoto, 2018) to use along with the feature prediction models. Because of the limitation of time and computational resources the experiment was not successful. WaveNet can generate a speech that mimics any human voice and produce sounds more natural than the best existing Text-to-Speech systems, reducing the gap with human performance by over 50% (Aäron van den Oord, 2016).

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORKS

This chapter presents the conclusion drawn from the works done in this thesis based on experiments performed and the result provided in the first section. The recommendation for future works has been presented in the second section.

7.1 Conclusion

Speech synthesis mostly refers to the Text-to-Speech (TTS) system is the artificial generation of equivalent audio from the given text transcription. It is an interdisciplinary technology that takes into account several disciplines such as acoustics, linguistics, digital signal processing, and statistics. The qualities of speech synthesis systems are assessed by their naturalness and intelligibility. TTS systems can be modeled using a variety of techniques that are grouped into three these are rule-based, concatenative, and statistical parametric methods. In this work, a first attempt is made to develop a speech synthesizer for Afaan Oromo using the deep neural network method.

During the development of the Afaan Oromo text-to-speech synthesis model, we have prepared our speech dataset as there is no previously prepared and open-sourced Afaan Oromo speech dataset in the domain of text-to-speech. To accomplish this, we have collected a phonetically rich and balanced text corpus from three major sources, these are news sources, non-fiction books, and Holy Bible. Since the model consumes <text, audio> pairs we had to create the audio signals recorded from the corresponding text transcription. High-quality audio signals are recorded in a professional studio using a microphone by a male native speaker. A total of 8,076 speech datasets with text transcription and equivalent audio signals were prepared for training the model.

The proposed solution architecture contains the training and inference phases. In the training phase, the aligned speech dataset was passed through the preprocessing step where both text and audio data are preprocessed before being fed to the model for training. The text preprocessing performed here are text cleaning, normalization, number to word conversions, and tokenization. The audio preprocessing performed here ar: silence trimming, re-scaling, and pre-emphasizing. the linguistic features are extracted from the input texts and acoustic features are extracted from the input audio signals, and then the DNN-based acoustic model is trained based on linguistic and acoustic features.

In the synthesis or inference phase, the trained acoustic model predicts acoustic features from the given linguistic features (the given unseen text), and then the vocoder produces the synthesized audio signal from the predicted acoustic feature parameters. We used the Griffin-Lim algorithm to convert the acoustic features (mel and linear spectrograms) predicted by the acoustic feature prediction network into time-domain waveforms.

The objective and subjective evaluations are conducted in this study to evaluate the performance of both the recurrent neural network and convolutional neural network acoustic models. In the objective evaluation, the Tacotron 2 model made very few attention errors quantitatively 2 errors out of 148 words whereas the Deep Voice 3 made 16 attention errors out of 148 words. For subjective evaluation, we have used the mean opinion score (MOS) listening test on the user side. We found the MOS result of 4.32 and 4.21 out of five for Tacotron 2, and 3.28 and 3.02 out of five for Deep Voice 3 in terms of intelligibility and naturalness respectively. Furthermore, as per the subjective and objective evaluation results, we found that the Tacotron 2, based on a recurrent neural network model outperforms the Deep voice 3 that is a fully convolutional neural network model.

Many human-computer interaction systems such as recommendation systems, telephone inquiry services, and smart educations can be benefited from this work by creating an interaction with the users. It improves their experience by providing a human-like communication method.

The major contributions of this research are presented as follows:

- Development of audio recording tools that facilitate the audio recording process by providing the text transcription to be recorded with its aligned audio file name.
- We have addressed the text analysis issue by developing the Afaan Oromo text analysis frontend.
- Development of Text-to-Speech synthesis model for Afaan Oromo using deep neural network.
- Collection and preparation of phonetically rich and balanced Afaan Oromo speech dataset for Text-to-Speech problems.

7.2 Future Works

Even though we have attempted to implement the two most outperforming acoustic parameter prediction models, there are also a variety of models that are based on deep neural networks and achieved a good result for other languages. From these models, it could be good progress to implement EATS – End-to-end Adversarial Text-to-Speech – generative models that can synthesize in an unsupervised way (where input and output are not aligned) for Afaan Oromo. It contains a component that maps the unaligned input into a representation that is aligned with the output called the aligner and the decoder that upsamples the aligner's output to the full audio frequency.

We have used a Griffin-Lim algorithm during inference as a vocoder to generate the audio signals from the predicted spectrogram representation. The requirement of many iterations for the signal generation process and generation of low-quality audio signals, make it difficult to apply for real-time systems. To handle these shortcomings, we recommend the use of a learning-based approach such as WaveNet that models the audio waveforms generation process using a deep neural network and training it using the idea of a generative adversarial network.

In this study, the single-speaker method has been implemented we suggest the interested researchers incorporate the voice cloning techniques such as speaker adaptation and speaker encoding. Voice cloning enables the model to interpret the voice of the new and unseen speaker from a few voice samples provided. The use of phoneme level embeddings outperforms character embedding in several studies (Jia et al., 2018) incorporating the phoneme embedding by using a phoneme dictionary will minimize the attention errors such as repeated words, mispronunciations, and skipped words. The inclusion of all non-standard word normalization can be achieved by using the machine learning classifier with many features.

***

SPECIAL ACKNOWLEDGMENT

This research was funded by Adama Science and Technology University with grant number **ASTU/SM-R/228/11**

REFERENCES

Alam, M., Samad, M. D., Vidyaratne, L., Glandon, A., & Iftekharuddin, K. M. (2020). Survey on Deep Neural Networks in Speech and Vision Systems. *Neurocomputing*, *417*, 302–321. https://doi.org/10.1016/j.neucom.2020.07.053

Alan S. Kaye and Peter T. Daniels. (1997). Oromo Phonology. *Phonologies of Asia and Africa*.

Aliero, A. A., Muhammed, D., & Ibrahim, A. (2019). *Taxonomy, Review and Research Challenges Of DNN-Based Text-To-Speech System for Hausa as Under-Resourced Language*. *10*(7), 548–560.

Aliero, A., Muhammed, D., & Ibrahim, A. (2020). *Taxonomy, Review and Research Challenges Of DNN-Based Text-To-Speech System for Hausa as Under-Resourced Language*. *10*, 13.

Araya Keletay, M., & Seid Worku, H. (2020). Developing Concatenative Based Text to Speech Synthesizer for Tigrigna Language. *Internet of Things and Cloud Computing*, *8*(2), 24. https://doi.org/10.11648/j.iotcc.20200802.12

Arik, S. Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Raiman, J., Sengupta, S., & Shoeybi, M. (2017). Deep Voice: Real-time Neural Text-to-Speech. *CoRR*, *abs/1702.0*. http://arxiv.org/abs/1702.07825

Arık, S. Ö., & Miller, J. (2017). *Deep Voice 2*. *1*(Nips), 1–9.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, 1171–1179.

Black, A. W. and T. P. (1994). *CHART: a generic speech synthesis system*.

Britz A. (2016). *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Burrows, T. (1996). *Speech processing with linear and neural network models*.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using {RNN} Encoder-Decoder for Statistical Machine Translation. *CoRR*, *abs/1406.1*. http://arxiv.org/abs/1406.1078

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-Based Models for Speech Recognition. *CoRR*, *abs/1506.0*. http://arxiv.org/abs/1506.07503

De La Puente, P. (2016). *Deep learning applied to Speech Synthesis*. https://upcommons.upc.edu/bitstream/handle/2117/100133/msc-thesis-final.pdf

Dertat A. (2018). *Applied Deep Learning - Part 4: Convolutional Neural Networks*. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Docio-Fernandez, L., & García Mateo, C. (2015). Speech Production. In S. Z. Li & A. K. Jain (Eds.), *Encyclopedia of Biometrics* (pp. 1493–1498). Springer US. https://doi.org/10.1007/978-1-4899-7488-4_199

Dutoit, T. (2008). A Short Introduction to Text-to- Speech Synthesis. *Reading*, *73*(10–12), 1–16. http://linkinghub.elsevier.com/retrieve/pii/S0925231210001025

Edward M. (2019). *Besides Word Embedding, why you need to know Character Embedding?*

Englewood Cliffs, N. J. (1993). *Fundamentals of speech recognition*. PTR Prentice Hall.

Fahmy, F. K., Khalil, M. I., & Abbas, H. M. (2020). A transfer learning end-to-end arabic text-to-speech (tts) deep architecture. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12294 LNAI*, 266–277. https://doi.org/10.1007/978-3-030-58309-5_22

Figueiredo, A., Imbiriba, T., Bruckert, E., & Klautau, A. (2006). *Automatically Estimating the Input Parameters of Formant-Based Speech Synthesizers*.

Flanagan, J. (1971). *Speech Analysis, Synthesis and Perception*.

Gada Melba. (1988). *the Oromo people and Oromia: Introduction*.

Gebreselassie, T. A. (2009). No Title. In *Text-To-Speech Synthesizer for Wolaytta Language*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. *CoRR*, *abs/1705.0*. http://arxiv.org/abs/1705.03122

Girma, T. (2014). Human Language Technologies and Affan Oromo Issn : 2278-6252 I . Introduction. *International Journal of Advanced Research in Engineering and Applied Sciences*, *3*(5), 1–13.

Griffin, D., & Lim, J. S. (1983). Signal estimation from modified short-time Fourier transform. *ICASSP*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Holmes, J., & Holmes, W. (2002). *Speech Synthesis and Recognition* (2nd ed.). Taylor & Francis, Inc.

Hunt, A. J., & Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *1*, 373–376. https://doi.org/10.1109/icassp.1996.541110

Isewon, I., Oyelade, J., & Oladipupo, O. (2014). Design and Implementation of Text To Speech Conversion for Visually Impaired People. *International Journal of Applied Information Systems*, *7*(2), 25–30. https://doi.org/10.5120/ijais14-451143

ITU-T. (1994). P.85 : A method for subjective performance assessment of the quality of speech voice output devices. *ITU-T Recommendation*, *1*.

J. Brownlee. (2021). *How to Choose an Activation Function for Deep Learning*. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/?__cf_chl_captcha_tk__=pmd_131ad8a63cdf6cd87c5ca33f650a45c0a34c3ec5-1629141233-0-gqNtZGzNAvijcnBszQnO

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I. L., & Wu, Y. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in Neural Information Processing Systems*, *2018-Decem*(NeurIPS), 4480–4490.

Kebede, F. B. (2014). Dissimilation in Oromo Phonology. *International Journal of Innovative Research & Development*, *13*(13), 187–196.

Klatt, D. H. (1987). Review of text-to-speech conversion for English. *The Journal of the Acoustical Society of America*, *82*(3), 737–793. https://doi.org/10.1121/1.395275

Lemmetty, S. (1999). Review of speech synthesis technology. *Helsinki University of Technology*, *320*, 79–90. http://www.acoustics.hut.fi/~slemmett/dippa/%5Cnhttp://www.acoustics.hut.fi/publications/files/theses/lemmetty_mst/

Li, N., Liu, S., Liu, Y., Zhao, S., & Liu, M. (2019). Neural speech synthesis with transformer network. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 6706–6713. https://doi.org/10.1609/aaai.v33i01.33016706

Macon M. W. (1996). Speech Synthesis Based on Sinusoidal Modeling. *Doctorial Thesis, Georgia Institute of Technology*.

Mengasha Riqitu. (1993). *How to read Oromiffa and use its grammar*. Magic Press.

Mishev, K., Ristovska, A. K., Trajanov, D., Eftimov, T., & Simjanoska, M. (2020). Makedonka: Applied deep learning model for text-to-speech synthesis in macedonian language. *Applied Sciences (Switzerland)*, *10*(19), 1–14. https://doi.org/10.3390/app10196882

Morise, M., YOKOMORI, F., & Ozawa, K. (2016). WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications. *IEICE Transactions on Information and Systems*, *E99.D*, 1877–1884. https://doi.org/10.1587/transinf.2015EDP7457

Morka, M. (2001). Text-to-Speech system for Afaan Oromoo. *A Thesis Submitted in Partial Fulfilment of the Requirement for the Degree of Master of Science in Information Science, June*.

Näslund, P. (2018). *Artificial Neural Networks in Swedish Speech Synthesis*.

Ning, Y., He, S., Wu, Z., Xing, C., & Zhang, L. J. (2019). Review of deep learning based speech synthesis. *Applied Sciences (Switzerland)*, *9*(19), 1–16. https://doi.org/10.3390/app9194050

Obius, B., Schroeter, J., Santen, J., Sproat, R., & Olive, J. (2003). *Recent Advances In Multilingual Text-To-Speech Synthesis*.

Omniglot. (2021). *Oromo language, alphabet and pronunciation*. https://omniglot.com/writing/oromo.htm

Oord, Aaron van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). *WaveNet: A Generative Model for Raw Audio*. 2020. http://arxiv.org/abs/1609.03499

Pärssinen, K. (2007). Multilingual text-to-speech system for mobile devices: development and applications. *Unpublished Doctoral Thesis*.

Ping, W., Peng, K., Gibiansky, A., Arık, S., Kannan, A., Narang, S., Raiman, J., & Miller, J. (2018). Deep Voice 3: Scaling text-to-speech with convolutional sequence learning. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 1–16.

Rashad, M., El-Bakry, H., Isma'il, I. R., & Mastorakis, N. (2010). An overview of text-to-speech synthesis techniques. *ICC 2010*.

Ryuinchi Yamamoto. (2018). *WaveNet Vocoder*. github.com/r9y9/wavenet_vocoder

Sagar Sharma. (2017). *What the Hell is Perceptron?*

Samson, T. (2011). *Concatenative Text-To-Speech System for Afaan Oromo Language*.

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions On*, *45*, 2673–2681. https://doi.org/10.1109/78.650093

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., & Wu, Y. (2018). Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *2018-April*, 4779–4783. https://doi.org/10.1109/ICASSP.2018.8461368

Sisamaki Eirini. (2019). End-to-End Neural based Greek Text-to-Speech Synthesis. *Unpublished Master's Thesis*.

Terfa., G. G. B. and K. (1982). Oromo Dictionary. *African Studies Center*.

Tesema, W., & Tamirat, D. (2017). *Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods*. www.pubicon.in

Thu, T., Nguyen, T., Prosodic, H. V. T., & Modeling, P. (2016). *To cite this version : Université Paris-Sud Specialty : Computer Science Thi Thu Trang NGUYEN HMM-based Vietnamese Text-To-Speech : Prosodic Phrasing Modeling , Corpus Design System Design , and Evaluation*.

Tilahun Gamta. (1994). *Afaan Oromo*. https://www.africa.upenn.edu/Hornet/Afaan_Oromo_19777.html

Tokuda, K., Zen, H., & Black, A. W. (2002). An HMM-based speech synthesis system applied to English. *Proceedings of 2002 IEEE Workshop on Speech Synthesis*, 227–230. https://doi.org/10.1109/WSS.2002.1224415

van den Oord, Aäron. (2016). *wavenet-generative-model-raw-audio*. https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

van den Oord, Aäron, Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel Recurrent Neural Networks. *CoRR, abs/1601.0*. http://arxiv.org/abs/1601.06759

Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., & Le, Q. (2017). Tacotron: Towards end-To-end speech synthesis. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, *2017-Augus*, 4006–4010.

https://doi.org/10.21437/Interspeech.2017-1452

Wardbaugh R. (1977). *Introduction to Linguistics*. McGraw-Hill.

Weweler Y. (2018). *SINGLE-SPEAKER END-TO-END NEURAL TEXT-TO-SPEECH SYNTHESIS*.

Winfred Philipp Lehmann. (1972). *Descriptive linguistics: An introduction* (R. House (ed.)).

Wosho, M. K. (2021). *Text to Speech Synthesizer for Afaan Oromo Using Hidden Markov Model*. *02*(03), 21–25.

Zegeye, A. T. (2010). *SCHOOL OF GRADUATE STUDIES FACULTY OF INFORMATICS DEPARTMENT OF INFORMATION SCIENCE SCHOOL OF GRADUATE STUDIES FACULTY OF INFORMATICS DEPARTMENT OF INFORMATION SCIENCE Signature of Board of Examiners for Approval*.

Zen, H., Oura, K., Nose, T., Yamagishi, J., Sako, S., Toda, T., Masuko, T., Black, A. W., & Tokuda, K. (2009). Recent development of the HMM-based speech synthesis system (HTS). *APSIPA ASC 2009 - Asia-Pacific Signal and Information Processing Association 2009 Annual Summit and Conference*, 121–130.

Zen, H., Senior, A., & Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 7962–7966. https://doi.org/10.1109/ICASSP.2013.6639215

APPENDICES

Appendix A. Questionnaire

The following questionnaire contains eleven items with a 1-5 scale out of this we used items such as voice pleasantness, naturalness, audio flow, ease of listening, and acceptance for measuring the Naturalness and items such as overall impression, listening effort, pronounciation, speaking rate, comprehension, and articulation are used to measure Intelligibility of the trained model.

NAME: _____ DEPT. _____ PHONE _____ DATE _____

Part I: Are you fluent in Afaan Oromo?

A. Yes B. No

Part II:

1. **Overall impression (Type I & Q): how do you rate the quality of the sound of what you just heard?**

□ Excellent □ Good □ Fair □ Poor □ V. Poor

2. **Listening effort (Type I): how would you describe the effort you were required to make in order to understand the message?**

□ Complete relaxation possible; no effort required

□ Moderate effort required

□ Considerable Effort required

□ Attention necessary; no appreciable effort required

□ No meaning understood with any feasible effort

3. **Pronunciation (Type Q): did you notice any anomalies in pronunciation?**

□ No □ Yes, annoying □ Yes, very annoying

□ Yes, but not annoying □ Yes, slightly annoying

4. **Speaking rate (Type Q) The average speed of delivery was:**

□ Just right □ Fairly fast or fairly slow □ Extremely fast or extremely slow

□ Slightly fast or slightly slow □ Very fast or very slow

5. **Voice pleasantness (Type Q): how would you describe the pleasantness of the voice?**

□ Very pleasant □ Fair □ Very unpleasant

□ Pleasant □ Unpleasant

6. **Naturalness: how would you rate the naturalness of the audio?**

□ Very natural □ Neutral □ Very unnatural

□ Natural □ Unnatural

7. **Audio flow: how would you describe the continuity or flow of the audio?**

□ Very smooth □ Neutral □ Very discontinuous

□ Smooth □ Discontinuous

8. Ease of listening: would it be easy or difficult to listen to this voice for a long period?

□ Very Easy □ Neutral □ Very Difficult

□ Easy □ Difficult

9. Comprehension problems (Type I): did you find certain words hard to understand?

□ Never □ Occasionally □ All of the time

□ Rarely □ Often

10. Articulation (Type I) Were the sounds in the audio distinguishable?

□ Very clear □ Neutral □ Much Less Clearer

□ Clear □ Less Clearer

11. Acceptance (Type I): do you think that this voice can be used for an interactive information system?

□ Yes □ No

Appendix B. Evaluation Sentence List

(**#1,** "Buna dhaabame lafa fagoo teenyee apilikeeshiniidhaan bishaan amma barbaadu kennaa kunuunsinas jedhan.")

(**#2,** "Hojjettoonnii Ministeera Dhimma Alaa Itiyoophiyaa Raayyaa Ittisa Biyyaaf qarshii miiliyoona 4.5 arjooman.")

(**#3,** "Ambaasaaddar Dammituu Hanbisaa Pireezedaantii Koree Nageenyaa Paarlaamaa Itilii waliin haala yeroo irratti marii'ataniiru.")

(**#4,** "Oomishni bunaa Itiyoophiyaatti heektaaraan hanga kuntaala 7 fi 8 argamaa jiru teeknolojjii kanaan gara kuntaala 30 olitti guddisuuf hojjataa jirra jedhan itti gaafatamaan ministeerri ministeera qonnaa.")

(**#5,** "Ministeerri Galiiwwanii meeshaalee ijaarsaaf tajaajilan waldaa tola oolaa Maqdooniyaaf deeggarsaan kenne.")

(**#6,** "Siifan fiigicha gulaallii meetira 1500 irratti har'a ganama yoo fiigdu kuftee ture. Yeroo sana sadarkaa 13ffaa irraa ka'uun 1ffaan xumurte.")

(**#7,** "Atileet Lammeechaa Girmaa Olompikii Tookiyootti adeemsifamaa jiruun meetira 3000 gufachiisa dhiiraan meedaaliyaa meetii argate.")

(**#8,** "Marii kanarratti Ministeerri Fayyaa Dr. Liyaa Taaddasaa, dureewwan dhaabbilee amantaa fi keessummoonni garaa garaa argamaniiru.")

(**#9,** "hojii, bashannana, hariiroo maatii fi namootaa fi kkf. walmadaalsisuu dhabuu.")

(**#10,** "jijjiirraa maallaqaatiin, $1 itiyoophiyaatti qarshii 42.")

Appendix B.1. Normalized Evaluation Sentences

(**#1,** "Buna dhaabame lafa fagoo teenyee apilikeeshiniidhaan bishaan amma barbaadu kennaa kunuunsinas jedhan.")

(**#2,** "Hojjettoonnii Ministeera Dhimma Alaa Itiyoophiyaa Raayyaa Ittisa Biyyaaf qarshii miiliyoona afur tuqaa shan arjooman.")

(**#3,** "Ambaasaaddar Dammituu Hanbisaa Pireezedaantii Koree Nageenyaa Paarlaamaa Itilii waliin haala yeroo irratti marii'ataniiru.")

(**#4,** "Oomishni bunaa Itiyoophiyaatti heektaaraan hanga kuntaala toorba fi saddeet argamaa jiru teeknolojjii kanaan gara kuntaala sodoma olitti guddisuuf hojjataa jirra jedhan itti gaafatamaan ministeerri ministeera qonnaa.")

(**#5,** "Ministeerri Galiiwwanii meeshaalee ijaarsaaf tajaajilan waldaa tola oolaa Maqdooniyaaf deeggarsaan kenne.")

(**#6,** "Siifan fiigicha gulaallii meetira kuma tokkof dhibba shan irratti har'a ganama yoo fiigdu kuftee ture. Yeroo sana sadarkaa kudha-sadaffaa irraa ka'uun tokkoffaan xumurte.")

(**#7,** "Atileet Lammeechaa Girmaa Olompikii Tookiyootti adeemsifamaa jiruun meetira kuma sadi gufachiisa dhiiraan meedaaliyaa meetii argate.")

(**#8,** "Marii kanarratti Ministeerri Fayyaa Dooktar. Liyaa Taaddasaa, dureewwan dhaabbilee amantaa fi keessummoonni garaa garaa argamaniiru.")

(**#9,** "hojii, bashannana, hariiroo maatii fi namootaa fi kan kana fakkaatan. walmadaalsisuu dhabuu.")

(**#10,** "jijjiirraa maallaqaatiin, doolaar tokko itiyoophiyaatti qarshii afurtamii-lama.")

Appendix C. Detailed Evaluation Results

Appendix C. 1. Detailed Objective Evaluation Results

Deep Voice 3 Result

Error Type	***Repeated words***	***Mispronunciations***	***Skipped words***
Sentence #1	2	2	0
Sentence #2	0	2	0
Sentence #3	0	1	0
Sentence #4	1	1	2
Sentence #5	0	1	0
Sentence #6	1	0	0
Sentence #7	0	0	0
Sentence #8	0	0	0
Sentence #9	1	1	0
Sentence #10	1	0	0
Total	6	8	2

Tacotron 2 ***Result***

Error Type	***Repeated words***	***Mispronunciations***	***Skipped words***
Sentence #1	0	0	0
Sentence #2	0	0	0
Sentence #3	0	0	0
Sentence #4	0	1	0
Sentence #5	0	0	0
Sentence #6	0	0	0
Sentence #7	0	0	0
Sentence #8	0	0	0
Sentence #9	0	1	0
Sentence #10	0	0	0
Total	0	2	0

Appendix C. 2. Detailed Subjective Evaluation Results

Deep Voice 3

Listeners / ***Evaluation***	***L1***	***L2***	***L3***	***L4***	***L5***	***L6***	***L7***	***MOS***
Naturalness	2.7	1.62	3.77	3.9	2.4	3.72	3.1	3.02
Intelligibility	3.1	2.2	3.7	3.75	3.4	3.7	3.2	3.28

Tacotron 2 with Linear spectrogram

Listeners / ***Evaluation***	***L1***	***L2***	***L3***	***L4***	***L5***	***L6***	***L7***	***MOS***
Naturalness	4.6	4.15	5	3.27	3.92	3.92	4.14	4.14
Intelligibility	4.3	4.6	4.8	3.75	4.7	4.3	3.8	4.32

Tacotron 2 with Mel spectrogram

Listeners / ***Evaluation***	***L1***	***L2***	***L3***	***L4***	***L5***	***L6***	***L7***	***MOS***
Naturalness	4.5	4.52	4.65	3.9	3.75	4	4.2	4.21
Intelligibility	3.8	4.75	4.5	4.1	4.5	4.1	4.5	4.32

Appendix D. Hyper-parameters used in training

Deep Voice 3	
"replace_pronunciation_prob": 0.0,	"converter_channels": 256,
"speaker_embed_dim": 16,	"query_position_rate": 1.0,
"num_mels": 80,	"key_position_rate": 1.385,
"fmin": 125,	"num_workers": 2,
"fmax": 7600,	masked_loss_weight": 0.5,
"fft_size": 1024,	"priority_freq": 3000,
"hop_size": 256,	"priority_freq_weight": 0.0,
"sample_rate": 22050,	"binary_divergence_weight": 0.1,
"preemphasis": 0.97,	"guided_attention_sigma": 0.5,
"min_level_db": -100,	"batch_size": 8,
"ref_level_db": 20,	"adam_beta1": 0.5,
"rescaling_max": 0.999,	"adam_beta2": 0.9,
"downsample_step": 4,	"adam_eps": 1e-06,
"outputs_per_step": 1,	"initial_learning_rate": 0.0005,
"embedding_weight_std": 0.1,	"nepochs": 2000,
"speaker_embedding_weight_std": 0.05,	"weight_decay": 0.0,
"padding_idx": 0,	"clip_thresh": 0.1,
"max_positions": 2048,	"checkpoint_interval": 1000,
"dropout": 0.050000000000000044,	"eval_interval": 10000,
"kernel_size": 3,	"save_optimizer_state": true,
"text_embed_dim": 256,	"window_ahead": 3,
"encoder_channels": 512,	"window_backward": 1,
"decoder_channels": 256,	"power": 1.4

Tacotron 2

```
#Mel spectrogram
num_mels = 80,
num_freq = 1025,
rescale = True,
rescaling_max = 0.999,
clip_mels_length = False,
max_mel_frames = 900,
use_lws=False,
silence_threshold=2,
n_fft = 2048,
hop_size = 275,
win_size = 1100,
sample_rate = 22050,
frame_shift_ms = None,
magnitude_power = 2.,
trim_silence = True,
trim_fft_size = 2048,
trim_hop_size = 512,
trim_top_db = 40,
signal_normalization = True,
allow_clipping_in_normalization = True,
symmetric_mels = True,
max_abs_value = 4.,
normalize_for_wavenet = True,
clip_for_wavenet = True,
wavenet_pad_sides = 1,
preemphasize = True,
preemphasis = 0.97,
min_level_db = -100,
ref_level_db = 20,
fmin = 55,
fmax = 7600,
power = 1.5,
griffin_lim_iters = 60,
GL_on_GPU = True,
outputs_per_step = 1,
stop_at_any = True,
batch_norm_position = 'after',
clip_outputs = True,
lower_bound_decay = 0.1,
#Input parameters
embedding_dim = 512,
#Encoder parameters
enc_conv_num_layers = 3,
enc_conv_kernel_size = (5, ),
enc_conv_channels = 512,
encoder_lstm_units = 256,
#Attention mechanism
smoothing = False,
attention_dim = 128,
attention_filters = 32,
attention_kernel = (31, ),
cumulative_weights = True,
#Decoder
prenet_layers = [256, 256],
decoder_layers = 2,
decoder_lstm_units = 1024,
max_iters = 10000,
#Residual postnet
postnet_num_layers = 5,
postnet_kernel_size = (5, ),
postnet_channels = 512,
#CBHG mel->linear postnet
cbhg_kernels = 8,
cbhg_conv_channels = 128,
cbhg_pool_size = 2,
cbhg_projection = 256,
cbhg_projection_kernel_size = 3,
cbhg_highwaynet_layers = 4,
cbhg_highway_units = 128,
cbhg_rnn_units = 128,
#Loss params
mask_encoder = True,
mask_decoder = False,
cross_entropy_pos_weight = 1,
predict_linear = True,
#Tacotron Training
tacotron_batch_size = 16,
tacotron_synthesis_batch_size = 1,
tacotron_test_size = 0.05,
tacotron_test_batches = None,
tacotron_decay_learning_rate = True,
tacotron_start_decay = 40000,
tacotron_decay_steps = 18000,
tacotron_decay_rate = 0.5,
tacotron_initial_learning_rate = 1e-3,
tacotron_final_learning_rate = 1e-4,
tacotron_adam_beta1 = 0.9,
tacotron_adam_beta2 = 0.999,
tacotron_adam_epsilon = 1e-6,
tacotron_reg_weight = 1e-6,
tacotron_scale_regularization = False,
tacotron_zoneout_rate = 0.1,
tacotron_dropout_rate = 0.5,
```