Addis Ababa University

College of Natural Sciences

Development of Text-to-Speech Synthesis Model for Afaan Oromoo using Transformer Neural Network

Bayisa Bedasa Abdisa

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

March, 2025

Addis Ababa University

College of Natural Sciences

Bayisa Bedasa Abdisa

Advisor: *Yaregal Assabie (PhD)*

This is to certify that the thesis prepared by *Bayisa Bedasa*, titled: *Development of Text-to-Speech synthesis Model for Afaan Oromoo using Transformer Neural Network* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor	Yaregal Assabie (PhD)	_____	_____
Examiner:	Dagmawi Lemma (PhD)	_____	_____
Examiner	Minale Ashagrie (PhD)	_____	_____

Abstract

Text-to-speech (TTS) is a process of converting written text into spoken words. It analyzes the incoming text, processes linguistic data, and produces audio output using algorithms. TTS systems are widely utilized in applications such as virtual assistants, accessibility tools for individuals with visual impairments, and language learning software. Afaan Oromoo is a Cushitic language mostly spoken in Ethiopia and other parts of Africa and serves as an essential means of communication for the Oromo people. For Afaan Oromoo, developing a TTS system is essential for enhancing accessibility and promoting the use of the language in digital environments. This study focuses on a transformer-based neural network model technique for Afaan Oromoo TTS. The model architecture comprises an encoder-decoder structure. The encoder processes input text by converting it into a contextualized representation, while the decoder generates speech waveforms from this representation. We enhanced the model with multi-head attention mechanisms to capture long-range dependencies in the input text, improving prosody. Additionally, we employed a HiFi-GAN-based vocoder for converting the model's output into high-fidelity audio waveforms, enhancing the overall quality of the synthesized speech. Utilizing the transformer architecture, the implementation is carried out in Python. We have produced 17 hours of audio dataset and their corresponding text transcription from the Afaan Oromoo speech corpus by a male speaker. The transformer-based text-to-speech synthesis architecture has outperformed the previously done model based on BLSTM-RNN for Afaan Oromoo language TTS, whose results are 3.77 and 3.76 in terms of intelligibility and naturalness, respectively. We used the Mean Opinion Score (MOS) to assess naturalness and intelligibility subjectively. Experimental results indicate that our transformer-based TTS system achieved a MOS score of 4.21 for naturalness and 4.23 for intelligibility, reflecting a commendable performance level. Our model also enables prosody modeling with user input parameters to generate deterministic speech, positioning it as a state-of-the-art solution.

Keywords: Afaan Oromoo, Text-to-speech, Transformer neural network, speech synthesis, end-to-end architecture

Dedication

I can't put into words how grateful I am to each and every one of you. Your unwavering support, encouragement, and belief in me have been the driving force behind my academic journey. Mom, Dad, brothers, sisters and Sabboontuu (my wife), you have been my pillars of strength, instilling in me the values of hard work, resilience, and determination. Your sacrifices and the countless ways you have nurtured my dreams have been the foundation upon which I have built my success.

Acknowledgements

I would like to express my gratitude to the following individuals, without whom I could not have finished this thesis or successfully completed my master's degree!

I want to start by giving thanks to my Lord Jesus Christ for giving me this chance. I am really appreciative of my advisor, Dr.Yaregal Assabie, whose insight and knowledge of the subject matter steered me through this thesis. Special thanks to Dr. Feda Negesse for providing me with useful support and guidance on Afaan Oromoo linguistics, which has been invaluable in refining my ideas and improving the quality of this thesis. Also, many thanks to the teachers of Addis Ababa University's MSc in Computer Science, who, regardless of the difficult situations, did their best to make me feel confident enough for my next professional steps. I extend my heartfelt thanks to my parents and brother for their unwavering love, understanding, and encouragement. Their belief in me and constant motivation have been the driving force behind my academic pursuits. The completion of this thesis would not have been possible without the collective support and contributions of all those mentioned above. I am truly grateful for their invaluable assistance and encouragement, and I humbly dedicate this work to them. Lastly, I would like to extend my gratitude to all the volunteers who helped evaluate our models.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms and Abbreviations

AI	Artificial Intelligence
Bi-LSTM	Bidirectional-Long Short-Term Memory
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EATS	End-to-End Adversarial Text-to-Speech
FFN	Feed-forward Neural Network
G2P	Grapheme-to-Phoneme
GPU	Graphical Processing Unit
HMM	Hidden Makrov Model
IPA	International Phonetic Alphabet
LSTM	Long Short-Term Memory
ML	Machine learning
MOS	Mean Opinion Score
NLP	Natural Language Processing
NMT	Neural Machine Translation
RNN	Recurrent Neural Network
SPSS	Statistical Parametric Speech Synthesis
TTS	Text-to-Speech
VAE	Variational Auto-Encoder
VITS	Variational Inference Text-to-speech

CHAPTER ONE: INTRODUCTION

1.1. Background

Language is the capacity to convey ideas through a collection of sounds, gestures, and signs (written). Being the only animals that employ such a system, it is a characteristic that sets humans apart. Afaan Oromoo is one of the languages that have huge speakers in the Horn of Africa, with millions of native speakers. It is also one of the under resourced languages, like other Ethiopian languages. However, there is a lack of digital resources and tools specifically designed for the language. The absence of an efficient Text-to-speech (TTS) system in Afaan Oromoo limits individuals' ability to access digital content, engage with technology, and fully participate in various aspects of society in their native language. Therefore, the development of an efficient and high-quality TTS system for Afaan Oromoo is crucial to bridge this gap and empower the community. Speech is the most common means of human expression, and speech technology is rapidly becoming the most prevalent mode of knowledge delivery today. For example, speaking to a computer is more natural than pressing buttons. Building a strong and reliable infrastructure is essential since text-to-speech serves as a link between people and machines.

Text-to-speech (TTS) synthesis, also known as "speech synthesis," converts input texts into speech by using them as a source. Converting written normal language text into spoken words is the goal of the machine learning job known as text-to-speech synthesis. It encompasses numerous fields, including computer science, linguistics, digital signal processing, acoustics, etc. Regarding information processing [1], it is a cutting-edge technology, especially for the current intelligent speech interaction systems. The TTS synthesis process consists of two main steps. The first uses text analysis to transform the input text into a phonetic or other linguistic representation; the second uses this phonetic and prosodic information as the output to create speech waveforms. Voice synthesis can be defined as the process of creating human speech artificially. The term voice synthesizer refers to a computer system used for this purpose. TTS plays a vital role in the teaching and learning process, in assisting those who are visually impaired or in situations where reading is not a priority to listen to printed materials, in telecommunications, and in industries.

The speech synthesis methods can be divided into two main groups: (1) deep machine learning-based methods and (2) conventional machine learning-based methods. Concatenative voice synthesis [2] and parametric speech synthesis [3] are two particular techniques utilized for TTS in classical machine learning. However, deep learning-based voice synthesis techniques give priority to speech naturalness. To create a TTS system for Afaan Oromoo, we have explored a unique model structure that combines the effective parallelizable training of CNNs with the efficient inference of RNNs. This approach utilizes Transformer Neural Networks, hybrids of both CNN and RNN techniques for improved voice synthesis techniques. Recent years have seen a rapid breakthrough in text-to-speech (TTS) due to the development of deep learning. Tan et al. [4] describe how neural network-based TTS has evolved from CNN/RNN-based models [5, 6] to transformer-based models [7]. Transformers rely on attention mechanisms instead of operating on sequences step-by-step like RNNs to capture relationships between all input and all output tokens. The transformer network raises the obvious question of why self-attention is better than recurrent or convolutional models.

The study by A. Vaswani et al. [8] titled "Attention is All You Need" introduced Transformers as a new architecture, which has been investigated as a potentially more effective replacement for existing neural networks and states three factors account for the preference: (a) computational complexity of each layer, (b) concurrency, and (c) path length between long-range dependencies. Speech synthesis is one of the many Natural Language Processing (NLP) tasks in which Transformer has shown to be incredibly effective. To create a TTS system that produces genuine and understandable speech, we intend to capture the linguistic subtleties, intonations, and phonetic qualities unique to Afaan Oromoo by training a Transformer-based model on a sizable dataset of Afaan Oromoo speech samples.

Transformers were first pioneered in the seminal work of Vaswani et al. [8] titled “Attention Is All You Need” for machine translation tasks in NLP and have recently shown impressive performance in other domains, including speech processing. In contrast to RNNs, which have trouble capturing long-range dependencies among the input sequence, transformer models employ self-attention layers to do so. Furthermore, self-attention allows for more parallelization compared to RNNs, as these can process the speech sequence as a whole without relying on past states to capture dependencies. The Transformer model will learn to

map input text-to-speech output by analyzing the patterns in the dataset. In the testing stage, the trained model will be used to synthesize speech from written text. The input text will be preprocessed to extract features that are relevant for speech synthesis. The Transformer model will then use these features to generate speech that closely matches the natural speech patterns of Afaan Oromoo.

1.2. Motivation

After Arabic and Swahili, Afaan Oromoo is the third most spoken language in Africa, with more than 40 million speakers in Ethiopia (about 40% of the population) [9], Kenya, Sudan, Somalia, and Djibouti [10]. Afaan Oromo (meaning the Oromo language) is rated the second most widely used language among the Indigenous languages of Africa [11]. There is a rich cultural history and significance to the Afaan Oromoo language. These considerations highlight the necessity, from the point of view of the study, for an Afaan Oromoo TTS that could be of high quality, lightweight, and free. The creation of an efficient TTS system for the language can help preserve and advance the language, enabling future generations to access and engage with their linguistic and cultural traditions. The TTS system for Afaan Oromoo can enable the community of speakers of that language to communicate more successfully in their mother tongue and to express themselves. However, the efficiency of TTS synthesis for Afaan Oromoo is limited, hindering the accessibility and usability of digital content for the language-speaking population.

The motivation for this study is to see Afaan Oromoo recognized as a universal language in communication, transportation, education, and technology, specifically through the development of TTS synthesis aimed at improving intelligibility and naturalness. Despite being one of the most widely spoken languages in the Horn of Africa [12], Afaan Oromoo has faced challenges in gaining visibility within modern technological frameworks [13]. By leveraging transformer-based models known for their effectiveness in NLP, we aim to enhance the intelligibility and naturalness of Afaan Oromoo in TTS applications. This integration is crucial for preserving cultural identity and empowering speakers, particularly in educational contexts. Ultimately, our work seeks to elevate Afaan Oromoo's status in the digital landscape, promoting inclusivity and diversity in technology while ensuring that synthesized speech is both intelligible and close to natural for users.

1.3. Statement of the Problem

From formant-based methods [14], which are based on the source-filter model of speech, to concatenative-based methods [15, 16], which are based on breaking down the spoken sentence into different words and syllables within it, the ongoing advancement of TTS techniques has significantly improved the intelligibility and naturalness of synthesized speech. There are different approaches used in concatenative synthesis based, some are unit selection-based waveform cascade methods [17], diphone-based, domain-specific, and the hidden Markov model (HMM)-based, also known as statistical parametric speech synthesis (SPSS) methods [18].

In recent years, deep learning (DL) has emerged as a new field of study within machine learning. This method can effectively capture the latent information and association in data and has more powerful modeling ability than traditional statistical learning methods [19]. TTS methods based on deep learning have been widely researched [20]. For example, in the SPSS model based on Deep Neural Networks (DNN), DNN can learn the mapping function from linguistic features (input) to acoustic features (output). DL-based models have gained significant progress in many fields, such as handwriting recognition [21], machine translation [22], speech recognition [23], and speech synthesis [24]. To address the problems existing in speech synthesis, many researchers have also proposed DL-based solutions and achieved great improvements. Deep learning techniques are now widely used in TTS systems due to their ability to generate higher-quality synthesized speech than traditional methods [25]. Previously, Ethiopian researchers have been able to develop a prototype TTS synthesizer model for some regional languages such as Amharic, Afaan Oromoo, Wolaytta and Tigrigna, respectively [15, 16, 26, 27] Moreover, the researcher's work considers the traditional machine learning-based techniques, which is necessary to propose deep learning techniques.

Although several works have been conducted in the area of TTS synthesis for technologically favored languages for many years [13], every language has distinct phonetic systems and differs in prosodic characteristics, which makes creating a TTS model unique. So, speech synthesizer systems developed for one language cannot be directly applied to another language because the structures of one language are not presumably representative

of others. Every program is built on the framework that corresponds to a particular language's phonetic rules. Besides, the existing TTS synthesizer for Afaan Oromoo was reviewed in this study, and still, their performance needs a lot of improvement in terms of intelligibility and naturalness using novel deep learning approaches. Here, naturalness means that the system should sound just like a human. Intelligibility, on the other hand, will be defined as the listeners' ability to properly decode and hence understand the message from the produced speech.

For this language with numerous speakers, there have been only a few attempts made, as far as the knowledge of the researchers is concerned, to develop a TTS for Afaan Oromoo [13]. A variety of solutions were put forth to address such problems, including the application of concatenative techniques like unit selection or parametric [13, 28]. But they needed a great deal of hard work and domain knowledge. Another reason for such poor performance of Afaan Oromoo speech synthesizers is the lack of speech corpora, unlike English, which has many publicly available corpora and audiobooks [29]. Some of the limited Afaan Oromoo TTS researches include Morka Mekonnen [30] made the first attempt to create TTS for Afaan Oromoo by employing the diphone speech units approach. Nevertheless, prosodic features those that appear when humans combine sounds in connected speech were overlooked in that study, and the created prototype's performance suffered greatly as a result. Similar to this, Samson Tadesse [16] used a diphone and triphone-based technique to create a TTS synthesizer for Afaan Oromoo. However, there are a lot of shortcomings, such as false information resulting from concatenation point discontinuities. Due to the lengthy training and recording process, the produced system performed poorly in terms of naturalness and intelligibility. Muhidin Kedir [28] made another attempt to utilize statistical parametric speech synthesis based on HMM techniques. It is chosen for this research because it is a model-based approach that requires less storage, learns properties of data rather than stores the speech, has a small runtime, and is easy to integrate with small handheld devices.

Thus, in reference to the Afaan Oromoo language from earlier research (literature examined), the Transformer Neural Network methodology has not yet been systematically investigated for TTS synthesis. Soressa Beyene and Raja.K [31] used the deep learning approach with the bidirectional long short-term memory (BLSTM)-based RNN (recurrent

neural network). However, because both the current input and the prior hidden state are necessary to construct the current hidden state, RNNs can only consume the input and produce the output sequentially. The characteristic of a sequential process limits the parallelization capability in both the training and inference processes. For the same reason, for a certain frame, information from many steps ahead may have been biased after multiple recurrent processing.

Transformer Vaswani et al. [8] suggest replacing the RNNs in NMT models in order to address these two issues. RNN-based models require sequential training and inference for every sample, but CNN-based models allow for parallel training. Considering the training time, CNN takes less time than RNN [6]. Transformer overcomes this by employing self-attention in both its encoder and decoder, whereas RNN and CNN-based models have trouble learning dependencies [7]. Compared to RNNs, which can only process tiny portions of a sentence at a time, transformers have the advantage of analyzing the complete sentence at once when analyzing speech. This is made possible by the unique self-attention-based architecture of transformers [8], which enables them to learn long-term dependencies, which is critical for speech-processing tasks. Moreover, the multi-head attention mechanism [32] is a specialized feature in transformers that allows for more efficient parallelization during training, making them ideal for handling large datasets, which is a common challenge in speech synthesis tasks. This unique combination of self-attention and multi-head attention empowers transformers to achieve exceptional performance in sequence-to-sequence modeling, making it an indispensable tool in the field of speech processing.

Although TTS technology has advanced significantly in recent years, much more can be done to better capture the meaning and style of the original text. The main motive for this research is to enhance the naturalness of generated speech from text. While numerous studies have explored the performance of TTS models, such as Tacotron 2 [5], there remains a need to investigate the impact of architectural conditioning [5] and auxiliary task learning on the quality and expressiveness of generated speech. This paper proposes the development of a TTS synthesizer for the Afaan Oromoo language using transformer network deep-learning speech synthesizer approaches. Therefore, this research was initiated to improve the efficiency of the TTS synthesizer for the language. As far as the researchers' knowledge is concerned, there were no attempts made to develop TTS for Afaan Oromoo using the

Transformer technique. Transformers are particularly effective in capturing spatial and temporal patterns in data and have been successfully applied to tasks such as neural machine translation (NMT) and other speech processing. CNN and Transformer-based TTS [7] can speed up the training over RNN-based models since there is no recurrent connection between frames. Our baseline RNN-based TTS model is Tacotron 2 [5], an RNN-based TTS model.

Hence, this research aims to come up with a TTS synthesis for the Afaan Oromoo language that generates natural-sounding and intelligible speech, which is vital for many application areas.

1.4. Objectives

General Objective

The general objective of this thesis is to develop a text-to-speech synthesis model for Afaan Oromoo by using transformer neural network.

Specific Objectives

The specific objectives of this research are:

- To review the literature on the area of text-to-speech synthesis and Afaan Oromoo speech characterization.
- To prepare Afaan Oromoo text-to-speech <text, audio> pairs corpus.
- To design a text-to-speech synthesis model for the Afaan Oromoo language.
- To develop the prototype of the system
- Evaluate the performance of the developed system with different data sets.

1.5. Methodology

To achieve the study objective with the Afaan Oromoo text-to-speech synthesis system, the following methods will be employed:

Literature Review

A literature review is carried out in the field of text-to-speech systems to gain a clear understanding of the work involved and to comprehend the types of techniques that

researchers are using for Afaan Oromoo speech synthesis as well as for other languages. A literature review is conducted to identify the strengths and limitations of the current voice synthesis approaches through a systematic process of searching, selecting, critically evaluating, and synthesizing relevant research articles, papers, books, and other sources. The deep learning approach to speech synthesis is also examined, along with the many deep learning-based approaches and their benefits and drawbacks, so that the most effective way can be selected from those developed via our research.

Data Collection

Both the appropriate text and the wav files are required in order to train a TTS model. One of the crucial building blocks for any NLP application is data. A language's resource availability is one of the criteria used to assess its resource level. For resource-rich languages such as English, German, French, Spanish, etc., the size of the dataset is not a problem because researchers have created a large set of corpora and tools for many NLP tasks [29]. Nonetheless, a large number of other languages are considered low-resource languages [33]. With this intuition, Ethiopian languages such as Amharic [34], Afaan Oromo [35], Tigrinya [36] and Wolaytta [37] lack data resources, linguistic materials, and tools, making them "low resource" languages. This has an impact on how various NLP tasks and tools were developed. The primary goal of the data collection activity is to create a dataset that can be utilized for testing and training to create the intended system. Using a corpus technique, speech and text are gathered concurrently. We will gather a substantial amount of high-quality Afaan Oromoo text-to-speech pair corpus from the standardized dataset. Since our approach is deep learning, data is required to accomplish the work. The dataset will consist a total of 17 hours of recorded speech data that corresponded to sentences collected from legitimate sources such as Afaan Oromoo books, Afaan Oromoo text-to-speech prior conducted research corpus, Afaan Oromoo Holy Bible, and Afaan Oromoo media like Oromia Broadcasting Network (OBN), VOA Afaan Oromoo, and other Afaan Oromoo news to train the transformer model. We can reuse existing Afaan Oromoo TTS approaches as their importance, including those for data preparation that will be the best for the transformer model.

Prototype Development

We will develop a prototype for Afaan Oromoo language TTS synthesis that converts text into speech using a deep learning Transformer-based model specifically tailored for the language TTS synthesis. The model will learn the mapping between input text sequences and corresponding acoustic features, enabling it to generate natural and intelligible speech.

Evaluation

The subjective evaluation technique is used to determine a TTS system's quality. The resulting audio quality is described by the subjective evaluation that is carried out in terms of intelligibility and naturalness. Intelligibility is the capacity for the reader to understand the original information, and the audio's quality is taken into consideration inside the understanding standard. The quality of the generated speech is characterized as naturalness. Language naturalness encompasses characteristics including timing, emotion, and pronunciation. Subjective measures of naturalness and intelligibility were used to assess the effectiveness of the suggested system. For the subjective assessment, the Mean Opinion Score (MOS) is employed.

1.6. Scope and Limitations of the Study

This work gives the first insight into the possibility of developing a text-to-speech system for Afaan Oromoo text using the Transformer model. However, it has targeted simple Afaan Oromoo sentences. In addition, the choice of the word embeddings used is constrained by the limited computational resources that were available to us. The size of the corpus is also small in comparison to high-resource languages like English and Chinese.

1.7. Applications of the Results

The result of this thesis proposal will contribute to the ongoing digital transformation in this domain area. The Afaan Oromoo Language plays a significant role in benefiting society in several ways:

- Customer Services: The technology will also find applicability in systems such as banking, telecommunications (automatic system voice output), transport, Internet

portals, accessing PCs, emailing, administrative and public services, cultural centers, and many others.

- Accessibility: TTS enables individuals with mild or moderate visual impairments or reading difficulties to access written information. By converting text into spoken words, TTS allows people who are blind or have low vision to listen to books, articles, websites, and other textual content. It promotes inclusivity and equal access to information, education, and entertainment.
- Multilingual Support: TTS systems can support multiple languages, allowing users to convert text into speech in different languages. This feature is particularly useful for language learners, travelers, or individuals who need to communicate in languages they are not familiar with.
- Scalability: TTS technology enables the efficient and rapid conversion of large volumes of text into speech. This scalability is advantageous for applications such as automated voice response systems, audiobook production, podcasting, and any scenario where generating speech from text in real-time or in bulk is required.
- Language Preservation: TTS can contribute to the preservation and revitalization of endangered or minority languages. By enabling the conversion of written text into spoken words, TTS helps in the dissemination and usage of such languages, preventing them from becoming extinct.
- Assistive Technology: TTS is utilized in various assistive technologies, including screen readers, voice assistants, navigation systems, and communication devices for individuals with disabilities. It empowers users to interact with devices, access information, and communicate effectively.
- Cost and Time Efficiency: TTS systems can automate speech generation, reducing the need for manual voice recordings. This saves time and resources, as creating human-recorded speech for every text input can be time-consuming and expensive.
- For people who have literacy difficulties: Often getting frustrated trying to browse the internet happens to some people who have basic literary levels. By giving them a

choice to hear the text instead of reading it, they can acquire valued information in a way that is easier for them.

- Educational application: TTS assists students with learning disabilities, such as dyslexia, by reading aloud the text, helping them comprehend and absorb information more effectively. TTS also aids language learners in improving their pronunciation and fluency by listening to correctly spoken words and sentences.
- Enhances Efficiency and Productivity: TTS enables individuals to listen to written content while performing other tasks, such as driving, walking, making dinner, exercising, or multitasking. By converting text-based information into speech, TTS reduces the time and effort required to read, enabling people to consume information more conveniently. This is very important for people with limited reading and writing abilities, or for those who prefer listening over reading.
- Voice-overs and Media Production: TTS technology is utilized in media production, including voice-overs for videos, movies, and animations. It offers a cost-effective and time-saving solution for generating spoken content. TTS can simulate different voices and accents, providing flexibility and creativity to content creators.

1.8. Organization of the Rest of the Thesis

The remainder of this thesis is structured as follows: The theoretical underpinnings of text-to-speech, pronunciations analysis, history of speech synthesis, and Afaan Oromoo linguistic characteristics are covered in Chapter 2. Chapter 3 provides an overview of the relevant earlier research on text-to-speech. The suggested methodology architecture for this investigation, the discussion of the components and their interaction is described in the fourth Chapter. The experiment conducted and the outcomes are shown in Chapter 5. Last but not least, Chapter 6 presents conclusion of our work and some directions for future research work.

CHAPTER TWO: LITERATURE REVIEW

2.1. Introduction

Natural language processing (NLP), a sub-discipline of artificial intelligence (AI), is emerging among the best researched and rapidly developing practices across various domains (Kalyanathaya et al., 2019). TTS is a subset of NLP, an area of study that attempts to close the communication gap between humans and machines. Thus, the present study is concerned with, among the fundamental subjects that require research, NLP for Afaan Oromoo. In the current digital era, where the use of technology resources has sharply increased, to expand the language's widespread usage.

Before undertaking an exploration of deep learning, it is crucial to distinguish between AI, ML and DL. We understand that AI and human intelligence are not a substitute for one another [38]. While AI can analyze and perform specific tasks for the customers, it lacks the full range of cognitive abilities, emotional understanding, and complex problem-solving that characterize human intelligence. Through the combination of AI and human intelligence, businesses can create complete solutions as well as elevate customer experience [38]. Human intelligence, which is found in both humans and animals and incorporates emotionality and consciousness. ML is a subset of artificial intelligence that empowers computer systems to learn and improve through experience and data without explicit programming. Basically, DL is a subset of machine learning techniques that employes representational learning in artificial neural networks.

Extracting additional features is the core advantage of using DL that it helps us to. With more features and the capacity to handle massive volumes of data simultaneously, we can perceive objects similarly to how humans do. As previously mentioned, TTS is one of the top-listed applications used in DL. This kind of application takes advantage of sequential data and DL. Time-series data or the need to comprehend natural language are examples of sequential data. Since the preceding word or feature in this case depends on the subsequent feature, it is called sequential data. For example, what time is it? If we let state "it is," for example, "what time is it over here?" it will be included in the sentence. Additionally, being

aware of past events is essential to understand or establish a connection. So the technology used to accomplish this known as RNNs.

This study is based on the principle that attention is fundamental because it enhances the model's ability to understand and generate close to human speech by allowing it to focus on relevant information in a flexible and efficient manner. A research paper titled "Attention Is All You Need" [8] proposes the transformer architecture. These models can be enhanced by implementing an "attention mechanism." An attention mechanism informs the network which components of the input vector it should focus on in order to generate the intended output [8]. By utilizing the attention mechanism, a transformer model can evaluate one input word while also considering the relevant information present in other input words. Furthermore, non-verbal words are hidden by attention mechanisms.

Transformers are a kind of neural network architecture that transformed NLP and many other fields by introducing a novel mechanism called self-attention [8]. Unlike traditional RNNs which process sequences in order, transformers can process input data in parallel, which brings significant efficiency and scalability [7]. The main idea of self-attention is that the model learns to identify which words or tokens in the input sequence are more important relative to one another when generating an output, to capture long-range dependencies effectively [7]. Especially for tasks like translation, text generation, and TTS, this capability is helpful where context is crucial in understanding meaning and coherence. Besides, Transformers can support both variable-length inputs and outputs, which makes them very suitable regarding different applications [7]. Transformers architecture is designed with encoder layers and decoder layers, where the encoder takes in the input and processes the input data and the decoder does the output generation [8]. In general, Transformers have become the backbone of many state-of-the-art NLP applications [29].

Both LSTM models and RNN models are encoder and decoder networks that process input data at different time steps, with the encoder network having the responsibility of generating an encoded representation of the words in the input data. The encoder network initially collects the input sequence and a hidden state from the previous part of that sequence at each step, then updates the hidden state values as the data moves through the network until the last moment, at which point a "context vector" is updated. Then, the decoding network

receives the context vector and uses it to predict which word is most likely related to the input word at each step to generate the target sequence.

RNNs process inputs sequentially, with each input word changing and maintaining a hidden state vector as it passes through the network. An RNN's hidden states usually don't include a lot of pertinent information about the inputs that came before it. The present state is frequently overwritten by new inputs, which results in information loss and gradually lowers performance. Unlike other models, transformer models, on the other hand, handle the input sequence in full at once. Because the attention method allows each output word to be influenced by every input and hidden state, the network becomes more reliable for longer text passages.

Long Short-Term Memory (LSTMs) are an enhanced RNN variant designed to manage longer input sequences. The LSTM design contains a "gates" structure that includes "input gates," "output gates," and "forget gates" [21] . The gated architecture addresses the information loss that RNN models frequently encounter. LSTM models can be trained using parallel computing, but the architecture is recurrent and this causes training time to drag along with it. Data is still processed sequentially. Since attention processes were recognized to improve the model's performance, LSTM developers often incorporated them into the network. Ultimately, though, it was found that accuracy may be enhanced by the attention mechanism alone. The synergy between GPU technology and transformer has facilitated the development of transformer networks that utilize attention mechanisms and enable parallel processing [39].

2.2. History of Speech Synthesis Technology

Beginning in the 12th century, attempts have been made to construct machines capable of synthesizing human speech [40]. In the second half of the 18th century, the Hungarian scientist Wolfgang von Kempelen constructed a speaking machine with a series of bellows, springs, bagpipes, and resonance boxes to produce some simple words and short sentences [41]. In the later part of the 20th century, the first computer-based speech synthesis system was released [40]. The early computer-based speech synthesis methods include articulatory synthesis [42], formant synthesis [14], and concatenative synthesis [43]. Later, with the development of statistics ML, statistical parametric speech synthesis (SPSS) is proposed

[44], which predicts parameters such as spectrum, fundamental frequency, and duration for speech synthesis. Since the 2010s, neural network-based speech synthesis [24] has gradually become the dominant method and achieved much better voice quality.

Articulatory Synthesis: Articulatory synthesis [42] produces speech by simulating the behavior of human articulators such as the lips, tongue, glottis, and moving vocal tract. Ideally, articulatory synthesis can be the most effective method for speech synthesis since it is the way humans generate speech. However, modeling these articulator behaviors in practice is quite challenging. For instance, collecting data for articulator simulation is challenging [45]. Because of this, articulatory synthesis usually results in speech of lower-quality than later formant and concatenative synthesis.

Formant Synthesis: Rule-based or formant speech synthesis [14] produces audible output depending on a specified rules that control a simplified source-filter model. To mimic the formant structure and other spectral properties of speech as closely as possible, these rules are usually developed by linguists. With varying parameters like fundamental frequency, voicing, and noise levels, the speech is synthesized by an additive synthesis module and an acoustic model. The formant synthesis can produce highly intelligible speech with moderate computation resources that are well-suited for embedded systems and do not rely on large-scale human speech corpus as in concatenative synthesis. However, the synthesized speech has artifacts and sounds less natural. Moreover, it is difficult to specify rules for synthesis.

Concatenative Synthesis: Concatenative synthesis [43] relies on the concatenation of pieces of speech that are stored in a database. Usually, the database consists of speech units ranging from whole sentences to syllables that are recorded by voice actors. In inference, the concatenative TTS system searches speech units to match the given input text and produces speech waveforms by concatenating these units together. In general, concatenative TTS can usually produce audio that is very intelligible and has an accurate timbre that is close to the actual voice actor. A slight drawback of this concatenative approach is the requirement of a powerful computational method and sufficient storage. As memory cost has dropped, it is possible to increase the size of the speech database to increase the quality of the speech, as well as improve the computational algorithm used in the system. As concatenation can result

in less smoothness in stress, emotion, prosody, etc., the generated voice is less natural and emotional, which is another drawback.

Statistical Parametric Speech Synthesis (SPSS): to address the drawbacks of concatenative TTS, SPSS is proposed [44]. The basic idea is that instead of directly generating a waveform through concatenation, we can first generate the acoustic parameters [46] that are necessary to produce speech and then recover speech from the generated acoustic parameters using some algorithms [47]. A text processing module, a parameter prediction module (acoustic model), and a vocoder analysis/synthesis module (vocoder) are the usual three components of SPSS. The text analysis module first processes the text, including text normalization [48], grapheme-to-phoneme conversion [49], word segmentation, etc., and then extracts the linguistic features, such as phonemes, duration, and POS tags, from different granularities. Acoustic features include fundamental frequency, spectrum, or cepstrum [46], etc., are extracted from the speech through vocoder analysis [47]. Vocoder analysis are among the coupled linguistic features and parameters (acoustic features) are used to train the acoustic models (e.g., hidden Markov model (HMM)-based). The vocoders are an important part of neural speech synthesis systems because they help close the gap between the abstract representation of speech (acoustic characteristics) and the actual audio output. SPSS has several advantages over previous TTS systems: 1) naturalness the audio more natural; 2) flexibility is convenient to modify the parameters to control the generated speech; 3) reduced data cost compared to concatenative synthesis, it requires less recording data. But SPSS has disadvantages as well: 1) Artifacts like muffled, buzzing, or noisy audio make the generated speech less intelligible; 2) The generated voice is still robotic and easily distinguishable from human recording speech. In the near 2010s, as neural networks and deep learning have achieved rapid progress, some works first introduced deep neural networks into SPSS, such as deep neural networks (DNN)-based [24] and recurrent neural networks (RNN)-based [31]. While using neural networks instead of HMMs, these models use the SPSS model to predict acoustic features by taking into account their linguistic features. Later, Wang et al. [50] introduce to directly produce acoustic features from phoneme sequence rather than linguistic features, which may be considered as the first attempt for end-to-end (encoder -> vocoder together) speech synthesis. In this thesis, we mainly focus on an end-to-end models and neural-based speech synthesis. We provide a

brief overview of these models without going into more detail because later SPSS also uses neural networks as the acoustic models.

Figure 1: Common Structure of TTS Systems [45]

As shown in Figure 1, a modern TTS system consists of three basic components: a text analysis module, an acoustic model, and a vocoder [45]. The text analysis module converts a text sequence into linguistic features, the acoustic models generate acoustic features from linguistic features, and then the vocoders synthesize waveforms from acoustic features.

Neural Speech Synthesis: to address the drawbacks of SPSS TTS, while some early neural models are adopted in SPSS to replace HMM for acoustic modeling, neural network-based TTS, or neural TTS in short, is introduced with the improvement of deep learning, using (deep) neural networks as the model backbone for speech synthesis. A further proposal, WaveNet [51], which might be considered the first modern neural TTS model, is proposed to directly produce waveforms from linguistic features. Other models, like DeepVoice 2 [52], still follow the three components in statistical parametric synthesis but upgrade them with the corresponding neural network-based models. Additionally, various end-to-end models (e.g., Tacotron 2 [5], Deep Voice 3 [53], and FastSpeech 1 [54]) are proposed to simplify text analysis modules. These models use character/phoneme sequences as input and simplify acoustic features with mel-spectrograms.

Later, fully end-to-end TTS systems are developed to directly generate waveform from text, such as ClariNet [55], FastSpeech 2 [56], and EATS [57]. Neural network-based speech synthesis offers the advantages over previous TTS systems that relied on concatenative synthesis and statistical parametric synthesis, including superior voice quality in terms of naturalness and intelligibility, as well as less requirement on human preprocessing and feature development.

2.3. Overview of Afaan Oromoo Language

Unlike Amharic (an official language of Ethiopia), which belongs to the Semitic language family, Afaan Oromoo, so-called Oromiffaa or Afan Oromo, is an indigenous member of the Cushitic branch of the Afro-Asiatic language family. Next to Arabic and Hausa, Afaan Oromoo is the third most widely spoken language in Africa [12]. Most parts of Ethiopia and some areas of the neighboring countries speak it, making it one of the native African languages (Mekuria, 1994; Mohammed, 1994). Since it is widely spoken and used in Kenya (Mekuria, 1994) it ranked as one of the major indigenous African languages, it is the official language of the Oromia Regional State, which is geographically the biggest region among the current federal states in Ethiopia. Much of what is now Ethiopia, Somalia, Sudan, northern Kenya, and some areas of other East African nations comprise its original homeland (Abera, 1988). Afaan Oromoo is used by the Oromo people, who are the largest ethnic group in Ethiopia, which amounts to 50% of the total population in 2007 (2015 Census statistic of Ethiopia) [58].

Afaan Oromoo Writing System

The Latin-based alphabet known as Qubee, has been used as the official script of Afaan Oromoo, writing system since 1991 [59]. This writing system shares many features with English, incorporating all 26 English letters and adding 7 combined consonant letters (CH, DH, SH, NY, TS, PH, ZY) known as double consonants [48]. While the vowels ‘a’, ‘e’, ‘i’, ‘o’, and ‘u’ are the same as in English, they are pronounced differently and can be either short or long vowel, affecting word meanings. For instance, “lafa” (ground) versus “laafaa” (soft). Doubling consonants also changes pronunciation, as seen in “dammee” (sweety) versus “damee” (branch). The alphabet excludes ‘p’, ‘v’, and ‘z’ for native words but uses them for foreign terms like “polisii” (police).

Table 1: The Afaan Oromoo letters in International Phonetic Alphabet (IPA) [60]

Qubee (capital and small letter)		IPA	Qubee (capital and small letter)		IPA	Qubee (capital and small letter)		IPA
A	a	/a/	L	l	/l/	W	w	/w/
B	b	/b/	M	m	/m/	X	x	/t’/
C	c	/č’/	N	n	/n/	Y	y	/w/
D	d	/d/	O	o	/o/	Z	z	/z/
E	e	/e/	P	p	/p/	CH	ch	/č/
F	f	/f/	Q	q	/k’/	DH	dh	/đ/
G	g	/g/	R	r	/r/	NY	ny	/ň/
H	h	/h/	S	s	/s/	PH	ph	/p’/
I	i	/i/	T	t	/t/	SH	sh	/š/
J	j	/ǧ/	U	u	/u/	TS	ts	/s’/
K	k	/k/	V	v	/v/	ZH	zh	/ž/

Afaan Oromoo Word Structure and Boundaries

The basic word order of Amharic and Afaan Oromoo languages is subject-object-verb (SOV) [12]. For instance, in the Afaan Oromoo sentence “Caaltuun bilisa baate," “Caaltuun “is a subject, “bilisa” is an object, and “baate” is a verb. The conversion of the sentence into English is looks like “Caaltuu has got freedom,” which has an SVO structure. This means Afaan Oromoo and English are different in sentence structuring.

Adjectives follow a noun or pronoun, and their normal position is close to the noun they modify in Afaan Oromoo. English, adjectives usually precede nouns. For example, namicha gaarii (good man), gaarii (adj.) follows namicha (noun). In Afaan Oromoo, like in other languages, the blank character (space) shows the end of one word. Moreover, parenthesis, brackets, and quotes are used to show a word boundary. Furthermore, sentence boundary

punctuation is almost similar to the English language, i.e., a sentence may end with a period (.), a question mark (?), or an exclamation point (!) [16].

Afaan Oromo, like many Latin-based languages, uses whitespace characters to separate words, facilitating word segmentation. The language is highly organized, with roots that are basic pronounceable sounds. Words are formed by combining these roots with prefixes and suffixes, known as forms, to convey specific meanings. For instance, the root /Bar/ combined with /-e/ forms “Bare” (learned), and with /-te/ forms “Barte” (she learns). This morphological structure allows Afaan Oromo to express a wide range of verb nuances, showcasing its linguistic richness and complexity.

Table 2: The Afaan Oromoo conjugating System

Person	Past			Present			Continuous			Perfect		
		Phonetic Transcription	English meaning		Phonetic Transcription	English meaning		Phonetic Transcription	English meaning		Phonetic Transcription	English meaning
1st	Deeme -me	De:me	I went	Nideema Ni- -a	Ni.de:ma	I go	Deemaara -aara	De:ma:ra	I was going	Deemeera -eera	De:me:ra	I gone
	Deemne -ne	De:m.ne	we went	Nideemna Ni- -na	Ni.de:m.na	We go	Deemaarra -aarra	De:ma:r.a	We was going	Deemneerra -neerra	De:m.ne:r.a	We gone
2nd	Deemte -te	De:m.te	you went (single)	Nideemta Ni- -ta	Ni.de:m.ta	you go (single)	Deemaarta -aarta	De:ma:r.ta	You was going (single)	Deemteerta -teerta	De:m.te:r.ta	You gone (single)
	Deemtan -tan	De:m.ta.n	you went (multiple)	Nideemtu Ni- -tu	Ni.de:m.tu	You go (multiple)	Deemaartu -aartu	De:ma:r.tu	You was going (multiple)	Deemtaniirtu -taniirtu	De:m.ta.ni:r.tu	You gone (multiple)
3rd	Deeme -me	De:me	he went	Nideema/ti Ni- -a/ti	Ni.de:ma/.ti	he/she go	Deemaara/Deemaarti -aara/-aarti	De:ma:ra/.ti	he/she was going	Deemeera/ Deemteerti -eera/-teerti	De:me:ra/de:m.te:r.ti	he/she gone
	Deeman -man	De:ma.n	they went	Nideemu Ni- -u	Ni.de:mu	They go	Deemaaruu -aaruu	De:ma:ru	They was going	Deemaniiru -maniiru	De:ma.ni:ru	they gone

Afaan Oromoo Punctuation Marks

A series of symbols known as punctuation serves to make a piece of written text more understandable. Except for the apostrophe mark (‘), punctuation marks in the Afaan Oromoo and English are same and serve the same function. The Afaan Oromoo uses the apostrophe mark (‘) to indicate a glitch (called hudhaa) sound in writing, while in English it indicates possession. It is an important part of the Afaan Oromoo reading and writing system. For instance, it is used to write words that have usually two vowels appear together, like “kaa’e” to mean “putting down," except some words like “ja’a” to mean "six,” which is identified from the sound created. Sometimes an apostrophe mark (‘) in Afaan Oromoo is substituted with the spelling “h.” For instance, “ja’a” and “kaa’e” can be interchanged used by the letter "h," like "jaha” and “kaahe” respectively, without changing the senses of the word. The following are some of the most often used punctuation signs in Afaan Oromoo, much like in English.

Full stop (.) is used at the end of a sentence and also in abbreviations.

Question mark (?) is used in interrogative or at the end of a direct question.

Exclamation mark (!) is used at the end of commands and exclamatory sentences.

A Comma (,) is used to separate listings in a sentence or items in a list of three or more.

A Semicolon (;) is used to join two closely related independent clauses.

Colon (:) is used to separate and introduce lists, clauses, and quotations, along with several conventional uses, and etc.

2.4. Text-to-Speech Synthesis

Text Analysis

Text analysis/text processing is one of the most influential and complex tasks of the NLP module that contains rich information about pronunciation and prosody to ease speech synthesis. It analyzes the syntax and semantics of the text and is where the input text is transcribed into a phonetic or some other linguistic representation. In other words, the computer can identify words, sentences, and pronunciations and knows how to pronounce them and when and how long to pause.

Statistic parametric synthesis employs text analysis to derive a set of linguistic feature vectors [44] and encompasses various features such as text normalization [61], word segmentation [62], prosody prediction, and grapheme-to-phoneme conversion [63]. The end-to-end neural TTS is facilitated by the large modeling capacity of neural-based models, which means that character or phoneme sequences are used as input for synthesis, simplifying the text analysis module. Text normalization is necessary to obtain standard word format from character input, and grapheme-to-phoneme conversion is needed to get phonemes from standard words. Despite claims that full end-to-end synthesis can generate waveform from text in certain TTS models, it is still necessary to use text normalization for handling non-standard formats of raw text. In addition, some end-to-end TTS models include traditional text analysis features. By using neural networks, DeepVoice 2 [52] incorporates character-to-linguistic feature conversion into its pipeline, while some works explicitly anticipate prosody features with text encoders. To summarize some typical tasks in text analysis:

Text Normalization: It converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. The arbitrary text is an unrestricted set of characters which may consist of different numbers (e.g. X, VI, 0,2.1), abbreviations such as Dr., Ar., Er., Mr., Mrs., Prof., and/or special characters such as #, &, etc. and/or currency($, €, £, ₾), upper and lowercase, punctuation(!,?,“ ”), date(01/06/1989), mathematical expressions(=, >, <,~, %), acronyms, etc. Normalization means transforming the input text into a series of pronounceable words. For example, before normalization: “1989” After normalization: “nineteen eighty-nine," before normalization: “Jan. 24." After normalization: “January twenty-fourth." After that, the final text is transformed into phonemes. Not all letters are similar in their phoneme codes. The purpose of phoneme-to-speech conversion is to produce speech signals based on phonemes. Following the early work on rule-based [48], word recognition, neural networks were used to model a sequences-to-sequence process in text identification where non-standard words were identified as source and target sequence.

Text Segmentation: For character-based languages such as Afaan Oromoo, word segmentation [62] is necessary to detect the text boundary from input text, which is

necessary to ensure the accuracy for later grapheme-to-phoneme conversion process and prosody generation.

Character-to-Phonetic / Grapheme-to-Phoneme (G2P) Conversion: Converting text (grapheme) into pronunciation (phoneme) can greatly ease speech synthesis. This is also called phonetic analysis, which refers to the conversion of the linguistic description in orthographic form to phonemes. Following word normalization, this component is in charge of the text-to-phoneme conversion for the target language. The input consists of the text structure from the preceding component, to which the normalized word pronunciations are added (Hasim, 2000). For instance, the word “speech” is converted into “s p iy ch." A manually collected character-to-phoneme lexicon is usually used for conversion. However, for alphabetic languages like Afaam Oromoo, the lexicon cannot cover the pronunciations of all the words. Thus, the G2P conversion for Afaan Oromoo is mainly responsible for generating the pronunciations of out-of-vocabulary words.

Prosody Generation: The prosody information, such as rhythm, stress, and intonation of speech, corresponds to the variations in syllable duration, loudness, and pitch, which play an important perceptual role in human speech communication. To identify each type of prosody, prosody generation uses tagging methods. The methods and procedures for prosody tagging vary specific to the language.

Text analysis is one step in the development of the Afaan Oromo language TTS model, and it relies on the sum of tokenization and normalization. Tokenization is the process of breaking sequences of sentences into their constituent words. During tokenization, the special characters and white space delimiter (like, @ ', x?, …) are the main focus areas where, whenever there exist the mentioned delimiters between characters in a sentence, sequences of characters are broken to produce a meaningful word for a given specific language. Example, “Isheen nama gaarii dha” (She is a good person) is tokenized as "isheen," "nama," "gaarii, “dha. The white space is used as one of the delimiters that is mentioned often and creates an utterance boundary; here then, the script was written to break a token whenever there is a white space in the source input sentence and continues until the sentence is finished. Though the white space is the most commonly used delimiter between words and is extensively used for tokenization, it has some limitations; a token type that

allows the occurrence of whitespace within the token will not be recognized as a single token but will be split up into two or more tokens. Example, consider the telephone number “0999 99 99 99." This can identify as a single token of type ‘telephone number', but if tokenization is exclusively based on whitespace, then we end up having 4 tokens.

Acoustic Models

The acoustic models that produce acoustic information from linguistic features or directly from phonemes or characters are reviewed here. Sequence-to-sequence models based on encoder-attention-decoder frameworks (e.g., LSTM, CNN, and self-attention) [5, 7, 53], the most recent feed-forward networks (CNN or self-attention) [54] for parallel generation, and the early HMM and DNN-based models in statistical parametric speech synthesis (SPSS) [24, 31] are among the acoustic models that have been adopted with the construction of TTS.

Producing acoustic features that are further converted into waveforms using vocoders is the aim of an acoustic model. The types of TTS pipelines are largely determined by the choice of an acoustic feature. Various kinds of acoustic features, such as mel-cepstral coefficients (MCC), mel-generalized coefficients (MGC) [46], band aperiodicity (BAP), fundamental frequency (F0), voiced/unvoiced (V/UV), bark-frequency cepstral coefficients (BFCC), and the most widely used mel-spectrograms, have been tried. There are two time periods of an acoustic model: 1) neural-based end-to-end TTS acoustic models, which use phonemes or characters to predict acoustic features like mel-spectrograms, and 2) SPSS acoustic models, which usually use linguistic features to predict acoustic features.

Using neural-based end-to-end TTS acoustic models (such as CNN, RNN, and Transformer (self-attention)) instead of SPSS models (like HMM, DNN, or RNN) may have the following advantages: 1) Compared to traditional acoustic models, which necessitate alignments between linguistic and auditory data, sequence-to-sequence-based neural models are more end-to-end and require less preprocessing. These models implicitly learn the alignments through attention or jointly forecast the duration. 2) As neural networks' modeling capabilities improve, linguistic features are reduced to only character or phoneme sequences, but acoustic features have evolved from low-dimensional and condensed

cepstrum (like MGC) to high-dimensional mel-spectrograms or even more high-dimensional linear spectrograms.

RNN-Based Models: RNN-based models (like the Tacotron Series) Tacotron [64] employs a framework of encoder-attention-decoder, which accepts characters as input and outputs linear spectrograms and uses the Griffin-Lim vocoder to produce waveforms. Using an extra WaveNet [51] model, Tacotron 2 [5] is added to create mel-spectrograms and transform them into a waveform. It greatly advances the quality of voice over previous methods (i.e., concatenative TTS, parametric TTS, and neural TTS such as Tacotron).

CNN-Based Models: CNN-based models (like, DeepVoice Series) DeepVoice [65] is a convolutional neural network-enhanced SPSS system. DeepVoice uses a WaveNet [51]-based vocoder to produce waveforms after acquiring linguistic information using neural networks. Following DeepVoice's fundamental data conversion mechanism, DeepVoice 2 [52] improves on DeepVoice by adding multi-speaker modeling and better network architecture.

Additionally, DeepVoice 2 also uses a Tacotron + WaveNet model pipeline, which first uses Tacotron to generate linear spectrograms and next uses WaveNet to generate waveforms. DeepVoice 3 [53] leverages a fully convolutional network structure for speech synthesis, which generates mel-spectrograms from characters and can scale up to real-world multi-speaker datasets. DeepVoice 3 outperforms earlier DeepVoice 1/2 systems by using a more condensed sequence-to-sequence model and directly predicting mel-spectrograms rather than complex linguistic features.

Another researcher, ClariNet [55], explores the entirely end-to-end generation of waveforms from text. One completely convolutional-based non-autoregressive model that can generate mel-spectrograms more quickly while still achieving passably high speech quality is ParaNet [66]. To create mel-spectrograms from character sequences, the Discrete Diffusion Model with Contrastive Learning for Text-to-Speech Generation (DCTTS) [6] uses a fully convolutional-based encoder-attention-decoder network and a data conversion pipeline that is similar to Tacotron's. After that, it generates linear spectrograms using a spectrogram super-resolution network and uses Griffin-Lim to create waveforms.

Transformer-Based Models: Transformer-based models (like the FastSpeech Series) Transformer TTS [7] generates mel-spectrograms from phonemes using an encoder-attention-decoder architecture based on Transformer [8]. Two issues are alleged to exist with Tacotron 2 and other RNN-based encoder-attention-decoder models: 1) because the RNN-based encoder and decoder cannot be learned in parallel and the encoder cannot be parallel in inference, the recurring nature affects the efficiency of both training and inference. 2) RNNs struggle to model the lengthy dependencies in speech and text sequences since they are typically very lengthy.

Transformer TTS incorporates several of Tacotron 2's ideas, including stop-token prediction and decoder pre-net/post-net, while maintaining Transformer's core model structure. Despite having a quicker training period, it produces a voice quality that is comparable to Tacotron 2. However, the transformer's encoder-decoder attention is not as resilient as RNN-based models like Tacotron, which use stable attention methods like location-sensitive attention. This is because the transformer uses parallel processing. Therefore, some works suggest improving transformer-based acoustic models' resilience. For example, MultiSpeech [55] increases the robustness of the attention mechanism by using decoder bottleneck, encoder normalization, and diagonal attention constraint, whereas RobuTrans [56] increases the resilience in autoregressive generation by using duration prediction.

There are several issues with previous neural-based acoustic models that employ autoregressive generation, such as Transformer TTS [7], DeepVoice 3 [53], and Tacotron 1/2 [64, 5]. Those problems are: 1) Inference speed is slow. Particularly for lengthy speech sequences, the autoregressive mel-spectrogram generation is slow (for example, if the hop size is 10 ms, a 1-second speech is a long sequence, with almost 500 frames of mel-spectrogram). 2) Strong issues. Since encoder-attention-decoder-based autoregressive generation relies on faulty attention alignments between text and mel-spectrograms, the generated speech typically has a lot of word skipping and repeating problems. FastSpeech [54] is therefore suggested as a solution to these problems: 1) It can significantly speed up inference by generating mel-spectrograms in parallel using a feed-forward Transformer network. 2) To improve robustness and prevent word skipping and repetition, it removes the attention mechanism that separates speech and text.

Vocoders

To speak roughly, the development of vocoders may be classified into two stages: the vocoders used in statistical parametric speech synthesis (SPSS) [67] and the neural network-based vocoders [51]. In the development of text-to-speech (TTS) systems for the Afaan Oromoo language, vocoders play a crucial role in converting text into natural-sounding speech. A technique that uses audio signal encoding and decoding to create synthetic human speech is a vocoder, short for "voice coder." In the context of TTS transformer models, vocoders like HiFi-GAN are often employed to generate high-quality speech waveforms from spectrograms. These vocoders are integrated into the TTS pipeline to enhance the naturalness and intelligibility of the synthesized speech.

For Afaan Oromoo, a language with diverse dialects and unique phonetic characteristics, the use of advanced vocoders is essential. The transformer-based TTS models utilize vocoders to handle the complex acoustic features of the language. These models incorporate a flow-based module that predicts spectrogram-based acoustic features, which are then decoded into speech waveforms using a stack of transposed convolutional layers, similar to the HiFi-GAN vocoder.

The integration of vocoders in TTS systems for Afaan Oromoo ensures that the synthesized speech is not only intelligible but also natural-sounding, making it more effective for applications such as language education, accessibility tools for the visually impaired, and automated customer service systems. The continuous improvement of these vocoders and their adaptation to the specific needs of Afaan Oromoo speakers is a promising step towards more inclusive and accessible speech technology.

2.5. Pronunciation Analysis

Prosodic Analysis

The prosody of a language is one of the parts of speech synthesis systems that depend on the nature of the language. It refers to the extraction of supra-segmental features such as intonation, stress, and timing from written text. These features are considered to be the melody, rhythm, and emphasis of the speech at the perceptual level. The intonation tries to estimate how the pitch pattern or fundamental frequency changes during speech. The term

prosody refers to certain properties of the speech signal that are related to audible changes in pitch, loudness, and syllable length. The following can be mentioned as factors contributing to the prosodic features: feelings (anger, happiness, and sadness), speaker characteristics (gender, age, and dialects), the meaning of the sentence (neutral, imperative, and question), and fundamental frequency (duration and stress).

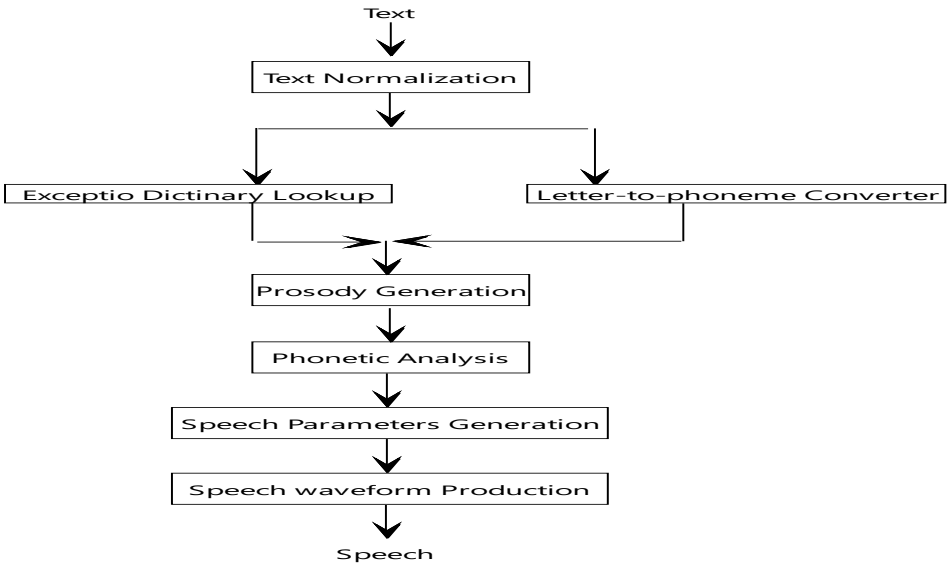

Figure 2: Diagram of TTS the conversion process [68]

Language Specific Issues

There are many challenges for text-to-speech systems. As high-quality TTS synthesis became more and more popular, researchers began to analyze the impact on society of such technologies. Certain text-to-speech systems require large databases and complex modeling that can increase cost production. The usability is also a challenge. Speech can be understood, but it is nonetheless restricted by its lack of emotional focus. Stereotypical views of synthesizers are that they sound robotic and overall are inefficient to be introduced into societal practices.

Another challenge to text-to-speech synthesis is pronunciation, which occurs when the system is "reading." Although some languages, like Afaan Oromoo, contain regular pronunciation rules that have a relatively consistent letter-to-sound correspondence, other languages, like English, contain many irregular pronunciations. For example, the English

pronunciation of the phoneme /f/ will differ when referring to the word “of,” in which /f/ is pronounced more like /v/. These irregular pronunciations can also be discovered in the alternate spelling of “fish” as “ghoti.” The /gh/ indicates the ending of the word “tough,” the /o/ is pronounced similarly to the /o/ in “women,” and finally the /ti/ is spoken like in the word “action.”

Pronunciation of numbers is also problematic. There are various ways to pronounce the number 3421. While the simple synthesis of “sadii afur lama tokko” may be practical for reading social security numbers or sequences of numbers, it is simply not practical for all occasions. One occasion can refer to the number “kuma sadiifi dhibba afuriifi digdamii-tokko,” while another may need to imply the address of a house such as “soddomii-afur digdamii-tokko.”

Other pronunciation hazards are common in the form of abbreviations and acronyms. Some abbreviations, such as the units for inches (in.), form a word in itself, relying on the system to know the correct understanding of when the proper pronunciation must be used. Acronyms cause databases to become greatly complex. As in the case of the pronunciation of the YKN, it is simply pronounced as the word “’y’, ’k’, ’n’,” not by the pronunciation of the letters “Y,” “O,” “O,” “K,” “A,” “A,” and “N.”

As it is simple to make synthetic speech in some languages but not in others. The quantity of resources available for creating voice synthesis is also impacted by the fact that different countries and languages have very varied markets and prospective users. The majority of languages also have certain unique characteristics that can either greatly simplify or greatly complicate the development process.

A language's prosody is one of the aspects of speech synthesis systems that is dependent on its nature. A language's prosody mostly refers to the way words are pronounced. To provide more prosodic and expressive control over unit selection TTS for dialog applications while retaining naturalness, we have focused on speech acts, the communicative function of an utterance. The current working set of speech acts being used include: imperative (directive, request, wait, repeat, and warning); interrogative (question-wh, question-yes/no, and question-multiple choices); assertive (informative-general, informative-detail); affective (apology, exclamation-positive, exclamation-negative, greeting, good-bye, thanks); and

others (confirmation, disconfirmation, back-channel, cue phrase). Prosody is, in fact, the key element of the Afaan oromoo TTS system. A shift in emphasis can alter a word's meaning in a sentence. This emphasis can manifest as a change in timing, pitch, gemination, or voice quality. This is a place where the intonation and durations of every diphone or triphone unit of any word will be tracked. In any language, a word may differ contextually because of its dependent nature or because there is a punctuation mark attached. For example, “Sirrii dha” is equivalent to saying “It is right,” but whenever the punctuation mark ‘?’ is put with it, like “Sirrii dhaa?” The previous normal sentence is changed to “Is it right?” which is a question type.

CHAPTER THREE: RELATED WORK

3.1. Introduction

This Chapter will explore various speech synthesis techniques that work for both local and foreign languages. TTS synthesis has shown substantial improvement in recent years, particularly with the development of transformer-based models. These models have revolutionized natural language processing by leveraging self-attention mechanisms, resulting in high-quality, context-aware speech. While much research has concentrated on widely spoken languages, there is still a significant gap in TTS systems for under-represented languages such as Afaan Oromoo. In this part we analyzes the existing literature on TTS methodology for both local and international languages, focusing on classical approaches, statistical models, and recent deep-learning techniques, specifically Transformers. We investigate the challenges and opportunities involved with synthesizing speech in Afaan Oromo, emphasizing the significance of linguistic traits and cultural variations in constructing an efficient TTS system.

3.2. A Comprehensive Overview of Pre-Deep Learning TTS Techniques: Concatenative and Statistical Parametric Synthesis

Pre-deep learning speech synthesis methods requ0ire extensive domain expertise and may contain brittle design choices. This may be time-consuming and require a lot of human resources. On the other hand, for Ethiopian languages in general, there are few traditional speech synthesis methods research works that can be categorized into concatenative systems (Mezgebe Araya [27], Samson Tadese [16], Tewodros Abebe [26], and Alula Tafare [69]. They developed a concatenative-based, also known as corpus-based approach, a TTS synthesizer for Tigrigna, Afaan Oromoo, Wolaytta, and Amharic languages, respectively) and parametric systems (Muhidin Kedir [28] proposed a new TTS using statistical parametric speech synthesis called also corpus-based approach based on HMM).

In another study, Mahlet Awel [70] carried out a bidirectional long-short-term memory-based TTS synthesis for the Amharic language. The author collected 600 sentences size of an audiobook of the Amharic Bible read by a male Amharic speaker online from YouTube, and the recorded audio Bible has relatively good recording conditions. Before evaluation

training, testing, and validation were held using the collected corpus. Of the sum of 600 sentences, 90% have been used to train the neural network model, 5% sentences will be used for testing, and 5% for validation purposes. The experimentation with a listening test by 10 volunteers gave a score in MOS of 3.9 for naturalness and 3.8 for intelligibility to the researcher BLSTM model, whereas 3.7 for naturalness and 3.65 for intelligibility to the researcher DNN model, and the MCD of BLSTM and DNN is 4.68 and 4.7, respectively. However, this approach can lead to longer training times and increased resource requirements. While it is true that utilizing High-Performance Computing (HPC) can enhance the performance of BLSTMs, the architectural strengths and efficiency of transformers often make them the preferred choice for many applications, especially in NLP and tasks that require handling large corpuses [71].

Soressa Beyene and Raja.K [31] explores speech synthesis for the Afaan Oromoo language using a deep learning method based on a BLSTM-RNN model. The authors created a corpus of 1000 sentences and matched them with transcriptions of Afaan Oromoo speech by a female speaker; from the dataset domains, 700 sentences were trained and 300 sentences were tested. The MOS evaluation technique was used to subjectively test the performance of synthesized speech by the evaluator of native speech to listen to the audio recorded within their test. The results of a subjective test using the thirteen sentences were 3.76 and 3.77 out of 5 for intelligibility and naturalness, respectively. Objective evaluation is where the measurement of speech quality performance is approximated by applying appropriate speech in waveforms. When recording audio at 16 kHz to create the MCD BLSTM based on RNN, the total average process of Mel cepstral distortion (MCD) at Mel cepstral coefficients is 3.89, and the Merlin wave produced is around 3.71. In contrast, it processes data sequentially, which makes it less efficient for parallel processing and requires high memory to store the hidden states from both the forward and backward directions at each time step. Finally, the researcher recommended future work on an effective acoustic feature extraction model for the fast-speed processing of large datasets.

3.3. Transformer Neural Network Based Speech Synthesis

In 2017, Vaswani et al. [8] introduced the transformer TTS concept. In 2019, Li et al. [7] proposed a novel end-to-end TTS model. To train the transformer network model, the authors used 4 Nvidia Tesla P100 GPUs with an internal US English female dataset, containing 25-hour professional speech (17584 <text, audio> pairs, with a few too-long waves removed). The study based on the Transformers network replaced the conventional recurrent neural networks and original attention protocols with a multi-head attention mechanism and the Tacotron2 framework. With this modification, they were able to build hidden states in both the encoder and the decoder in tandem, which greatly increased training efficiency. Here, the original RNN structures in the encoder and decoder as well as the attention were substituted with a multi-head. To represent the frame relationship in a variety of ways, it can divide attention into subspaces. Long sentences are better for learning because the resulting sample sounds more natural. Long-range dependency problems can also be successfully resolved by the self-attention mechanism, which makes it easier for inputs to interact with one another at different times. Using phoneme sequences, their Transformer TTS network generates mel-spectrograms, which are then processed by a WaveNet vocoder to generate the final audio outputs. Those MOS tests are rigorous and reliable, as each audio is listened to by at least 20 testers, who are all native English speakers (compared to Tacotron2's 8 testers in Shen et al. [5]), and each tester listens to less than 30 audios. The effectiveness and efficacy of the Transformer neural network are evaluated through experiments. When it comes to efficiency, the Transformer TTS network can train 4.25 times faster than Tacotron 2. According to thorough human testing, the Transformer model operates at a level that is highly comparable to human quality (4.39 in MOS) and outperforms Tacotron2 with a gap of 0.048 in CMOS, achieving state-of-the-art performance.

Karita et al. [71] experimentally compared transformers with conventional RNNs. Based on the results, they showed various training and performance benefits achieved with transformers in comparison to RNN-based TTS using two corpora: M-AILABS [72] (Italian, 16 kHz, 31 hours) and LJSpeech [73] (English, 22 kHz, 24 hours). A single Italian male speaker (Riccardo) was used in the case of M-AILABS. They conclude that transformer-

based TTS achieves similar loss convergence compared to RNN-based TTS in the case of both corpora (M-AILABS and LJSpeech). Loss convergence refers to the point where the model's training loss stabilizes. The validation loss values are provided for both Tacotron2 (an RNN-based TTS model) and Transformer models. Lower validation loss typically suggests better performance. It indicates that the model is better at minimizing the difference between the predicted speech features (such as mel-spectrograms) and the ground truth features during the training process. In the case of M-AILABS, the Transformer model achieves a lower validation loss (0.320 on 1 GPU and 0.316 on 3 GPUs) compared to Tacotron2 (0.329 on 1 GPU). In the case of LJSpeech, the Transformer model has a slightly higher validation loss (0.398 on 3 GPUs) compared to Tacotron2 (0.390 on 1 GPU).

Ren et al. [54] train the Transformer TTS model on 4 NVIDIA V100 GPUs, with a batch size of 16 sentences on each GPU. They assessed the performance of transformers in comparison to Tacotron 2. The researcher experiments with the LJSpeech [73] dataset, which comprises 13,100 English audio samples and the corresponding text transcripts, with an entire audio duration of approximately 24 hours. They assess the audio quality and use the MOS evaluation method on the test dataset. Each audio is assessed with a minimum of twenty testers, who are all fluent English speakers. They contrasted the MOS of the generated audio samples by the Transformer model with the Tacotron 2 model. The results indicate that in terms of audio quality, the assessment metric provided suggests that Transformer TTS with a score of 3.88 ± 0.09 is better than Tacotron 2 with a score of 3.86 ± 0.09.

The Tacotron system, introduced by Wang et al. [64], was one of the first models to demonstrate an end-to-end trainable neural TTS system. It achieved substantial improvements in the naturalness of the speech, but it had no explicit control over prosody. The authors trained the model on US English, which contains about 24.6 hours of speech data spoken by a professional female speaker. The model achieves a MOS of 3.82 score, which outperforms the 3.69 MOS of the parametric (based on LSTM) system in terms of naturalness. In a recent study [74], the subtle prosodic characteristics of Dutch sarcastic speech were examined. According to sentence-level analysis, the findings showed certain prosodic indicators for sarcasm, such as reduced vocal loudness, decreased intensity and longer duration in comparison to ground truth speech. It’s noteworthy to remark that the use

of pitch and duration varied depending on the sort of statement and the gender of the speaker.

The recently introduced model VITS (Variational Inference with adversarial learning for end-to-end text-to-speech) [75] employs an end-to-end architecture based on conditional variational auto-encoders (VAEs) [76] and dynamic programming for real-time monotonic alignment search. The proposed model demonstrates exceptional performance in generating contextual speech with varied pitches and rhythms by sampling latent vectors stochastically, although this method does not allow for control over these aspects. One-stage TTS systems, utilizing direct transformation approaches, simplify the conversion of text to synthetic voice through end-to-end neural network topologies, such as sequence-to-sequence models with attention mechanisms, allowing for direct conversion of raw text into auditory waveforms. To improve the rhythm of generated utterances, VITS incorporates a duration predictor, enabling a variety of utterances for multilingual speakers. The authors conducted experiments on two different datasets; the LJ Speech dataset [73], which has a single speaker and produced a MOS of 4.43, and the VCTK (Voice Cloning Tool Kit) dataset, which has multiple speakers and achieved a MOS of 4.38. The results demonstrate that their approach outperforms the best publicly available TTS systems comparable to ground truth.

3.4. Summary

In this Chapter, we reviewed various speech synthesizers developed using different techniques, including formant, concatenative, and parametric approaches such as statistical parametric and Bi-LSTM RNN methods. Unlike recurrent neural networks (RNNs) or convolutional neural networks (CNNs), transformers do not impose a specific structure on the relationships between inputs, both temporally and spatially. The studies reviewed demonstrate that researchers have explored various methods for building TTS synthesis systems for multiple languages, including Afaan Oromoo. Several researchers have employed different techniques to develop TTS for Afaan Oromoo, aiming to enhance intelligibility, naturalness, and sound quality. Previous studies highlight that varying techniques can significantly improve performance. However, the existing TTS systems for Afaan Oromoo still face intelligibility and naturalness which are common challenges in TTS areas. Thus, we strive to improve TTS for the under-resourced Afaan Oromoo language to

make it more intelligent and natural. We will accomplish this by examining the advantages of the transformer-based deep learning technique, which can process multiple words simultaneously than RNN and solves the problem that CNN cannot capture long-distance dependent while also carefully considering the prosody of the language. Additionally, due to the unique structure and characteristics of Afaan Oromoo, adapting TTS systems from other languages is not straightforward. Therefore, we propose leveraging a deep learning transformer approach to achieve better results in Afaan Oromoo speech synthesis.

CHAPTER FOUR: DESIGN OF TTS SYTHESIS FOR AFAAN OROMOO

4.1. Introduction

In this Chapter, we present the design and architecture of our Transformer-based model tailored for Afaan Oromo language processing. Building upon the strengths of the Transformer architecture, which has demonstrated remarkable success in various NLP tasks, our model aims to address the unique linguistic characteristics and challenges associated with Afaan Oromo. We will outline the key components of the model, including the input representation, attention mechanisms, and decoding strategies. Additionally, we will discuss the modifications made to accommodate the specific prosody features of Afaan Oromo, ensuring that the model not only learns effectively but also generates high-quality, contextually relevant TTS. By integrating these design considerations, we aim to contribute to the advancement of TTS technology in underrepresented languages and enhance accessibility for Afaan Oromo speakers.

Additionally, we describe the design of Afaan Oromoo TTS synthesis using speech segments of recorded voice to convert text into speech, which can be saved as audio files in formats such as WAV or MP3. We begin by illustrating the generalized model of the proposed system, highlighting the architecture and workflow involved in the synthesis process. Following this, we provide a detailed description of each component within the model, including their roles and interactions. Moreover, we address the training procedures, inference procedures, and the prosody prediction mechanisms employed in the model. This comprehensive overview aims to elucidate the systematic approach taken in developing an effective TTS system for the Afaan Oromo language, leveraging the capabilities of transformer-based architectures.

The architecture is made up of an encoder-decoder structure, in which the encoder converts the input text into several contextual embeddings that accurately represent the linguistic features of the Afaan Oromoo language. With the rich morphological and phonetic traits of Afaan Oromoo, this is very significant. The model may dynamically focus on different areas of the input text thanks to the decoder's multi-head attention methods, which generate audio

waveforms from these embeddings. We include a prosody prediction to further improve the synthetic speech's naturalness.

4.2. System Architecture

This Section addresses the proposed technique and its architecture. As illustrated in Figure 3, our approach for Afaan Oromoo TTS conversion comprises of components: In this area, we will provide comprehensive explanations of the components used in our framework. The suggested model's overall architecture includes, among other things, text input, text normalization, prosody modeling, model, vocoder, and waveform output.

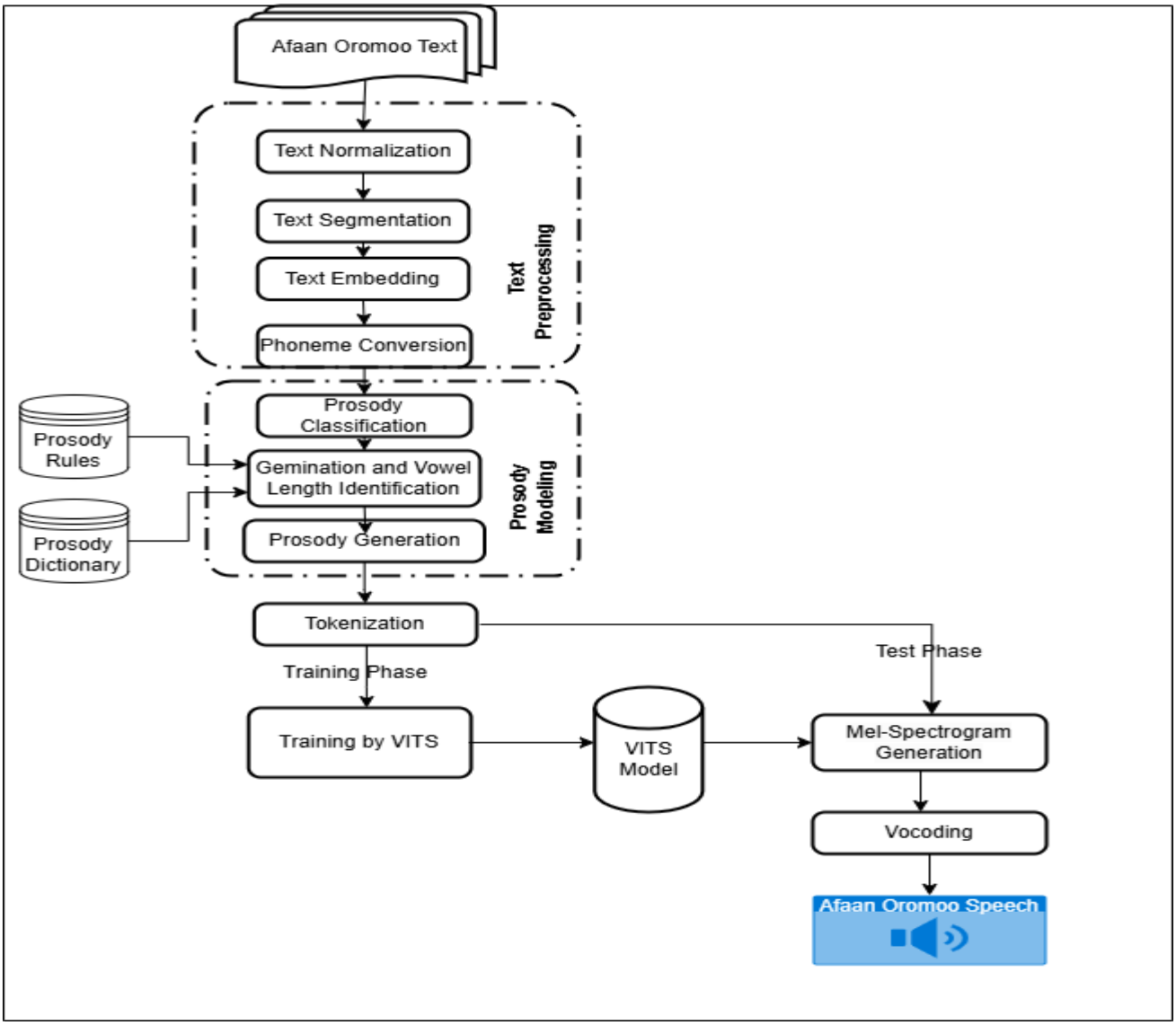

Figure 3: Overall architecture of proposed TTS transformer-based model for Afaan oromoo

4.3. Text Preprocessing

In the development of Text-to-Speech (TTS) systems, the quality of the synthesized speech heavily depends on the way input text is processed before it is fed into the speech synthesis. Text preprocessing serves us a critical step to convert raw text into a format that is both comprehensible and pronounceable by the TTS model. This report describes the preprocessing methods we implemented to enhance the performance of the TTS system. The preprocessing pipeline includes text normalization, text segmentation, text embedding, and phonetic conversion which collectively ensure that the output speech is natural and intelligible.

4.3.1. Text Normalization

In text normalization we ensures as all variations in the input text are handled in a consistent way by converting non-standard text into a standard format that is easier for the TTS system to process in later stages. Since raw text often contains numbers, abbreviations, dates, and other symbols that need to be converted into full, readable text. We address variations in spelling, number formats, and other textual inconsistencies to produce natural speech output. In text normalization we considered number expansion, abbreviation expansion, date and time expansion, currency and units expansion and special character conversion.

Number Expansion: Digits are converted into words. This step ensures that the numbers are read in their full form, which is important for natural speech synthesis.

Example: "123" → "dhibba tokkoof digdamii sadii" (one hundred twenty-three),"3.5%" →"dhibbeentaa sadii tuqaa shan" (three point five percent)

Abbreviation Expansion: Abbreviations like "Fkn." or "T/Lakk." are expanded into their full forms. This ensured that the TTS system would pronounce these words accurately rather than attempting to read them as they are spelled.

Example: "Fkn." → "Fakkeenya" (example), "T/Lakk." → "toora lakkofsaa" (Serial number), "BBO" → “Biiroo Barnoota Oromiyaa” (Oromia Education Bureau)

Date and Time Expansion: Structured data like dates or time is converted into a natural spoken format.

Example: "03/08/2025" → "Bitootessa saddeet, digdama digdamii-shan" (March eight, twenty twenty-five)

Currency and Units Expansion: Symbols like "Qr." or "kg" are expanded into their full spoken form.

Example: "Qr. 6" → "Qarshii ja'a" (six Birr), "10kg" → "kiiloogiraama kudhan" (ten kilograms)

Special Character Conversion: Symbols like "$", "%","#" were mapped to their verbal equivalents (e.g., "$" → "dolaaraa" (dollars), "%" → "dhibbeentaa" (percent) , "#" → "mallattoo riqichaa" (hashtag). This step prevented the system from misreading symbols as random characters, ensuring the speech output was contextually appropriate.

4.3.2. Text Segmentation

Once the text is normalized, we need to break the input text down into smaller, manageable chunks, like sentences or phrases, to make the text easier for the TTS system to process and convert into speech. We did segmentation to determine where to introduce pauses in speech, and to provide the necessary structure for later stages like phoneme conversion. In text segmentation we considered segmentation of sentences, phrases and words.

Sentence Segmentation: The input text is split into individual sentences based on punctuation marks like periods, commas, question marks, and exclamation points.

Example: "Fardis abbaa warraa gatee hin galu. Yoo kufee ala bule waliin ala bula malee." → ["Fardis abbaa warraa gatee hin galu.", "Yoo kufee ala bule waliin ala bula malee."]

Phrase Segmentation: In longer sentences, phrases are also split based on punctuation or conjunctions, creating smaller units that represent logical groupings in speech.

Example: "Waan gara sammuu dhaqu qajeelaafi sirrii yoo ta'e; jireenya namaaf illee qajeelchaa ta'a." → ["Waan gara sammuu dhaqu qajeelaafi sirrii yoo ta'e", "jireenya namaaf illee qajeelchaa ta'a."]

Word Segmentation: The text was split into individual words, ensuring that each word was treated as a separate unit. This step can help the TTS system map each word to its phoneme representation in the next steps.

Example: "Fardis abbaa warraa gatee hin galu" → ["Fardis", "abbaa", "warraa",”gatee”,” hin”,”galu”]

4.3.3. Text Embedding

We implement embeddings to represent text data as numerical vectors, which can be used by machine learning models. They capture the semantic meaning of words, phrases, or entire documents, allowing models to understand and process text more effectively. In text embedding we considered a contextual embedding.

Contextual Embeddings: In advanced systems (like transformer-based models), embeddings are context-dependent, meaning the same word in different sentences will have different vector representations depending on its meaning in that specific context.

Example: The word "Bara" can mean "year" or "time/period." If we pass this word through a contextual embedding model (like transformer-based models), it will give different vector representations for "Bara" in sentences like:

"Bara kana hojii guddaan hojjetamaa jira" (This year, a big project is being worked on.)

"Bara dheeraa booda wal arguu dandeenye" (After a long time, we were able to meet.). This allows the model to capture the appropriate meaning for each word based on sentence context.

4.3.4. Phoneme Conversion

One of the key components of the preprocessing pipeline was phonetic transcription, where we convert words into their corresponding phonetic representations. This step helped the TTS model accurately pronounce words that might not follow standard spelling-to-sound rules.

Phonetic Mapping: Words such as "jaalala" (love) were transcribed into their IPA (International Phonetic Alphabet) equivalent, /dʒaːlɑːlɑ/, ensuring that the TTS system would generate the correct pronunciation.

Context-Aware Phonetic Variants: For words with multiple pronunciations depending on context (e.g., "baruu" as /baˈruː/ in the present tense vs. /baˈre/ in the past tense), we implemented a context-aware phonetic transcription system. This system utilizes

surrounding words and the overall sentence context to determine the correct phonetic variant for each word.

Phonetic transcription ensured that the TTS system produced accurate and natural-sounding speech, especially for complex words and phrases.

Generally, this order helps us that the TTS system processes the text in a way that we capture both the syntactic structure (via segmentation) and the semantic meaning (via text embedding) before we finally convert the text into phonemes and producing natural-sounding speech.

4.4. Prosody Modeling

Prosodic features play a crucial role in conveying meaning, emotions, and emphasis in spoken language. The subject of prosody refers to the features of speech such as the level and patterns of syllables, words, and phrases that exhibit the principles of duration, pitch, and loudness. The degree of human-likeness of a TTS system is dependent on prosodic elements such as intonation modeling (phrasing and accentuation), amplitude modeling, and duration modeling (which includes phrasing and stressing), amplitude modeling, and duration modeling (this includes how long sounds last and pauses, affecting syllable length and speech speed). These features are usually observed over longer segments of speech. The sequence of syllable duration robust and fine-grained prosody is defined as a duration pattern. Variation in time length patterns determines the naturalness of speech. Intonation can be defined as the dynamics of fundamental frequency (F_0) contour over time, caused by vocal fold vibration. The intensity is considered to be closely related to the perceived loudness. The dynamic behavior of an intensity pattern is known as intensity or energy contour.

Even predicting prosodic labels from text is not an issue of TTS. To make our proposed solution comprehensive and practical, we have designed and implemented a prosody estimation module through prosody dictionary and prosody rules and integrated it into the TTS model. The prosody estimation module computes the prosodies of the input texts. It first classifies the input text to the corresponding reading types through the prosody label classifier and then generates the prosodies based on those reading types using the prosody generator.

4.4.1. Prosody Classification

The prosody classification component generates the tagged text after receiving the input text and the prosody dictionary. There are two components to the classification task: the dictionary-based and the rule-based. Both approaches can be combined as well, where a rule-based would defines a set of rules that govern how vowel lengthening and gemination should occur, and a dictionary-based approach involves referencing an extensive database where each word's correct form (including its vowels and geminates) is explicitly listed. This method relies heavily on stored word forms and typically does not involve active generation or rule application. Before returning the tagged text, the input text is first searched via the prosody dictionary. If the text is found there, it is given back. The dictionary search uses two main steps—exact match and partial match—to compare each word in the text with each entry in the dictionary. A tagged input text is generated when the exact match and part match have verified that the complete input text from the prosody dictionary has occurred and the individual units have been checked.

A dictionary-based prosody classification is a system that classifies prosody features in speech (such as pitch, intensity, duration, etc.) based on a predefined dictionary of prosodic patterns or rules. It relies on the mapping between specific words or phrases and their corresponding prosody patterns to make predictions. In the context of Afaan Oromoo, a prosody classification would be trained to recognize and classify prosodic patterns related to vowel length (duration) and gemination (consonant lengthening) in the language. It could be used to analyze and label the prosodic features in a given text or speech sample.

4.4.2. Gemination and Vowel Length Identification

As words are pronounced, vowels and consonants may be repeated to make the sound long. For instance, to say the Afaan Oromoo word aannan ("milk"), there is an emphasis on the sound of the letter n in the first case compared to its second pronunciation, which can be compared to the use of the English language in the word "pen-knife." Additionally, the two vowel sounds make the vowel long and can often change the meaning of the word, as in lafa ("ground") and laafaa ("soft"). *dh*, *ch*, *dh*, *ph*, *sh*, *ny*, and *zy* count as single consonants even if they are composed of two letters in a written form. Afaan oromoo has the typical Eastern

Cushitic set of five known short and five long vowels. Phonemic vowel length is used in writing.

Table 3: The five basic short and long vowels of Oromo (Lloret 1988)

	Front	**Central**	**Back**
High	i:ii		u:uu
Mid	e:ee		o:oo
Low		a:aa	

In Afaan Oromoo, the following are germination and vowel length algorithms that have association with this work:

Algorithm 1: *Algorithm to vowel length prosody rules for Afaan Oromoo*

Input: *Afaan Oromoo sentence*
Output: *Analyzed vowel length prosody rules for Afaan Oromoo*
Start
Define set of Afaan oromoo ***vowels*** ← *{'a', 'e', 'i', 'o', 'u'}*
Splitting the Sentence: *Split the input* ***sentence*** *into words based on whitespace and initialize it as* ***words*** *list and make in* ***lowercases*** ***//words = sentence.lower().split(' ')***
Initialization of a list to hold results: *Initialize a list named* ***results*** *// results =[] to store the analyses of each word*
for index, word in enumerate(words) do: *// Process each word in the sentence*
    Initialization of a dictionary: *// Initialize a dictionary named* ***word_analysis*** *// word_analysis ={} for the current word analysis*
    Append current word to word analysis: *//APPEND word_analysis[index] = word using the index as the key*
    Initialize a length type dictionary: *// as length_type={} which is a vowel type*
    Initialization of an Index variable: *assign zero to variable* ***i // i=0***

```
    While(i<length(word)) do: //Read all the words in the
    inputted sentence
        Assign current character to variable char: //assign
        word[i] to variable char
        if char in vowels then // Check if the char is a
        vowel
            Assign current character to variable
            current_vowel: //assign char to variable
            current_vowel, i.e., current_vowel=char
            IF current_vowel NOT in length_type://Initialize
            the entry in length_type if it doesn't exist
                length_type[current_vowel]=[] # Use a list to
                store multiple entries
            IF (i > 0 AND word[i - 1] = current_vowel) THEN:
            // Check if the current vowel is preceded or
            followed by the same vowel
                Long vowel is detected:
                length_type[current_vowel].append('long')
                //APPEND If the current vowel is the same as
                the preceded one, it's long
            Else IF((i = 0 OR word[i - 1] NOT in vowels)  AND
            (i= length(word)-1 OR word[i + 1] NOT in vowels))
            // If current vowel surrounded by consonants or
            it's at the edges of the word
                Short vowel is detected:
                length_type[current_vowel].append('short') //
                APPEND If current vowel surrounded by
                consonants or it's at the edges of the word,
                it's short
            End if
        End if
        Increament Index: i = i + 1 // just move to the
        next character, regardless of whether the character
        is a vowel or not by iterating i value
    End while
    Add the vowel length analysis to the word analysis:
    //add length_type to store the word and its vowel
    length classifications to initialized word_analysis
    Storing results: // APPEND the word_analysis for the
    current word to results
End for
Return: the list of word length_type analyses: // Output
results
End
```

Algorithm 2: Algorithm to vowel length prosody dictionary for Afaan Oromoo

```
Input: Afaan Oromoo sentence
Output: A dictionary containing the prosody analysis of
Afaan Oromoo vowels
Start
Define set of Afaan oromoo  vowels← {'a', 'e', 'i', 'o', 'u'}
Initialize empty Prosody Dictionary: // a dictionary named
as prosody_dictionary
Splitting the Sentence: Split the input sentence into words
based on whitespace and initialize it as list of words and
make in lowercases  //words = sentence.lower().split(' ')
for each word in words do: // Process each word in the
sentence
  Initialization of an Index variable: assign zero to variable
  i   // i=0
  While(i<length(word)) do: //Read all the words in the
  inputted sentence
    Assign current character to variable char: // assign
    word[i] value to variable char
    if char in vowels then // Check if the character(char) is
    a vowel
      Assign current character to variable current_vowel:
      assign char value to variable current_vowel , i.e.,
      current_vowel=char
      IF (i > 0 AND word[i - 1] = current_vowel) OR (i + 1 <
      LENGTH(word) AND word[i + 1] IN vowels) THEN // check if
      the vowel is long
        Long vowel word is detected: //APPEND word to
        prosody_dictionary[current_vowel]['long']['examples'],
        word // Long vowel condition
      Else //check if the vowel is short
        Short vowel word is detected: //APPEND word to
        prosody_dictionary[current_vowel]['short']['examples'],
        word // Short vowel condition
      End if
    End if
    Increament Index: i = i + 1 // move to the next character,
    regardless of whether the character is a vowel or not by
    iterating I value
  End while
```

End for
for each vowel in vowels do: *// Loop each vowel in each words*
Return the appended dictionary entries for each vowel:
prosody_dictionary[vowel] = { 'description': 'Vowel can be long or short.', 'long': { 'condition': 'Occurs when repeated or followed by another vowel.', 'examples': [] }, 'short': { 'condition': 'Occurs when not followed by another vowel.', 'examples': [] } }
End for
Return: *the completed vowel length type prosody dictionary:*
// Return the ***prosody_dictionary,*** *which now contains the vowel length analysis*
End

Algorithm 3: Algorithm to consonant gemination prosody rules for Afaan Oromoo

Input: *Afaan Oromoo sentence*
Output: *Analyzed consonants prosody rules for Afaan Oromoo*
Start
Define set of Afaan oromoo ***consonants*** *{'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'}*
Splitting the Sentence: *Split the input* ***sentence*** *into words based on whitespace and initialize it as list of* ***words*** *and make in* ***lowercases*** ***//words = sentence.lower().split(' ')***
Initialization of results list: *Initialize a list named as* ***results []*** *to store gemination results*
for each word in words do *// Process each word in the sentence*
    Initialize gemination status: *//SET is_geminated=FALSE for this word as default*
    gemination_results = {} ***#*** *Initialize the gemination_results dictionary*
    Initialization of Index variable: *//assign zero to variable* ***i*** ***// i=0*** *for character process*
    While(i<length(word)) do *// Process each character in the word*
        Assign of current character to a variable char:

```
    //assign word[i] to current character char variable
    IF char IN consonants THEN // Check if the character is
    a consonant
        IF char not in gemination_results:
            gemination_results[char] = []  # Use a list to
            store multiple entries
        IF (i > 0 and word [i - 1] == char: // THEN Check
        for gemination
            A geminated consonant is detected:
            gemination_results[char].append('geminated')
            //APPEND geminated
        ELSE IF (i == 0 or word[i - 1] not in consonants)
        and (i == len(word) - 1 or word[i + 1] not in
        consonants): // THEN  Check for non-gemination
            A non-geminated consonant is detected:
            gemination_results[char].append(non-geminated)
            //APPEND non-geminated
        ELSE IF (i == 0 or word[i - 1] in consonants): # If
        previous character is also a consonant but
        different, called in afaan oromoo 'irra
        butaa',’qubee-dachaa’
            gemination_results[char].append('non-geminated')
            # as non-geminated
        End if
    Increament Index: i = i + 1 // just move to the next
    character,
    End if
  End while
  results.append(f"{word,gemination_results} ")// APPEND
  results for the current word to results
End for
Return: results gemination or non-gemination for each word:
// Output results
End
```

Algorithm 4: Algorithm to consonant gemination prosody dictionary for Afaan Oromoo

```
Input: Afaan Oromoo sentence
Output: A dictionary containing the prosody analysis of  Afaan Oromoo consonants
Start
Define set of Afaan oromoo  Consonants← {'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z'}
create a prosody dictionary: //CREATE dictionary named as gemination_dictionary
Splitting the Sentence: Split the input sentence into words based on whitespace and initialize it as words list and make in lowercases  //words = sentence.lower().split(' ')
for each word in words do: // Process each word in the sentence
    Initialize flag for gemination: // SET is_geminated = FALSE as default value
    Initialization of a variable: assign zero to variable i // i=0 for character processing
    While(i<length(word)) do  // While there are characters left in the word
        Assign current character to variable char: // Assign word[i] to variable char
        IF char in Consonants then // Check if the character is a consonant
            IF (i + 1 < LENGTH(word) AND word[i + 1] = char) THEN // Check for gemination
                A geminated consonant is detected:  // is_geminated = TRUE
            Else
                A non_geminated consonant is detected:  // is_geminated = FALSE
            End if
        End if
        Increament Index: i = i + 1 // move to the next character
    End while
    IF is_geminated THEN // Add the word to the appropriate category based on gemination or not
        Add the word to geminated //APPEND word to gemination_dictionary['geminated']['examples']
```

```
    Else
        Add the word to non_geminated  //APPEND word to
        gemination_dictionary['non_geminated']['examples']
    End if
End for
Collect the gemination type dictionary of words: gemination
_dictionary = { geminated: { 'examples': [] }, non_geminated: {
'examples': [] }}
Return: the completed gemination type dictionary for each word:
// Return the gemination_dictionary,
End
```

4.4.3. Prosody Generation

A prosody generator is a system that is designed to generate or synthesize prosodic features for speech. In the context of Afaan Oromoo, a prosody generator would be trained to generate the correct prosodic patterns, including vowel length and consonant gemination, based on input text. This could be used in TTS systems to produce natural-sounding speech with the appropriate prosody for Afaan Oromoo.

A prosody generator for Afaan Oromoo based on vowel length and consonant gemination would aim to synthesize speech with natural prosodic patterns. Prosody generation use a rule-based or statistical approach to determine the duration and gemination for each syllable based on its prosodic features. Under the concept of vowel length, it is assumed that the longer a vowel is, the longer its duration will be in pronounciation. In the case of gemination, the duration of geminated consonants would be longer than that of single consonants. The output of prosody generation is generate the synthesized speech with the appropriate prosodic features, including vowel length and consonant gemination.

4.5. Tokenization

The tokenization process is a crucial step which helps us to convert the raw text (e.g., sentences) into a sequence of tokens that the TTS model can understand. This step is necessary because neural networks process numerical data, not raw text. For our TTS model, these tokens represent phonemes, graphemes, or other linguistic units that are necessary to generate corresponding speech. Once we normalized the text and converted to phonemes, tokenization breaks the text into smaller units (tokens) that our model can process. Typically, each phoneme is assigned a unique integer (or embedding) to represent it in the model. Our model uses character-level tokenization which breaks down text into characters (e.g., "N", "a", "m", "a") gives tokenized form [5, 17, 11, 17] (where each number corresponds to a unique character ID in the dictionary). In this step raw text (e.g., “Nama”) is the input whereas tokenized text (e.g., [N, a, m, a]) is the output of the tokenization phase.

4.6. Training by VITS

Once our dataset is prepared, we can start training our model which was built on the VITS framework. In this phase tokenized text (e.g., [N, a, m, a]) is the inputs. The core training process involves learning the mapping from tokenized text to speech (waveform) through the use of neural network architectures. VITS relies on multiple neural networks such as but not all text preprocessing, prosody model, and tokenization. Before training, we adjusted the training hyper-parameters (i.e. batch size, learning rates, etc.) in the configuration file for our dataset, based on our dataset size, audio features, and training preferences. When, we started training by running the training script and the model would be trained for several epochs, and we would verified if any loss values printed at regular intervals to ensure that the training progresses without errors. Training can take a time (depending on our GPU and dataset size). We monitored the training progress by visualizing training losses which stored in our logs and generated audio samples to evaluate the model's progress. Throughout training, the model saved checkpoints to disk in a file typically under the checkpoints directory. We can loaded these checkpoints later to resume training or for inference. After training, we able to use the trained model to inference (synthesize speech) from text. Our trained VITS model is produced a learned mel-spectrogram as output, which represents the audio in a time-frequency domain.

In our transformer-based TTS model, a trained model, VITS model that represents a unified approach adopts variational inference and an adversarial learning for end-to-end text-to-speech (VITS) model. VITS integrates a conditional variational autoencoder (VAE) with adversarial learning to produce high-fidelity speech directly from textual inputs. This model comprises of the encoder encodes input text into latent representations, which the decoder then converts into mel-spectrograms. Pre-trained checkpoints of VITS enable immediate inference for TTS conversion and can be fine-tuned on specific datasets for enhanced performance in particular voices or languages. The trained VITS model outputs mel-spectrograms, which are finally converted into audio waveforms by a vocoder.

4.7. Mel-Spectogram Generation

The input for this phase is tokenized text or trained model that has processed. The mel-spectrograms generated by creating 2D matrix where one axis represents time and the other axis represents frequency. We used mel-spectrogram to represent the frequency content of an audio signal. Our mel-spectrogram is then used to reconstruct the waveform (audio signal). This mel-spectrogram contains the necessary information to synthesize natural speech. The output of this stage is to produce a predicted mel-spectogram. The generated mel-spectrogram can then be passed to a vocoder (e.g., HiFi-GAN) to convert the predicted mel-spectrogram into a waveform for playback.

4.8. Vocoding

In this model architecture a vocoder (voice encoder/decoder) play a crucial role in the final stage of Afaan Oromoo speech generation. After the text input is transformed into an intermediate representation (like a mel-spectrogram), the vocoder converts this representation into a natural-sounding audio waveform. In this work, we used HiFi-GAN (High Fidelity Generative Adversarial Network), which can efficiently generate high-quality speech audio from mel-spectrograms. It uses a generative adversarial network (GAN) architecture, which includes a generator and discriminators. The generator samples mel-spectrograms to produce audio waveforms, while the discriminators ensure the generated audio is realistic.

CHAPTER FIVE: EXPERIMENT

5.1. Introduction

This Chapter is going to reveal the most fundamental parts of our research, starting with a comprehensive insight into the development environment and tools we have used. The following Section details the software and hardware settings used in our experiment, pointing out some implementation aspects that contributed to making a successful methodology, followed by methods of data collection including pairs of <text, audio>, experimental results, and discussion providing a concise overview of our experimentation.

5.2. The Development Environment and Tools

Regarding the implementation of our proposed model, we use a computer (with a Windows 11 operating system, a 2.40 GHz Intel CPU, 8 GB of RAM, and 1 TB of hard disk) and Python 3.6.6 to implement our algorithm/coding. And install different machine learning functions and utilities from Keras, Numpy, itertools, sci-kit-learn, gensim, and Pandas.

We used the following tools to develop the system:

- Microsoft Office 2013 for documentation
- PRAAT is freely available software for sound manipulation, phonetic, and acoustic analysis.
- Festival 2.5.0 is a currently freely available multilingual framework for building a speech synthesis system.
- Festvox 2.7.3 is currently open-source software for the voice-building process and includes program tools such as phone sets, lexicons, phrasing, etc.
- Speech Tools 2.5.0 is free software for manipulating the sorts of objects used in speech processing.
- SPTK 3.6: freely available software for speech signal processing and extracting speech parameters and re-synthesizing speech from the parameters
- Microphone for recording speech data.

- Adobe Audition CS6 is freely available software for displaying and labeling the speech waveform.
- Audacity is a tool used to split one audio dataset into multiple audio files.
- Lucidchart to create powerful, professional research and design online subscribing with an email account. No download is needed.
- Draw.io drawing different architecture design activities.

5.3. Dataset Preparation

Text Dataset Preparation

To create the Afaan Oromo speech database, one must first collect phonetically balanced sentences and store them. These sentences are gathered from various sources, including but not limited to news, blogs, stories, political essays, literary works, sports sections, magazines, holy books, proverbs, and newspapers. We have used 1600 sentences randomly selected out of the trained dataset for the evaluation purpose. We reuse best existing previous TTS approaches, including those for data preparation and waveform generation that we configured to be the best for Transformer. For our experiment, we gathered a dataset of 8,076 Afaan Oromoo sentences and corresponding audio clips.

Audio Dataset Preparation

We have a considerable amount of background noise, which we do not control in an audio recording. Nighttime recording will be saved from not losing in quality. We used the audio processing software Audacity, as well as Adobe Audition, to preprocess and denoise the recorded audio. Recording audio using a Samsung Galaxy A13 personal phone and an audio recording microphone. Last but not least, the corpus/collection can support this research on natural language processing tasks, especially TTS synthesis for a conversation system.

The partial recording process took place in an office environment with minimal noise. The speech is recorded at a noise-free studio using a PC at Oromia Broadcasting Networks (OBN) in Finfinnee, Ethiopia, with male journalists. The speech sample was recorded at 44.1 kHz stereo, and the files are stored in waveform format (.wav). The audio files are normalized and converted to conform to 16 kHz, 16-bit RIFF format as required by the

Festvox system and to make it simple to create raw files of small sizes. The software that is used to record the speech corpus is Praat. A regular microphone and a normal office computer made up of hardware equipment used to record the speech corpus. The text data is provided in 'metadata.csv', which consists of one record per line. The fields include (1) ID: the name of the corresponding audio file. (2) Text: the actual texts read by the reader. (3) Prosody: the reading types of the words tagged over the text. In general, it took a long time to prepare the dataset. The incoming text data is transformed into numerical values during training. The dataset has a total length of about 17 hours, and each audio.

5.4. Test Result

Many ways that result tests can be carried out. The method of evaluation employed in this study to measure the intelligibility and naturalness of TTS is assessed using the MOS. MOS is a widely-used metric, one type of subjective assessment, in which human listeners rate the quality of voice samples to determine the effectiveness of the speech synthesis. In MOS experiments, the subject is told to give their opinion on the condition based on a numeric scale, usually from one to five. Then the average of the responses is taken to indicate the success (or failure) of the experiment.

To assess the quality of this synthesis, we conducted MOS tests, which were evaluated subjectively by humans. We first prepare a guideline that explains the task to human annotators, including the ranking system employed (between 1 and 5, with increments of 0.5), where a scale, from 1 is the lowest natural/very unnatural and 5 is the highest natural/very natural. Then, we generated one thousands six hundred samples randomly from the test dataset sentences to compare the ground truth and the synthesized audio. There are 40 evaluators were asked to rate the naturalness and intelligibility of each sample during the tests. Audience members listened to randomly selected waveform samples and assessed their naturalness on a 5-point scale from 1 to 5. They were allowed to evaluate each audio sample once, and we normalized all the audio clips to avoid the effect of amplitude differences on the score. All of the quality assessments in this work were conducted in this manner. So, the performance of this model was measured, and the quality of synthesized speech is assessed in terms of intelligibility and naturalness, with results of 4.23 and 4.21, respectively.

5.5. Discussion

We presented the thesis for the Afaan Oromoo, which scaled speech technology to the language. We presented how we collected text and audio datasets, pre-trained models, and then built models for automatic speech synthesis. The overall results of the experiment demonstrate the effectiveness of using a transformer-based model for converting text into speech for the Afaan Oromoo language. The key findings include the use of spectrogram features and Mel-Frequency Cepstral Coefficients (MFCC) as the input acoustic features, which were able to effectively capture the sequential and temporal information in the speech data. The use of a transformer-based architecture rather than a traditional RNN proved advantageous, as the transformer's ability to model long-range dependencies and parallelization of computations led to improved performance compared to the RNN-based acoustic model.

Although the initial results were achieved on an overfitted model, further training and hyper-parameter optimization helped the model generalize better to the validation data, suggesting that with proper techniques, the Transformer-based model can achieve the desired results on the target Afaan Oromoo text-to-speech task. One of the key intermediate results was the successful generation of MFCC features from the speech source, where the model learned the parameters to predict MFCC sequences from the input speech, a crucial step in the text-to-speech pipeline.

We also mention the use of a transformer-based model for the acoustic modeling component of the system, which was able to effectively capture the sequential nature of the audio features and simulate the Afaan Oromoo language's 33 phonemes. The preprocessing steps include the removal of the apostrophe (‘) as punctuation and the mapping of the characters (phones) to numerical indices for the model input.

Overall, the transformer-based architecture demonstrates a promising approach for developing a high-quality Afaan Oromoo TTS system. With further refinement and optimization, the reported system has the potential to deliver accurate and natural-sounding speech synthesis for the Afaan Oromoo language.

CHAPTER SIX: CONCLUSION AND FUTURE WORKS

6.1. Conclusion

This study clearly addressed the development of TTS synthesis model for Afaan Oromoo using Transformer neural network. Afaan Oromoo, a Cushitic language spoken primarily in Ethiopia. TTS is all about a process of converting text as input into spoken words as output. Our work acknowledges previous efforts in developing TTS synthesizers for Afaan Oromoo but critiques them for lacking intelligibility and naturalness. We proposed and designed Afaan Oromoo TTS synthesis model using neural network approach to solve the previous efforts in developing TTS synthesizers for Afaan Oromoo for lacking intelligibility and naturalness. From neural network approach, we selected transformer-based neural network model because this model have demonstrated success in TTS for languages like English, achieved better naturalness and intelligibility and have appropriateness for Afaan Oromoo morphological analysis. The developed Afaan Oromoo TTS has many important applications, especially for people with vision impaired and an illiterate, which enable them to hear and understand with digital content through screen readers that are presented to different regions and languages. The proposed system architecture contains text preprocessing, prosody modeling components and other phases including text as input, tokenization, a training VITS model that learns to synthesize speech from text, mel-spectogram generation, vocoding and human-like speech as output. We also introduced consonant germination and vowel length in order to provide a prosodically rich dataset. As indicated by our experimental results subjective human evaluation (mean opinion score, or MOS) in terms of intelligibility and naturalness, a single speaker dataset, shows that our method outperforms the prior Afaan Oromoo TTS systems and achieves an expected MOS outcome comparable to ground truth. From this result, we believe that the prosody feature plays a vital role in getting the best result. We expect the proposed method to be applied in a wide range of voice synthesis tasks.

6.2. Contributions of this Work

The development of a TTS system for Afaan Oromoo using a transformer-based model makes significant contributions to the field of language technology, particularly for low-resource languages. Leveraging the transformer architecture, which excels at capturing long-range dependencies, allows the system to handle the complexities of Afaan Oromoo phonetics and syntax more effectively.

Among the major contributions of this study are:

- Proposed a new architecture for Afaan Oromoo TTS with the state-of-the-art approach.
- The implementation leads to synthesized speech that is notably more natural and intelligible,
- Improving the user experience for language learners and individuals with visual impairments.
- Cultural preservation by promoting the use of Afaan Oromoo TTS in digital environments and encouraging the development of applications that utilize the language.
- Highlights the potential for broader applications of the techniques developed, extending to other languages with similar characteristics.
- Provides valuable insights for the field of speech synthesis in under-resourced languages.

6.3. Future Works

Based on our findings in this thesis, we suggest the following to improve the quality of the system and to increase the quality of the synthesized speech.

- Speech emotion development for different types of emotions like normal, happy, anger and sadness, fear, and grief are some of the emotion types that make the speech output as well as waveform generation varied.
- Work with multi-speaker dataset.
- Excel at furthering naturalness in the subjective perception of synthesized speech quality.

- Improving existing models of speech synthesis with TTS approaches entails improving the quality, naturalness, and adaptability of synthetic speech.
- Improving the training dataset size by including a wide variety of speakers, texts, and linguistic variances.

References

[1] Yao Qian, Frank Soong, Yining Chen and Min Chu, "An HMM-Based Mandarin Chinese Text-to-Speech System," Beijing University of Posts and Telecommunications, Beijing, China, 2007.

[2] Rubeena A. Khan and J. S. Chitode, "Concatenative Speech Synthesis: A Review," *International Journal of Computer Applications,* vol. 136, no. 3, 2016.

[3] Yishuang Ning, Sheng He ,Zhiyong Wu, Chunxiao Xing, and Liang-Jie Zhan, "A Review of Deep Learning Based Speech Synthesis," *IEEE International Conference on Acoustics,,* vol. 9, no. 19, p. 4779–4783, 2019.

[4] Xu Tan, Jiawei Chen, Haohe Liu, Jian Cong, Chen Zhang, Yanqing Liu, Xi Wang, Yichong Leng, Yuanhao Yi, Lei He, Frank Soong, Tao Qin, Sheng Zhao and Tie-Yan Liu, "NaturalSpeech: End-to-End Text to Speech Synthesis with Human-Level Quality," in *Microsoft Research Asia & Microsoft Azure Speech*, 2022.

[5] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis and Yonghui Wu, "Natural TTS synthesis by conditioning wavenet on Mel-Spectrogram predictions," in *IEEE International Conference on Acoustics*, University of California,Berkeley, 2018.

[6] Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara., "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention.," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018.

[7] Naihan Li, Shujie Liu,Yanqing Liu,Sheng Zhao,Ming Liu and Ming Zhou, "Neural Speech Synthesis with Transformer Network," in *The AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, Long Beach, California, USA, 2017.

[9] Central Statistical Agency, "Population Projections for Ethiopia 2007-2037," UNFPA, Addis Ababa, July 2013.

[10] Eba Teresa Garoma, "Demonstratives in Afaan Oromoo," *Literature, Linguistics & Criticism,* vol. 11, no. 1, 2024.

[11] Tesfaye Tolessa, "Early History of Written Oromo Language up to 1900," in *Sci Tech Arts Res J.*, 2013.

[12] Workineh Tesema and Duresa Tamirat, "Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods," *American Journal of Computer Science and Engineering Survey,* vol. 1, no. 1, pp. 1-8, 2017.

[13] Tamrat Delessa Chala, Ashenaf Chalchisa Guta and Muluken Hussen Asebel, "Design and Development of a Text-to-Speech Synthesizer for Afan Oromo," *Spring Nature Computer Science,* vol. 3, p. 420, 2022.

[14] Dennis H. Klatt, "Review of text-to-speech conversion for English," *The Journal of the Acoustical So-ciety of America,* p. 737–793, 1987.

[15] Eyob Bayou Kasie and Yaregal Assabie, "Concatinative Speech Synthesis for Amharic Using Unit Selection Method," *Unpublished Masters Thesis, Department of Computer Science, Addis Ababa University, Ethiopia,* 2012.

[16] S. T. Ofgaa, "Concatenative Text-to-Speech System for Afaan Oromo Language," *Unpublished Masters Thesis, Department of Information Science, Addis Ababa University,Ethiopia,* 2011.

[17] Andrew J. Hunt and Alan W. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database," in *IEEE InternationalConference on Acoustics, Speech, and Signal Pro-cessing*, 1996.

[18] T. Nose, "Efficient Implementation of Global Variance Compensation for Parametric Speech Synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 24, no. 10, pp. 1694-1704, 2016.

[19] Yang J, Wang Y, Liu H, Li J and Lu J, "Deep learning theory and its application in speechrecognition," *Commun Countermeasure,* pp. 1-5, 2014.

[20] Paul Taylor, Alan W Black and Richard Caley, "The Architecture of the Festival Seech Snthesis System," in *Centre for Speech Technology Research*, University of Edinburgh, 80,South Bridge, Edinburgh EH1 1HN, UK, 2009.

[21] Alex Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," *Proc. Springer, Studies in Computational Intelligence,* vol. 399, pp. 1-20, 2012.

[22] Ilya Sutskever,Oriol Vinyals and Quoc V. Le, "Sequence to Sequence Learning with Neural Networks," in *In Proceedings ofthe Annual Conference on Neural Information Processing Systems*, Montreal, QB, Canada, December 2014.

[23] Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton, "Speech Recognition With Deep Recurrent Neural Networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 2013.

[24] Heiga Zen, Andrew Senior, and Mike Schuster, "Statistical parametric speech synthesis using deep neural networks," in *IEEE*, Vancouver, BC, Canada, 2013 ieee international conference on

acoustics, speech and signal Processing, 2013, pp. 7962-7966.

[25] Fady K. Fahmy,Mahmoud I. Khalil and Hazem M. Abbas, "A Transfer Learning End-to-End Arabic Text-To-Speech (TTS) Deep Architecture," in *Ain Shams University, Dept. Computer and Systems Engineering*, Cairo, Egypt, 2020.

[26] Tewodros Abebe, "Text-to-Speech Synthesizer for Wolaytta Language," *Unpublished Masters Thesis, Department of Computer Science, Addis Ababa University,* 2009.

[27] Mezgebe Araya and Minyechil Alehegn, "Text to Speech Synthesizer for Tigrigna Linguistic using Concatenative Based approach with LSTM model," *Unpublished Masters Thesis, School of Computing and Informatics, Mizan Tepi University, Ethiopia,* 2022.

[28] Muhidin Kedir Wosho, "Text to Speech Synthesizer for Afaan Oromoo using Statistical Parametric Speech Synthesis," *Unpublished Masters Thesis, Department of Computer Science, Addis Ababa University, Ethiopia,* June, 2020.

[29] Ishan Dongol and Bal Krishna Bal, "Transformer-Based Nepali Text-to-Speech," Information and Language Processing Research Lab, Kathmandu University, Dhulikhel, Nepal, 2023.

[30] Morka Mekonnen, "Text-to-Speech System for Afaan Oromoo," *Unpublished Masters Thesis, Department of Information Science, Addis Ababa University, Ethiopia,* 2001.

[31] Soressa Beyene and Raja.K, "Text-to-Speech Synthesis for Afaan Oromoo Language Using Deep Learning Approach," *Unpublished,Ambo University, Hachalu Hundessa Campus Institute of Technology Department of Computer Science,* vol. 101, 2022.

[32] Elena Voita, David Talbot, Fedor Moiseev and Rico Sennrich, "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned," in *arXiv preprint arXiv:1905.09418*, 2019.

[33] Awet Fesseha, Shengwu Xiong, Eshete Derb Emiru, Moussa Diallo and Abdelghani Dahou, "Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya," *Information,* vol. 12, no. 1, p. 52, 2021.

[34] Fantahun Gereme, William Zhu, Tewodros Ayall and Dagmawi Alemu, "Combating Fake News in “Low-Resource” Languages: Amharic Fake News Detection Accompanied by Resource Crafting," *Information,* vol. 12, no. 1, p. 20, 2021.

[35] Jemal Abate and Faizur Rashid, "A review of sentiment analysis for Afaan Oromo: Current trends and future perspectives," *Natural Language Processing,* vol. 6, no. 2949-7191, 2024.

[36] Omer Osman Ibrahim and Yoshiki Mikami, "Stemming Tigrinya Words for Information Retrieval," *COLING,* vol. 19, pp. 345-352, 2012.

[37] Atnafu Lambebo Tonja, OlgaKolesnikova, Alexander Gelbukh and Grigori Sidorov, "Low-Resource Neural Machine Translation Improvement Using Source-Side Monolingual Data,"

Applied Sciences, vol. 13, no. 2, p. 1201, 2023.

[38] Mohammad Hossein Jarrahi, Christoph Lutz and Gemma Newlands, "Artificial intelligence, human intelligence and hybrid intelligence based on mutual augmentation," *Big Data & Society,* 2022.

[39] Łukasz Kaiser and Ilya Sutskever, "Neural GPUs learn algorithms," in *In International Conference on Learning Representations (ICLR)*, 2016.

[40] "Wikipedia. Speech synthesis," 11 December 2023. [Online]. Available: https://en.wikipedia.org/wiki/Speech_synthesis. [Accessed 11 December 2023].

[41] Homer Dudley and Thomas H Tarnoczy, "The speaking machine of wolfgang von kempelen," *The Journal of the Acoustical Society of America,* vol. 2, no. 22, p. 151–166, 1950.

[42] Christine H Shadle and Robert I Damper, "Prospects for articulatory synthesis: A position paper," in *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 2001.

[43] Alan Black, Paul Taylor, Richard Caley, and Rob Clark, The festival speech synthesis system, 1998.

[44] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura, "Speech synthesis based on hidden markov models," in *Proceedings of the IEEE 101(5):1234–1252*, 2013.

[45] Xu Tan,Tao Qin, Frank Soong and Tie-Yan Liu, "A Survey on Neural Speech Synthesis," arXiv, 2021.

[46] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation," *Third International Conference on Spoken Language Processing,* 1994.

[47] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems,* vol. 99(7), p. 1877–1884, 2016.

[48] Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards, "Normalization of Non-Standard Words," *Computer speech & language,* vol. 15(3), p. 287–333, 2001.

[49] Maximilian Bisani and Hermann Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech communication,* vol. 50(5), p. 434–451, 2008.

[50] Wenfu Wang, Shuang Xu, and Bo Xu, "First step towards end-to-end parametric TTS synthesis: Generating Spectral Parameters With Neural Attention," *Interspeech,* p. 2243–2247, 2016.

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

Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A Generative Model for Raw Audio," in *arXiv preprint arXiv:1609.03499*, 2016.

[52] Andrew Gibiansky, Sercan Ömer Arik, Gregory Frederick Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou, "Deep Voice 2: Multi-Speaker Neural Text-to-Speech," in *NIPS*, 2017.

[53] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller, "DEEP VOICE 3: 2000-Seaker Neural Text-to-Speech," *ICLR,* p. 214–217, 2018.

[54] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao and Tie-Yan Liu, "FastSpeech: Fast, Robust and Controllable Text-to-Speech," in *arXiv:1905.09263v5*, 2019.

[55] Wei Ping, Kainan Peng, and Jitong Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *International Conference on Learning Representations*, 2018.

[56] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, "Fastspeech 2: Fast and high-quality end-to-end text-to-speech," in *International Conference on Learning Representations, https://openreview.net/forum?id=piLPYqxtWuA*, 2021.

[57] Jeff Donahue, Sander Dieleman, Mikołaj Binkowski, Erich Elsen, and Karen Simonyan, "End-to-End Adversarial Text-to-Speech," in *ICLR*, 2021.

[58] Kula Kekeba Tune, Vasudeva Varma and Prasad Pingali, Evaluation of Oromo-English Cross-Language Information Retrieval, Hyderabad,India: Language Technologies Research Centre IIIT, 2000.

[59] Fikadu Belda Kebede, "Dissimilation in Oromo Phonology," *INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT, College of Social Sciences and Humanities, Haramaya University, Ethiopia,* vol. 3, no. 13, pp. 187-196, December, 2014.

[60] Teferi Degeneh Bijiga, "The Development of Oromo Writing System," Unpublished Doctor of Philosophy (PhD) Thesis, University of Kent, November, 2015.

[61] Junhui Zhang, Junjie Pan, Xiang Yin, Chen Li, Shichao Liu, Yang Zhang, Yuxuan Wang, and Zejun Ma, "A Hybrid Text Normalization System Using Multi-Head Self-Attention for Mandarin," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[62] Nianwen Xue, "Chinese Word Segmentation as Character Tagging," *International Journal of Computational Linguistics & Chinese Language Processing,* vol. 8, no. 1 Special Issue on Word Formation and Chinese Language Processing, p. 29–48, February 2003.

[63] Kaisheng Yao and Geoffrey Zweig, "Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion," in *Sixteenth Annual Conference of the International Speech*

Communication Association, 2015.

[64] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, "Tacotron: Towards End-to-End Speech Synthesis," *Interspeech,* p. 4006–4010, 2017.

[65] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, "Deep Voice: Real-time Neural Text-to-Speech," *International Conference on Machine Learning,* pp. 195-204., 2017.

[66] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao, "Non-Autoregressive Neural Text-to-Speech," in *International Conference on Machine Learning, pages 7586–7598. PMLR,*, 2020.

[67] Yang Ai and Zhen-Hua Ling, "A Neural Vocoder with Hierarchical Generation of Amplitude and Phase Spectra for Statistical Parametric Speech Synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 28, p. 839–851, 2020.

[68] Shivangi Nagdewani and Ashika Jain, "A Review on Methods for Speech-to-Text and Text-to-Speech Conversion," *International Research Journal of Engineering and Technology (IRJET),* vol. 07, no. 05, pp. 4459-4464, May 2020.

[69] Alula Tafere Zegeye, "A Generalized Approach to Amharic Text-to-Speech (TTS) Synthesis System," in *Unpublished Masters Thesis, Department of Computer Science, Addis Ababa University*, Ethiopia, 2010.

[70] Mahlet Awel Temam, "Bidirectional Long-Short Term Memory Based Text-to-Speech Synthesis for Amharic Language," in *Unpublished Masters Thesis, Department of Computer Science, Addis Ababa University*, Ethiopia, 2020.

[71] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki5, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura and Wangyou Zhang, "A Comparative Study on Transformer vs RNN in Speech Applications," *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU),* p. 449–456, 2019.

[72] I. Solak, "The M-AILABS speech dataset," 2019. [Online]. Available: //www.caito.de/2019/01/the- m- ailabs- speech-dataset/.

[73] K. Ito, "The LJ speech dataset," 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset/. [Accessed Nov 2023].

[74] Nelleke Jansen, Aoju Chen, "Prosodic Encoding of Sarcasm at the Sentence Level in Dutch," in *10th International Conference on Speech Prosody*, Tokyo, 2020.

[75] Vineel Pratap ,Andros Tjandra , Bowen Shi , Paden Tomasello , Arun Babu , Sayani Kundu , Ali Elkahky , Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi , Alexei Baevski , Yossi Adi , Xiaohui Zhang , Wei-Ning Hsu , Alexis Conneau and Michael Auli, "Scaling Speech Technology to 1,000+

Languages," *arXiv,* 2023.

[76] Jaehyeon Kim, Jungil Kong and Juhee Son, "Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech," *arXiv,* 2021.

Signed Declaration Sheet

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Bayisa Bedasa Abdisa____

Signature: _____

Date: _____

Confirmed by Advisor:

Name: Yaregal Assabie (PhD)____

Signature: _____

Date: _____