
**SCHOOL OF GRADUATE STUDIES**

# END-TO-END SPEECH RECOGNITION FOR GURAGIGNA LANGUAGE USING DEEP LEARNING TECHNIQUES

**MSc. THESIS**

**ABDO NESRU EBRAHIM**

**MAY, 2025 G.C**

**WOLKITE, ETHIOPIA**

# Wolkite University

## School of Graduate studies

# END-TO-END SPEECH RECOGNITION FOR GURAGIGNA LANGUAGE USING DEEP LEARNING TECHNIQUES

**A Thesis Submitted to Wolkite University School of Graduate Studies, In Partial Fulfillment of The Requirements for The Degree of Master in Computer Science and Engineering in The Faculty of Computing and Informatics.**

**ABDO NESRU**

**Major-Advisor: - Sintayehu Hirpassa (Ph.D.)**

**Co-Advisor: - Azmeraw Desalegn (M.Sc.)**

**MAY, 2025 G.C**

**WOLKITE, ETHIOPIA**

# APROVAL SHEET

## WOLKITE UNIVERSITY

## SCHOOL OF GRADUATE STUDIES

We hereby, certify that we have read and evaluation this research titled " **End-to-End Speech Recognition for Guragigna Language Using Deep Learning Techniques** " prepared under our guidance by Abdo Nesru Ebrahim and we recommend that the thesis shall be submitted to as rewarding the requirements for the award of a MSc. Degree computer science and engineering.

| | | |
|---|---|---|
| Sintayehu Hirpassa (Ph.D.) | | 5/10/2025 |
| Major-Advisor | Signature | Date |
| Azmeraw Desalegn (MSc.) | | 5/10/2025 |
| Co-Advisor | Signature | Date |

As a member of aboard of examiner of master of sciences thesis open defenses examination we have read and evaluated this thesis prepared by Abdo Nesru Ebrahim and examined the candidate and we hereby certify that the thesis is accepted for the fulfillment the requirements for the award of the degree of master of science (MSc.) in computer science and engineering.

| | | |
|---|---|---|
| 1. Dr. MESFIN ABEBE | | 14/05/25 |
| Name of External Examiner | Signature | Date |
| 2._____ | _____ | _____ |
| Name of Internal Examiner | Signature | Date |
| 3._____ | _____ | _____ |
| Name of Chairman | Signature | Date |

Final approval and acceptances of the thesis is contingent up on the submission of its final copy to the council of postgraduate program (CPGS) through the candidate departments or school graduates committee (DGC or SGC).



II

# APPROVAL SHEET

## WOLKITE UNIVERSITY

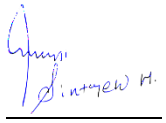
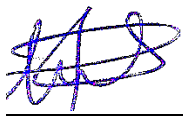
## SCHOOL OF GRADUATE STUDIES

End-To-End Speech Recognition for Guragigna Language Using Deep Learning Techniques

Submitted by:

| Abdo Nesru Ebrahim | _____ | 5/10/2025 |
|---|---|---|
| Name of student | Signature | Date |

Approved by:

| | | | |
|---|---|---|---|
| 1. | Sintayehu Hirpassa (PhD) |  | 5/10/2025 |
| | Name of Major-Advisor | Signature | Date |
| 2. | Azmeraw Dessalegn (MSc) |  | 5/10/2025 |
| | Name of Co-Advisor | Signature | Date |
| 3. | _____ | _____ | _____ |
| | Name of Chairman, DGC | Signature | Date |
| 4. | _____ | _____ | _____ |
| | Name of Deans, SGS | Signature | Date |

Stamp of SGS

# DEDICATION

This work is dedicated to my beloved children (***Abuzer & Nazeef***). Your unwavering love and boundless inspiration have been my greatest motivation. Thank you for being my guiding light.

# DECLARATION

I hereby affirm that this Thesis is my original work. I have adhered to all ethical guidelines in the preparation, data collection, analysis, and completion of this thesis. All scholarly materials included have been properly cited. I confirm that I have acknowledged and referenced all sources used in this document, and I have made every effort to avoid plagiarism.

This thesis is submitted as part of the requirements for a degree from the School of Graduate Studies at Wolkite University. It will be stored in the Wolkite University Library and will be accessible to borrowers according to library rules. I declare that this thesis has not been submitted to any other institution for any academic degree, diploma, or certificate.

Short quotations from this Thesis/Dissertation may be used without special permission, as long as proper acknowledgment is given. Requests for permission to use longer excerpts or to reproduce this thesis in whole or in part may be granted by the Head of the School or Department or the Dean of the School of Graduate Studies if deemed appropriate for scholarly purposes. In all other cases, permission must be obtained from me, the author.

Name: Abdo Nesru Ebrahim

Signature: _____

Date: _____

Department: computer Science and Engineering.

# BIOGRAPHY

The author, Mr. Abdo Nesru, was born on Sep 10, 1985 E.C, at ButaJira city East Gurage Zone, Central Ethiopia Region, Ethiopia, from his father, Ato Nesru Ebrahim, and mother Woizero Nuriya Darmulo. He attended his primary education at Mekicho primary school, his secondary education at ButaJira secondary and preparatory school. Then he joined Arbaminch University department of computer science and information technology in 2005 E.C and obtained his BSc degree in a computer science in June, 2008 E.C. He was employed in the wolkite university information communication technology (ICT) as a system administrator served for 7 years. Then, in November, 2014, he joined the department of Software engineering, in college of computing and informatics at Wolkite University to acquire his Master of Science degree in computer science.

# ACKNOWLEDGMENT

First and foremost, I want to express my gratitude to the Almighty Allah for being there for me throughout my life, protecting me from harm, blessing my working hours, and maintaining my health to a satisfactory level.

I also like to express my gratitude to Dr. Sintayew Hirpassa, my adviser, for his constant guidance and feedback during this research project. He gives me open-minded, critical advice that really motivates me to complete my work.

I also want to express my gratitude to those who give up their valuable time to read the written material I supply them during the data collection procedure.

Lastly, I would like to express my sincere gratitude to my entire family for supporting me during this study project and helping me get started.

Thank you

Name: Abdo Nesru Ebrahim

Signature: _____

Date: _____

# ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| AC | Acoustic modeling |
| ADAM | Adaptive Moment Estimation |
| AD | Analog to Digital |
| ASR | Automatic Speech Recognition |
| ANN | Artificial Neural Networks |
| BIGRU | Bidirectional Gated Recurrent Unit |
| BILSTM | Bidirectional Long Short-Term Memory |
| CV | Consonant vowel |
| CNN | Convolutional Neural Network |
| CTC | Connectionist Temporal Dependence |
| DL | Deep Learning |
| DNN | Deep Neural Networks |
| E2E | End-to-End |
| FM | Frequency Modulation |
| GD | Gradient Descent |
| GMM | Gaussian Mixture Model |
| GPU | Graphics Processing Unit |
| GRU | Gated Recurrent Unit |
| HMM | Hidden Markov Model |
| IDE | Integrated Development Environment |
| IEEE | Institute of electrical and Electronics Engineers |
| IPA | International Phonetic Alphabet |
| KHZ | Kilo Hertz |
| LM | Language model |
| LSTM | Long Short-Term Memory |
| LPC | Linear Predictive Coding |
| MFCC | Mel-Frequency Cepstral Coefficients |
| ML | Machine Learning |
| MPL | Multiple perceptron layer |

| | |
|---|---|
| NLP | Natural Language Processing |
| NN | Neural Network |
| OOV | out of vocabulary |
| RNN | Recurrent Neural Network |
| Seq2Seq | Sequence to Sequence |
| SGD | Stochastic Gradient Descent |
| SBG | Sebat Bet Gurage |
| SHL | Shared Hidden Layer |
| STFT | Short Times Fourier Transform |
| TDNN | Time Delay Deep Neural Network |
| WER | Word Error Rate |

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF TABLES IN THE APPENDIX

# LIST OF FIGURES IN THE APPENDIX

# ABSTRACT

*Speech recognition entails converting long sequences of acoustic features into shorter sequences of discrete symbols, such as words or phonemes. This process is complicated by varying sequence lengths and uncertainty in output symbol locations, making traditional classifiers impractical. Current automated systems struggle with speaker-independent continuous speech, particularly in low-resource languages like Guragigna, where the Cheha dialect poses additional challenges due to its purely spoken nature and lack of a rigid grammatical structure. To address these issues, this research develops an end-to-end speech recognition model utilizing deep learning techniques, specifically a hybrid CNN-BIGRU architecture combined with CTC and attention mechanisms. This approach aims to enhance alignment and robustness in noisy environments. To train and test the model, a text and speech corpus was created by compiling dataset from different sources like in Wolkite FM, the Old and New Testaments. Experimental results indicate that the CNN-BIGRU model achieves a Word Error Rate (WER) of 2.5%, showcasing improved generalization capabilities. Additionally, four recurrent neural network models LSTM, Bilstm, GRU, and BIGRU were evaluated, each configured with 1024 hidden units and optimized using the Adam optimizer over 50 epochs. The BIGRU model outperformed the others, achieving an accuracy of 97.50%, while the LSTM, Bilstm, and GRU models achieved maximum accuracies of 95.99%, 96.92%, and 96.25%, respectively. The successful implementation of this end-to-end speech recognition system significantly advances communication technologies for low-resource languages, enhancing accessibility for diverse linguistic communities. The findings underscore the effectiveness of deep learning methods in improving speech recognition performance in challenging linguistic contexts.*

Key words: Automatic Speech Recognition, NLP, Deep learning, LSTM, BILSTM, GRU, BIGRU, RNN, CNN.

# CHAPTER ONE

# 1. INTRODUCTION

## 1.1. Background of the Study

Language is a tool that humans used to express their feelings and emotions. Language is made up of words that make up a vocabulary and grammar that determines how these words should be used. It can take many different forms, such as oral communication sign language and written text. In particular speech involves using phonetic combinations of vowel and consonant sounds to express vocabulary words. On the other hand, phonetics deals with how people make and perceive sounds. People can communicate in their preferred language and express themselves through speech [1]. Humans communicate most naturally through speech. Numerous applications such as automated transcription streamlined human-machine communication and assistance for the physically and hearing challenged would benefit greatly from compact implementations of precise real-time speech recognizers. On speaker independent continuous-speech recognition tasks which people do with apparent ease current speech recognizers unfortunately perform seriously. Even while humans take speech recognition for granted and teenagers pick up the skill with little direct supervision it has proven to be challenging for robots to replicate [2].A series of uttered sounds, sometimes referred to as phonemes make up speech. Information is transferred from one speaker to another through speech. It is known as Automatic Speech Recognition (ASR) when the speech signal is transformed into a meaningful text or message [3].

Speech is the most fundamental effective and common way that people communicate. Worldwide a wide variety of spoken languages are in use. In order to exchange information, most human communication occurs verbally [4].Popularly referred to as ASR speech recognition is the process of using an algorithm carried out as a computer programmed to translate speech signal into a sequence of words Speech processing is one of the major fields of signal processing and in Speech recognition area aims at to develop techniques for speech input to a machine [5].

Speech recognition seeks to translate spoken language into text, whereas voice recognition seeks to identify a particular users voice. Voice recognition is often utilized for audio. Among ASR techniques natural language processing (NLP) is the most current and sophisticated. We are already witnessing some incredible outcomes in the form of smart smartphone interfaces like the

Siri app on the iPhone and a variety of systems utilized in high-tech and corporate settings, even if much more effort needs to be done before it reaches its full potential. This type of ASR is the closest thing to facilitating genuine human-machine conversation.

Speech recognition is the process of converting speech information into text that a computer system can understand using scientific techniques. This text may then be used to provide a range of services for either humans or machines. The following modules are typically found in speech recognition systems the decoder the acoustic extraction processing module the language associated model and the acoustic related model. The speech recognition systems basic idea is to gather the characteristic information from the speech information model, use training or other techniques to create an acoustic model that matches the speech model and then use scientific algorithms to decode this type of data to obtain the same information as the original [6].

Speech recognition systems receive audio input process it extracts and classify features and output the results as text. Formats for audio files include.au, .wav and raw audio data. The purpose of preprocessing is to advance the audio data. It entails feature extraction and segmentation. Speech signals are separated into frames of 10–30 ms per second during segmentation. Features such as pith, duration, SNR, and others are retrieved from segmented frames. A number of traits There are various extraction techniques including MFCC, LPC, PCA and others. An algorithm for classification is used to train the model [7].

Speech processing has drastically transformed as a result of deep learning capacity to automatically extract useful features from raw speech signals eliminating the need for human feature engineering. This discovery has led to significant advances in speech processing performance particularly in challenging scenarios involving noise and a range of accents and dialects [1]. The goal of end-to-end (E2E) speech recognition is to transcribe speech into text directly without the need for any preestablished alignments. In contrast to conventional hybrid approaches which typically entail intricate multi-stage hand-engineer pipelines E2E speech recognition streamlines training processes eliminates the need for time-consuming feature engineering and above all improves performance [8].With the benefit of directly anticipating target sequences from input speech E2E automated speech recognition ASR systems have made great strides [9].

Modern end-to-end approach on the other hand makes it possible to train and modify components using deep learning techniques within a single framework. Every component of this approach

might be viewed as being more successful than traditional components. The alignment issue is handled differently by end-to-end speech recognition techniques which integrate it into the optimization framework [10].

This research aims to create an end-to-end speech recognition model for the Guragigna language spoken in the Cheha dialect by utilizing deep learning techniques.

## 1.2. Motivation Of the Study

Communication through speech is a natural aspect of everyday life for the Gurage people. With the remarkable advances in speech recognition technology today it is essential for the Guragigna language to adapt to this speech recognition technology.

The lack of prior studies on end-to-end speech recognition for a very low-resource language, Cheha is one of the main motivations for conducting this study. Guragigna is a Semitic language The under-resourced nature of the Guragunga language in comparison to other languages is the driving force behind our research and development of an end-to-end speech recognition model.

Moreover, activating Hands-Free Technology when your hands and eyes are occupied while driving along with speech recognition is extremely beneficial and supportive for many visually impaired individuals who depend on screen readers and speech-to-text technologies.

## 1.3. Statement of the Problem

Cheha is one of the most significant West Gurage dialects. Along with Arabic, geez, Amharic, Tigriyna, Argobba, Harari, and Gaft, it belongs to the Semitic language family. The Gurage Zone in Central Ethiopia is home to the primary language, Cheha. Gurage is also spoken by settlers in a number of Ethiopian cities such as Addis Ababa, Dire Dawa and Hawssa. A 2007 census estimates that there are roughly 500,000 Chenga native speakers. This statistic excludes a sizable portion of Cheha speakers who live outside of the Gurage zone [11].

Chenga is one of the guragigna dialects which belongs to Semitic family that is spoken by Cheha people and the language is transcribed with the Geez script or Ethiopic writing and it has its own alphabetic character. Cheha dialects have unique phonetic system syllabic structure phonological, and morphological characteristics. For example, it has a collection of speech sounds that are unique to it and are not present in other languages [12]. For example, the speech sounds q', k', g', x', xw,

pw, bw, fw, and mw is not present from other language. There are not many studies being done with the Guragigna language these days. Nonetheless several tools have been developed such as a language keyboard and part-of-speech (pos) labeling. As far as researchers are aware there are still relatively few studies being done in the field of linguistic natural language processing [13]. For less resource languages speech recognition is essential for preventing extinction. One of the less resource languages Cheha faces the issue of data source scarcity and some of its phonological morphological and orthographic features to develop speech recognition research is mandatory.

Speech recognition involves converting long sequences of acoustic features into shorter sequences of discrete symbols, like words or phonemes. This task is complicated by the differing lengths of the sequences and the uncertainty of output symbol locations. Consequently, classifiers that predict targets for each input frame are not practical. Instead, the studied model is a Recurrent Neural Network (RNN) that doesn't require explicit alignment. It uses an attention mechanism during decoding to evaluate input frames against its hidden state, selects relevant frames, and generates a context vector to update the hidden state and produce the next output symbol [14].

Traditional ASR systems are made up of a lot of different parts and information sources, particularly speech signal preprocessing, robustness techniques for recording conditions, phoneme inventories and pronunciation lexica, phonetic clustering, handling of words that are not in the vocabulary, different adaptation/normalization techniques, complex training schedules with multiple goals, such as sequence discriminative training, etc. However, deep learning's promise sparked effective methods for combining previously distinct modeling stages, such as incorporating feature extraction and speech signal pre-processing into acoustic modeling [15].

Voice search and other applications use automatic speech recognition (ASR), a well-established technology. However, current systems rely on difficult structures like Gaussian mixture models (GMMs) and hidden Markov models (HMMs). It is challenging for non-experts to create ASR systems for new languages or applications because of this complexity, which includes manually created pronunciation dictionaries and text preparation. From beginning to end ASR uses a deep learning framework with a single-network architecture in an attempt to make this simpler. These techniques eliminate the need for linguistic understanding and enable easier training by using simply matched auditory and language data. This method makes it easier to create ASR systems without specialized skills [16].

A feature extraction module an acoustic model, a language model and a decoder are some of the various parts that make up traditional speech recognition systems. Large volumes of data and manual feature engineering are needed for this kind of speech recognition which can be labor-intensive and time-consuming. In contrast the end-to-end model replaces multiple modules with a deep neural network realizing the direct mapping of acoustic signals into label sequences without carefully-designed intermediate states. Besides, there is no need to perform posterior processing on the output [17]. According to many studies there are three main types in speech recognitions for end-to-end with encoder-decoder Attention, RNN Transducer and CTC based model development.

a hybrid CTC/attention model to address misalignment problems in extended sequences and noisy environments for end-to-end speech recognition. Although it assumes conditional independence, CTC efficiently computes loss and permits label repetitions. Although the attention-based encoder-decoder requires human adjustment and suffers from misalignment, it directly maps auditory frames to characters. CTC/attention successfully utilizes CTC's alignment capabilities to enhance performance and speed up learning by utilizing a shared encoder trained with both CTC and attention [18]. We have motivated to do this research because of end-to-end speech recognition is a hot and an interesting research area We have formulated the following research questions for these the above problems.

## 1.4. Research Questions

The following research questions used to find solution for defined problem:

RQ1. Which pre-processing steps best for cheha speech and text dataset preparation for model training and testing?

RQ2. Which feature extraction procedures best to extract appropriate combination of features from cheha dialects for model building?

RQ3. Which deep learning Models best performer to build cheha speech recognition model?

## 1.5. Objective of the Study

### 1.5.1. General Objective

- The general objective is to develop end-to-end speech recognition model using deep learning approaches for Guragigna language in case of Cheha dialect.

### 1.5.2. Specific Objective

To achieve the general objective, the specific objectives will be targeted.

- To collect speech and text data set from different source of cheha dialect.
- To design speech to text model for cheha dialect using deep learning algorithms.
- To extract relevant feature from speech data using feature-extraction techniques and text transcription encoding with sequence padding.
- To select the best performer deep learning algorithms from RNN models.
- To train the selected model with experimentation and result analysis for the developed speech to text model.
- To test the performance of models on unseen data.
- To write the outcomes result, recommendation, contribution, and future directions.

## 1.6. Scope of the study

The goal of this research to develop model for Guragigna speech recognition (transcribe speech to text) specifically for the Cheha dialect. This will be achieved by utilizing deep learning techniques and implementing with an RNN pipeline. This research does not address speech synthesis (text to speech) and it does not take into account visual information including body language gestures, facial expression and speech emotion recognition. The source of dataset includes from the bible, books and wolkite FM radio the merging the file and recording text by 27 native speakers. The proportions of age and gender have all been included in the data analysis.

## 1.7. Limitation of the study

The limitation of the study is absence of prior speech and text dataset for the language this is primary constraint for this research. Consequently, it is difficult to choose terms that were phonetically balanced as well as to find a volunteer to record their utterances. Since it may be some of the speakers is under stress and make inconsistent utterances and because the recording environment chosen will not primarily intended for recording speech it harm the recording data.

Human and environmental factors also have an impact on the outcomes of the experiment. Despite this limitation our contribution in the area of Natural language processing especially speech recognition by using deep learning approaches by using available and prepared corpus accurate and reliable result in speech recognition for the area of Natural language processing.

## 1.8. Significance of study

Beyond the academic difficulties it presents to qualify as a thesis studying research on Automatic speech recognition systems for Guragigna is useful. This research opens the door to discover some of the potential approaches to creating speech recognizer for guragigna. Future studies could build on this work to better the current man-machine interactions, particularly as they relate to language speakers. It can be used as an alternate learning and teaching tool for universities research academies and other interrelated organizations. The results of this study can also be used to create a fully functional Guragigna speech recognizer. It can also be used for standard automatic speech recognition tasks including dictation command and control telecommunication and use by individuals with disabilities.

In general, automatic speech recognition has the following benefits in addition to being a natural means of interacting with devices like computers phones televisions etc. Accelerates computer data entry (even for the fastest typists utilizing the speech is far faster than using the ubiquitous keyboard). Prevents pains associated with typing (such as boring stress injury) and allows for unrestricted eye and hand mobility when using the same as cannot be said of systems that rely on keyboards, mice or touch screens for input. Additionally, the proposed end-to-end Speech Recognition for the Guragigna language help for Aiding the visually impaired person.

## 1.9. Thesis Organization

This thesis organized into six chapters. In chapter one Background information problem statement, study objectives, study scope, researcher technique and study importance are all included. The second chapter reviews the literature including the history methodology and methods of the ASR system as well as analogous studies that have been conducted both domestically and internationally before this research. To determine what insights and suggestions they might have for this study and the Cheha language earlier research has been compiled in connection to one another. The languages overview phonetics and orthographic writing rules have all been covered. Additionally, the rules of linguistic syllabification and word creation were discussed. The third chapter describe

material and methodologies used in the research including data preparation methodology and that are use software and hardware tools. In Chapter 4 The design employed in the study including the feature extraction, analysis, and design procedures is described. In Chapter 5 The experiment and its outcome are explained Various experiments were carried out in this study and the interpretation of the results has been explored. In the final chapter the study report findings and suggestions for future research are covered.

# CHAPTER TWO

# 2. LITERATURE REVIEW

## 2.1. Introduction

The goal of the developing field of deep learning-based speech recognition is to create self-sufficient systems that can effectively and reliably transcribe speech into text with little help from humans. This investigation of the literature offers a thorough summary of the state of speech recognition today covering its history important models and techniques and most recent developments. Furthermore, we examine the constraints and difficulties encountered by these systems and suggest possible directions for further investigation. This review provides thorough examination of the body of research on end-to-end speech recognition making it an invaluable tool for scholars and experts interested in this dynamic and quickly developing topic.

## 2.2. Overview Of Traditional Automatic Speech Recognition

At Bell Telephone Laboratories, Davis, Biddulph, and Balashek developed the first automatic speech recognition (ASR) system in 1952, which recognized isolated digits from 0 to 9 for a single speaker. Olson and Belar developed a phonetic typewriter in 1956 that recognized ten distinct syllables, but this system was also speaker-dependent and required a lot of training [19]. ASR systems are made especially to deal with certain problems and system design is influenced by a number of factors and the operational scope of each produced speech recognition system must be established by taking into account a number of challenges. The modeling units used for recognition (words, syllables or phonemes) are important considerations. amount of vocabulary (from little to large) task syntactic complexity (using N-gram language models from simple to complex) task complexity speaker style (trained, adaptive, independent, or dependent) speaking environment (such as quiet rooms or noisy settings) and speaking mode (isolated, connected, continuous and spontaneous). Furthermore, the transmission route and the quality of transducers such as array microphones, cell phones, telephones and high-quality microphones also play important roles [20]. The essential components of a standard statistical speech recognition system are demonstrated including the pronunciation lexicon and the required knowledge sources which include textual and speech training resources. Additionally highlighted are the essential training and decoding procedures. Following the feature analysis carried out on the audio data through feature extraction

the acoustic and linguistic models produced during the training process function as knowledge sources throughout the decoding phase [21].

Automatic speech recognition is seen as a series of transformations that convert the fine details of an audio speech signal into its underlying phonetic structure. Generally, this means that a speech recognition system performs a transformation from spoken language to written text, where the output of the system represents the text corresponding to the recognized speech [22]. ASR is a technology that enables machines to transform speech into written text. ASR is utilized in various applications such as dictation, voice search, language translation and more. The ASR process comprises several steps. Initially, the input speech undergoes pre-processing to eliminate background noise and distortions that may impact transcription accuracy and next speech is analyzed to detect individual sounds, words and character a process known as acoustic modeling. This involves using statistical models to align the sounds with recognized phonemes and words. After that, the identified speech is converted into text through language modeling, where algorithms predict the most probable sequence of words the speaker intended based on the identified sounds. Finally, the text output is post-processed to correct any errors and enhance its readability.

## 2.3. Overview Of End-To-End Automatic Speech Recognition

A recent advancement in automatic speech recognition is E2E (End-to-end) automatic speech recognition is based on a neural network and has many benefits. With models that operate at a low speech frame rate E2E automatic speech recognition is a single integrated solution with a significantly simpler training methodology. This shortens the time needed for learning and decoding and enables cooperative optimization with later processing like comprehending natural language [23]. An important development in the field of automatic speech recognition (ASR) is represented by end-to-end speech recognition models. Conventional ASR systems usually entail a multi-stage complex pipeline that includes decoding language modeling and acoustic modeling. on the other hand, End-to-end models simplify the architecture and eliminate the need for intensive feature engineering by directly mapping acoustic signals to text sequences [24]. An end-to-end model converts a series of input acoustic properties straight into a series of graphemes or words. A system that has been trained to maximize parameters associated with the end evaluation metric of interest usually in word error rate [25]. The majority of conventional ASR model use separately

trained language, pronunciation and acoustic model components. It takes time and professional expertise to define phoneme sets and create pronunciation lexicon for a given language.

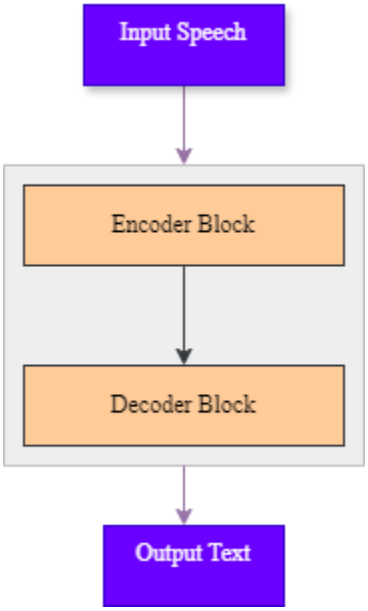



Figure 2.1 End-to-end Speech Recognition

Traditional speech recognition often employs a GMM-HMM acoustic model, while the DNN-HMM acoustic model distinguishes itself by using deep neural networks rather than the Gaussian mixture model. This modification enables more effective modeling of the input signal for probability estimation. Moreover, the input spectral features of the DNN model differ significantly from those used in traditional methods. Despite the widespread usage of Mel-frequency cepstral coefficients (MFCC) DNN uses windowing and framing to process sound waves. The DNN model also considers frame splicing whereas the GMM model simply accepts individual frame properties as input [26]. Transition to complete end-to-end ASR models have become more popular because to limitations in standard ASR model especially for low-resource languages. By combining the entire pipeline into a single neural network these models modernize the ASR process and have the potential to enhance performance in demanding language contexts [27].The end-to-end framework for speech recognition involves directly converting the input sequence of acoustic feature vectors into an output sequence of tokens, which can include phonemes, characters or words [28]. End-to-End speech recognition approaches address the alignment problem differently incorporating it into the optimization framework [10].

## 2.4. Types Of Automatic Speech Recognition

Speech recognition is categorized into various classes based on factors such as speaker model, vocabulary, channel, and utterance type, reflecting their ability to recognize speech.

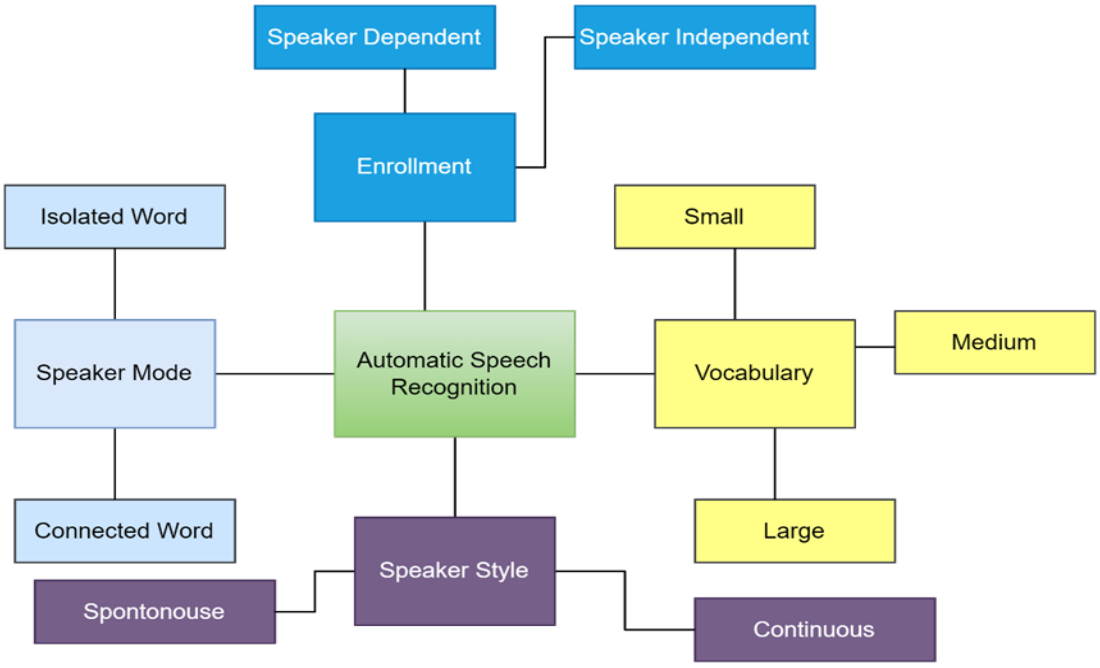



Figure 2.2 Types of Speech Recognitions

### 2.4.1. Types Of ASR Based on Speech Utterance

A spoken language or vocalization piece that gives the machine a single meaning is called an utterance. "Utterances can consist of a single word, a few words, a complete sentence, or even several sentences" [29].

#### 2.4.1.1. Isolated Words

Isolated word recognition is the process of recognizing a unique speech signal as a single word when the output is unambiguous and the signal is clearly segmented. For all possible hypotheses, this recognition mostly depends on the auditory models with little to no reliance on language models. Isolated word recognizers can only process one utterance at a time since they usually demand silence on both sides of the sample window for each utterance. These systems frequently use "Listen/Not Listen" statuses, which require the speaker to pause in between sentences [4].

#### 2.4.1.2. Connected Words

While connected word systems, or more precisely connected utterances, are comparable to isolated word recognition, they allow distinct utterances to be uttered consecutively with only a brief delay in between.

#### 2.4.1.3. Continuous Speech

The next development in speech recognition is continuous recognition. Since continuous speech recognizers need specific methods to detect utterance boundaries, creating recognizers that can handle continuous speech is especially difficult. Users can speak almost naturally as the computer analyzes their words thanks to continuous voice recognizers. This feature is essentially similar to computer dictation.

#### 2.4.1.4. Spontaneous Speech

The final kind of utterance-based speech recognition system is the spontaneous speech recognition system, which can identify natural speech. Speech typically comes out of the mouth abruptly with this kind of speech recognition system, and mispronunciation may occur. A discussion between two or more people might serve as a good illustration of this kind of speaking [30].

### 2.4.2. Types of ASR based on size of Vocabulary

A speech recognition system's accuracy, complexity, and processing demands can all be impacted by the amount of its vocabulary. Different vocabulary sizes small, medium, large, or extremely large may be required for different applications. In particular, a short vocabulary is between one and one hundred words, a medium vocabulary is between one hundred and one thousand words, a large vocabulary is between one thousand and ten thousand words, and a very large vocabulary is more than ten thousand words.

### 2.4.3. Types of ASR based on Speaker Model

All speakers possess distinctive voices shaped by their unique personalities and physical characteristics. Consequently, speech recognition systems can be broadly classified into three categories: speaker-dependent, speaker-independent, and adaptation-based systems, which adjust according to the specific model of the speaker.

#### 2.4.3.1. Speaker Dependent models

By using speaker-dependent data to customize their recognition abilities for each speaker, speaker adaptive speech recognition systems effectively reduce error rates [4].

#### 2.4.3.2. Speaker Independent models

One kind of ASR intended to identify speech from a wide range of speakers is speaker-independent systems. Although different voices can be accommodated by these systems, their development is frequently more difficult and expensive. Because each speaker has distinct speech features and because individual representations of speech parameters can differ greatly, their accuracy is typically lower than that of speaker-dependent systems.

## 2.5. Approaches To Automatic Speech Recognition

ASR systems can be divided into groups according to various methods. These methods can be broadly divided into three categories.

### 2.5.1. Acoustic phonetic Approach

The foundation of the acoustic phonetic method is the sound waves generated by the vocal organs of humans during speech. This approach shows that there are only a limited number of unique phonetic units in spoken language. These phonetic units are classified according to a set of speech signal characteristics, including strength, frequency, and duration, which aid in recognizing and distinguishing different speech sounds [31]. The acoustic phonetics hypothesis, which holds that spoken language is made up of distinct, limited phonetic units, is the foundation of the acoustic phonetic approach. Specific acoustic properties of each of these phonetic components emerge gradually in the speech signal or its spectrum. This technique's initial step involves both feature identification and speech spectrum analysis. Through this procedure, the spectral data is transformed into a collection of features that specify the general acoustic properties of different phonetic units. The speech signal is split up into stable acoustic regions in the next stage, which is called segmentation and labeling. The speech's phoneme lattice is then effectively described by assigning one or more phonetic labels to each divided region. More precise spoken language recognition based on its auditory characteristics is made possible by this methodical technique [32].

### 2.5.2. Pattern matching Approach

Pattern recognition, also known as pattern matching is the second strategy used in ASR systems. Pattern recognition is the process of measuring the similarity between an unknown test pattern categorized from a trained model and its reference class pattern. This approach's key features include a well-designed mathematical framework, robustness and invariance to various user speech vocabularies, and a collection of algorithms for feature extraction and pattern comparison. This feature enables it to use training methods to create a consistent representation of speech patterns. Since the previous couple decades, this technique has taken the lead in voice recognition. The pattern recognition approach contains two sub approaches: the stochastic approach and the template approach.

#### 2.5.2.1. Template Based Approach

The word "template" is frequently used to refer to two essentially distinct ideas: either the representation of a single speech segment with a known transcription or an average of several distinct speech segments. The DTW algorithm allows for the comparison of both kinds of templates with a speech input segment. It has a tag with meta-information, a transcription of the sounds or words it represents, a series of successive acoustic feature vectors, and knowledge about nearby templates [33]. template-based methods, where the best match is found by comparing an unknown speech to a list of prerecorded words (the template). This has the benefit of employing word models that are fully exact, but it also has the drawback that because the prerecorded templates are fixed, it is only possible to simulate speech variances by utilizing numerous templates per word, which eventually becomes impracticable. It soon becomes clear that the traditional DTW algorithm, which uses the Euclidean distance as a local distance metric in conjunction with a straightforward beam search, will not work well for the practical implementation of template-based recognition, either computationally or in terms of performance. Dynamic temporal warping (DTW) for speech pattern matching is the foundation for the development of isolated word speech recognition systems. To ensure that the input speech and reference templates have the same phoneme positions, the DTW procedure nonlinearly expands or contracts the time axis.

#### 2.5.2.2. Stochastic Approach

The process of choosing a series of non-deterministic choices from sets of options is referred to as stochastic. In order to handle ambiguous or partial information derived from speech signals from various sources, including confusable sounds, speaker variability, contextual effects, and homophones, the stochastic approach uses probabilistic models. The state of the art in speech recognition at the moment is the stochastic technique. The following stochastic techniques are employed in the construction of the ASR system in the state-of-the-art speech recognition technology.

##### 2.5.2.2.1. Discriminative learning-HMM-ANN

Neural network-based discriminative training offers a natural and effective way to evaluate a speech segment's probabilities. This kind of method is rarely utilized for continuous speech recognition, but it works best for small speech units like single words and phonemes [5].

##### 2.5.2.2.2. Generative learning-HMM-GMM

Conventional speech recognition system uses HMM-GMM approach to represent sequential structure of speech data. In the HMM-GMM, the spectral representation of sound wave is model by GMM and the short-term stationary speech or phone sub segment signal associated to HMM state. then using a sequence of HMM state and GMM observation probability, the speech recognition problem is model. state of art system uses this approach to achieve a good performance. HMM-GMM model can be trained automatically, easy and computational feasible to use. But GMM model statistically inefficient to model data on or nearly manifold in the data space [6].

##### 2.5.2.2.3. Deep learning HMM-DNN

Deep learning, also known as representative or unsupervised learning, is a popular speech recognition technology that can effectively take the place of GMM in ASR systems. The first type of deep learning, such as the deep auto-encoder or the deep Boltzmann machine, is designed to describe the correlation property of data. The third is a combination of these two types, whereas the second is meant to offer discriminative power of pattern classification utilized for speech coding, such as tandem ML.

### 2.5.3. Artificial intelligence Approach

It combines the phonetic method of acoustics with the pattern recognition approach. Examples of artificial intelligence approaches are Deep Neural Networks (DNN) and Recurrent Neural Networks (RNN) [7]. It combines the pattern recognition technique with the phonetic method of acoustics. Recurrent neural networks (RNN) and deep neural networks (DNN) are two examples of artificial intelligence techniques [32].

## 2.6. Deep Learning with Speech Recognition

In artificial intelligence, speech recognition has always been a difficult subject. Variability in speech patterns, background noise, and accent variances are only a few of the numerous obstacles that must be overcome. However, these difficulties have been somewhat lessened with the development of deep learning. We are now able to create speech recognition algorithms that are more accurate and dependable because in deep learning. One area of study within machine learning is deep learning. It is comparable to the creation of artificial neural networks. In essence, it is a technique for training deep structural models and an algorithm for representing intricate relationships between data over several layers. It is currently employed in social filtering, machine translation, and speech and picture recognition. The majority of the learning algorithms used today, including regression and classification, fall under the deep structure of the model, while the GMM, HMM, support vector machine, and multi-layer perceptron models fall under the shallow structure. The most popular classification model is this one, and the shallow structure's shared characteristic is the input signal and features can be converted to the feature space of a specific problem using this straightforward structure. A common characteristic of shallow structures is their ability to convert input signals and features into the feature space of a particular problem using a simple structure. However, complex functions are challenging to express and their processing of natural signals (such as natural images or human speech) has certain limitations [34].

Speech recognition involves the process of converting spoken words into text. There are two main approaches to speech recognition: acoustic modeling and language modeling. Acoustic modeling involves the use of statistical models to represent the relationship between speech signals and the corresponding phonemes or words. Language modeling involves the use of statistical models to represent the relationship between words in a sentence.

### 2.6.1. Language Modeling with Deep Learning

To improve the comprehension of words from the anticipated characters, language models are incorporated into the end-to-end character-based ASR model. The character-based ASR model may create a correlation between the specified character classes and spaces after learning various sounds and characters. By transforming the anticipated characters into pre-existing words, a language model enhances the ASR model. A language model enhances the current ASR model with word understanding and natural language processing (NLP). Deterministic and statistical models are the two categories of language models. In NLP, deterministic language models which are based on grammatical rules are less prevalent. Statistical language models, which are employed in NLP, use the likelihood that a word sequence will occur based on earlier words [35].

### 2.6.2. Acoustic Modeling with Deep Learning

Deep learning-based techniques like connectionist temporal classification (CTC) and the deep neural network-hidden Markov model (DNN-HMM) have significantly advanced the field of acoustic modeling (AM). HMM models have been extensively studied in ASR research offering a reliable framework for mapping audio signals to phonetic units. On the other hand, CTC provides an end-to-end training approach that just needs the input and output sequences doing away with the necessity for pre-alignment. Further increasing the adaptability and effectiveness of speech recognition systems is the sequence-to-sequence (S2S) model which has demonstrated efficacy in handling ASR tasks without the need for a language model (LM) or pronunciation dictionary [36].The act of converting an audio source into a string of phonemes or words is known as acoustic modeling. Estimating the probability of a phoneme or word from the audio input is the primary goal of acoustic modeling. Deep neural networks or Hidden Markov Models (HMMs) are typically used to do this. Because deep neural networks can learn to extract features from audio inputs and link them with phonemes or words they have shown efficacy in acoustic modeling for speech recognition. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are the two most common forms of deep neural networks used in speech recognition [17]. Speech recognition systems that use standard pipelines still only use deep learning algorithms to a limited extent. For collaborative training the end-to-end model combines several components of conventional speech recognition including the language model pronunciation lexicon and acoustic model, into a network. Without meticulously planning the intermediate state the end-to-end model achieves a direct mapping of the input sound sequence to the label sequence drastically simplifying

the training procedure and lowering the computational cost. This integration is also used in low-resource speech recognition since it lessens the reliance on prior expert knowledge and circumvents the problem that.

## 2.7. Model used in End-To-End Speech Recognition

### 2.7.1. Convolutional Neural Networks (CNN)

CNN is a kind of neural network that works especially well for processing speech and picture input convolutional pooling and fully connected layers are some of the layers that make up the CNN architecture. The speech signal which is a continuous waveform is the input for speech recognition. The short-time Fourier transform (STFT) method is used to convert the waveform into a spectrogram. The CNN architecture is then used to extract and classify features from the spectrogram. The central part of the CNN design is the convolutional layer which extracts local information from the input signal by performing a convolution operation. A feature map is created by each convolutional layer's numerous filters sliding across the input signal. By lowering the feature maps dimensionality, the pooling layer aids in avoiding overfitting. All of the neurons from the preceding layer are connected to the output layer which generates the final output by the fully connected layer. A CNNs design is made up of multiple filter layers applied to an input signal. Every filter creates a matching output signal after extracting a certain feature from the input signal. Higher level features are then extracted from the output signals by passing them through an additional layer of filters. The output is then processed through a fully connected layer to create the final output when this procedure is completed and the highest-level features have been removed and input signal for end-to-end speech recognition is a voice waveform, which is usually preprocessed to create a spectrogram. After that the spectrogram is fed into the CNN which hierarchically extracts characteristics from the input signal After passing through a decoder the CNN's output is mapped to a transcript [17].

Convolutional neural networks are a particular kind of deep neural architecture that consist of one or more pairs of alternating convolutional and pooling layers. A convolution layer applies filters that process small local sections of the input and these filters are replicated over the input space. A pooling layer lowers the resolution of convolution layer activations by selecting the greatest filter activation inside a specified window and moving across the activation map. CNNs have a grid like layout and are widely utilized for data processing. They are variants of neural networks

that are fully connected. Examples of grid-like structures are time-series data (1D grid) with samples at regular intervals or images (2D grid) with pixels. The spectrogram representation shows highly important frequency and temporal relationships. Because of its characteristics the spectrogram is a useful input for a CNN processing pipeline that must preserve locality in both the frequency and time axes. CNNs will be helpful in speech signal modeling of local correlations and the CNNs can also effectively extract the structural elements of the spectrogram and streamline the model by sharing weights [1].

### 2.7.2. Recurrent Neural Network (RNN)

A recurrent neural network is a kind of network that uses a series of prior or subsequent states to predict a future element this can be viewed as a type of memory that considers context rather than the input signals current state and an expertly trained recurrent neural network may forecast the subsequent transcripts in a phrase or a sentences continuation with certain number of layers with recurrent neural network cells make up these recurrent neural networks [37]. DNNs have the disadvantage of requiring a fixed dimensionality vector and a corporation mapping of sequential data is not always possible in a general sense and architectures that are specialized in modeling the data per timestep such as LSTM and RNNs have emerged [38].

All of the hidden layer's outputs are fed back to the input layer in the RNN three-layer network topology. With the outputs of the hidden layer at any one time being impacted by a complicated combination of all previous inputs this kind of RNN operates as a dynamic system. This feature helps distinguish between different speech patterns by enabling it to efficiently capture the dynamic features of the incoming speech signal [39]. RNNs are very good at processing sequential data in end-to-end training of RNNs for sequence labeling tasks is made possible by methods such as Connectionist Temporal Classification even in situations where the input and output alignment is unknown [40]. By preserving an internal state or memory of previous inputs RNNs are a kind of neural network that can process sequential data. For jobs involving sequential data such language modeling translation and speech recognition this makes them especially helpful. RNNs fundamental concept is to use the previous steps output as the input for the current phase which enables the network to identify temporal dependencies in the data.

#### 2.7.2.1. Long Short-term Memory (LSTM)

In 1997 Hochreiter and Schmid Huber developed LSTM. Researchers made it popular and widely used by improving it and using it to solve a wide range of important problems in a variety of industries. Its application has also grown. LSTM is a robust type of RNN which is one way to address the problem of long-term dependence in recurrent neural networks [41]. In speech recognition tasks LSTM, a type of RNN has been widely used. Speech signals might differ due to accents background noise and speaking styles making speech recognition challenging. By maintaining long-term information and recognizing the temporal correlations in speech signals LSTM can overcome these challenges [3]. Networks of Long Short-Term Memory Sequence prediction issues have existed for a while. In the field of data science, they are regarded as one of the most challenging issues to resolve. These involve a variety of issues such as forecasting sales identifying trends in stock market data comprehending movie plots identifying your speech patterns translating languages and anticipating your next word on the iPhone keyboard.

Long-term and short-term time dependency issues are specifically avoided by LSTM architecture. Long-term memory retention or forgetfulness is essentially their default behavior [42]. During speech recognition LSTM usually receives a series of acoustic features like MFCC or log Mel filter bank energies as input. The spectral characteristics of the speech signal are represented by features extracted from the raw audio signal. The LSTM analyzes the series of acoustic characteristics and generates a series of likelihoods for each phoneme or word in the speech signal. One more benefit of LSTM is its capacity to grasp extended relationships in speech signals. Speech signals contain a temporal organization and knowledge from previous frames may aid in forecasting the present frame. LSTM has the ability to capture these relationships and utilize them to enhance the precision of speech recognition. LSTM has been applied in different speech recognition activities such as phoneme identification word identification and converting speech to text. The objective in phoneme recognition is to categorize each section of the speech signal into a specific phoneme of the language. LSTM has been proven to be more effective than traditional HMM-based models in phoneme recognition tasks. The buried layers distinct segments referred to as memory blocks make up the basic structure of LSTM. Cells and input and output gates make up the first kind of LSTM block. A forget gate that will enable LSTM to modify its state was established in order to overcome a constraint in the standard structure of LSTM [3].

Although the input and output gates regulate the reading of inputs from the feature vector $x_t$ and writing of output to $ht$ respectively the "forget gate" ft resets the cell variable making the stored input $ct$ "forgotten". The gates control the memory blocks activity while the "forget gate" balances the data within the cells so that it will reset the states of the various cells once previously stored information is no longer relevant for some of them. By forcing cells to forget their prior states "forget gates" also allow for continuous prediction while limiting prediction biases. An LSTM block computing procedure operates as follows Only when allowed by the input gate are input values kept in the cell state. The following formula is used to determine the value of the input gate and the candidate value for the memory cell $\tilde{c}_t$ at time step t is calculated as follows.

$$f_t = \sigma(w_f [h_t - 1 , x_t] + b_f) \quad (1)$$

$$i_t = \sigma(w_i [h_t - 1, x_t] + b_i) \quad (2)$$

$$\tilde{c}_t = \tan h\,(w_c [h_t - 1, x_t]) + b_c \quad (3)$$

$$c_t = f_t * c_t - 1 + i_t * \tilde{c}_t \quad (4)$$

$$o_t = \sigma(w_0 [h_t - 1, x_t] + b_0) \quad (5)$$

$$h_t = o_t * \tan h(c_t) \quad (6)$$

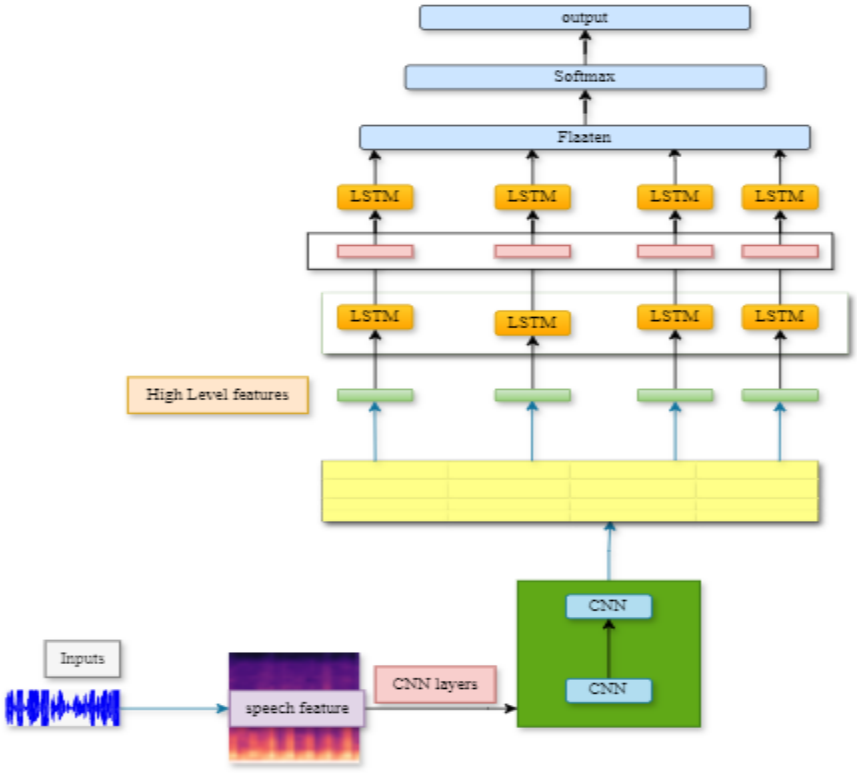



Figure 2.3. Lstm Architecture

### 2.7.2.2. Bi-directional Long Short-term Memory (BILSTM)

By taking into account both past and future context in sequence modeling tasks, Bidirectional Long Short-Term Memory (BILSTM), an extension of the LSTM architecture, overcomes the drawbacks of conventional LSTM models. BILSTM gets around this restriction by training the model in both forward and backward directions, whereas conventional LSTM models only process input data in the forward direction [43].

Bidirectional Long Short-Term Memory (BILSTM) is a modified version of the conventional LSTM design and is commonly applied in speech recognition assignments. BILSTM is very effective for capturing extended dependencies in speech signals by analyzing the input sequence in both forward and backward directions.

In classic LSTM, the input features' sequence is handled in a unidirectional manner, either from past to future or from future to past. Nevertheless, this one-way processing might overlook crucial data that is found in the reverse direction. BILSTM tackles this problem by handling the input sequence in two directions, allowing it to capture long-term dependencies in both directions. The

basic architecture of a BILSTM cell consists of two LSTM layers, one processing the sequence in the forward direction and the other processing it in the backward direction. The output of each Multiple layers is joined together to create the ultimate result of the BILSTM cell. The weights of both LSTM layers are acquired through backpropagation during the training process. BILSTM has been proven to enhance the precision of speech recognition assignments by capturing long-range dependencies in forward and backward directions. BILSTM can enhance phoneme and word recognition accuracy by analyzing the input sequence in forward and backward directions, allowing for better context understanding in the speech signal.

BILSTM has demonstrated superior performance over conventional unidirectional LSTM and HMM models in phoneme recognition tasks. BILSTM can improve its ability to understand the variations in speech signals due to various speaking styles, accents, and background noise by considering long-term connections in both directions. BILSTM has been utilized along with Language Models (LM) in speech-to-text transcription to enhance transcription accuracy. BILSTM is able to enhance the accuracy of transcribing sentences of varying lengths by capturing long-term dependencies in both directions and effectively modeling the variability in speech signals. An obstacle in utilizing BILSTM for speech recognition is the higher computational expense caused by analyzing the input sequence in two directions. Yet, this issue can be reduced by implementing methods like batch normalization and dropout.

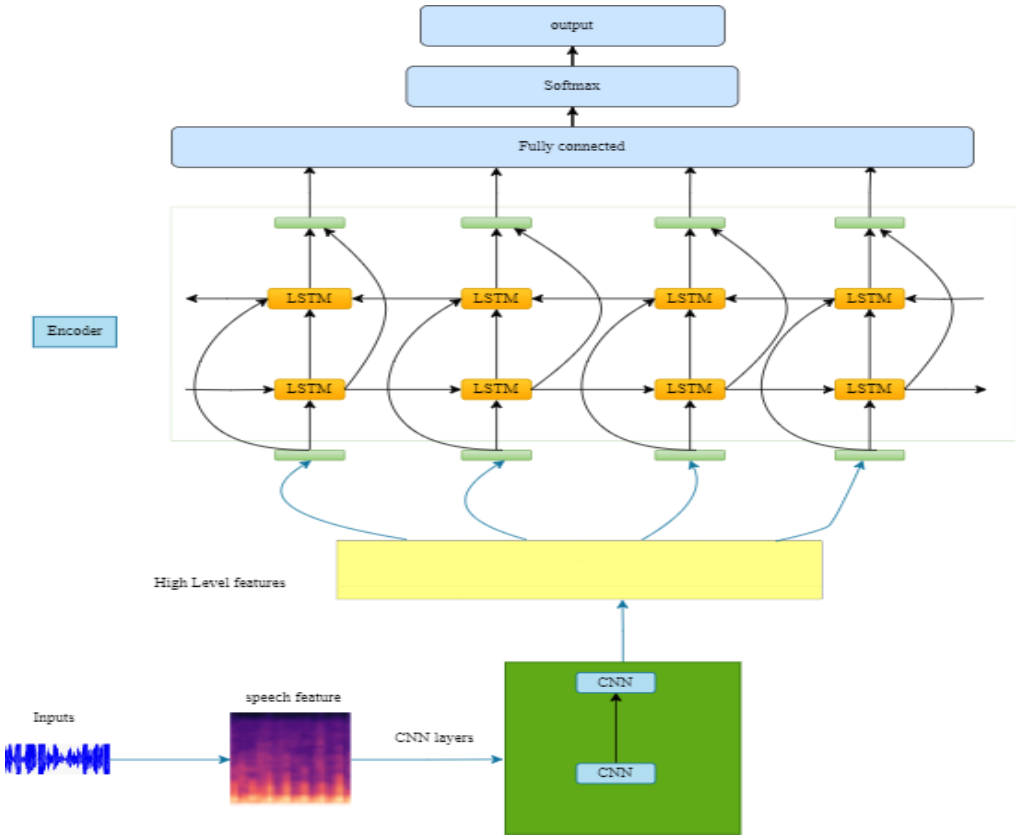



Figure 2.4. Bilstm Architecture

### 2.7.2.3. Gated Recurrent Unit (GRU)

Cho et al. initially suggested GRU as a way to get the recurrent unit to capture the long-term dependence. The GRU features a gating mechanism to adjust the information flow through the unit just like the LSTM unit [44]. Compared to LSTM, GRU has a simpler internal structure and requires fewer calculations making it easier to train. There are two main methods used to accomplish these simplifications. The update gate is the result of combining the input and forget gates into one single gate. The concealed state and the internal cell state are combined [45]. The vanishing gradient issue of a typical recurrent neural network is intended to be resolved by GRUs. It operates by means of an update gate and a reset gate. For some smaller datasets GRUs perform better than LSTM. GRUs require less data to generalize and can be trained more quickly [46].

The GRU shields a recurrent neural network from the vanishing gradient issue just like LSTM does. In certain situations, GRU is a condensed form of the LSTM cell has been demonstrated to provide outcomes comparable to those of LSTM. However, because GRU does not have an output

gate it requires less computing power. It has a single memory vector rather than two distinct ones. Thus, the same location is used to store both short-term and long-term memory. It just has a single update gate rather than an input and a forget gate [37].

Sequential data is learned using a GRU. GRU, which comprises of two gates the reset gate and the update gate manages the input weights to solve the vanishing gradient that exists in RNN. The reset gate is in charge of figuring out how much historical data should be discarded. The update gate is in charge of deciding how much historical data should be sent to the current memory the final memory and the future at any given moment [47]. As a result, compared to LSTMs, GRU contains a few fewer parameters. Because GRUs are easy to employ they save memory or calculation time and are frequently utilized in sequence learning applications. In certain, the standard GRU architecture is defined by the following equations, where $rt$ and $zt$ are the vectors of the reset and update gates respectively, while ht represents the state vector for the current time frame $t$.

$$z_t = \sigma(w_z x_t + u_z h_t - 1 + b_z) \quad (1)$$

$$r_t = \sigma(w_r x_t + u_r h_t - 1 + b_r) \quad (2)$$

$$\widetilde{h_t} = \tan h\,(w_h x_t + u_h\,(h_t - 1 \odot r_t) + b_h) \quad (3)$$

$$h_t = z_t \odot h_t - 1 + (1 - z_t) \odot \widetilde{h_t} \quad (4)$$

The "⊙" function denotes Element-wise multiplications. The activation functions of both gates are logistic sigmoid σ. By this way, $z_t$ and $r_t$are constrained to take values ranging from 0 and 1. The candidate state $\widetilde{h_t}$ is processed with a hyperbolic tangent. The current input vector $x_t$ (e.g. speech features) feeds the network and the matrices, $w_z$, $w_r$, $w_h$ (the feedforward connections) and $u_z$, $u_r$, $u_h$(the recurrent weights) represents the parameters of the model. Trainable bias vectors, $b_z$,$b_r$ and $b_h$ are added before the nonlinearities are applied [48].

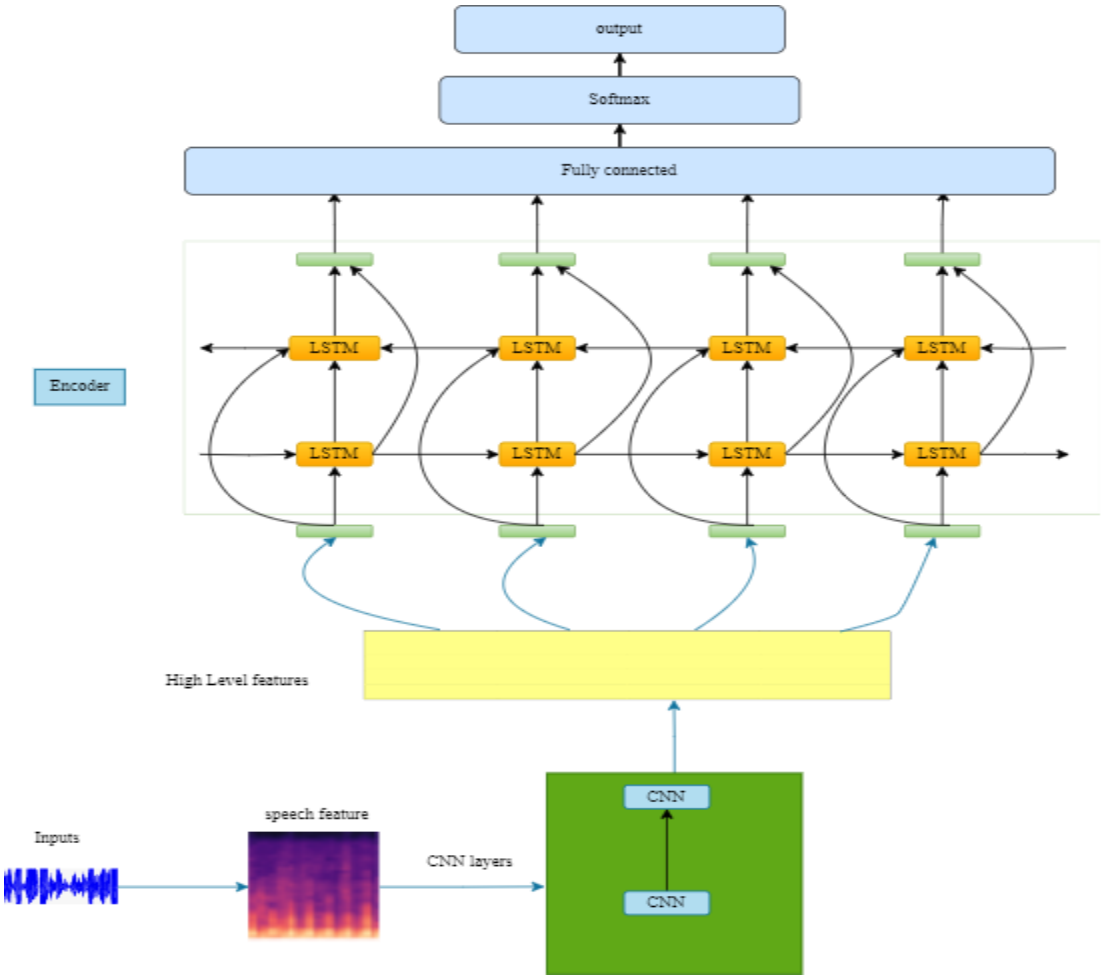



Figure 2.5. Gru Architecture

### 2.7.2.4. Bi-directional Gated Recurrent Unit (BIGRU)

A Bidirectional gated recurrent unit (BIGRU) is a sequence processing model made up of two gated recurrent unit one processes the input in the forward direction while the other does so in the backward direction. It functions as a bidirectional recurrent neural network, utilizing only the input and forget gates. Bidirectional GRUs use both forward and backward hidden layers to process data in both directions. The number of free parameters doubles in comparison to the unidirectional case. The output is therefore a concatenation of the results from both directions [49].

The temporal features will be obtained using two layers of the BIGRU. The last layer generates a character probability for each time step after the first BIGRU layer absorbs the characteristics that collected [50]. By combining past and future contexts in sequential modeling problems the BIGRU enhances the traditional GRU architecture. The BIGRU handles input sequences both forward and

backward unlike the traditional GRU which only handles forward input sequences. This is accomplished by using two concurrent GRU layers, one of which processes the input data in reverse and the other forward [43].

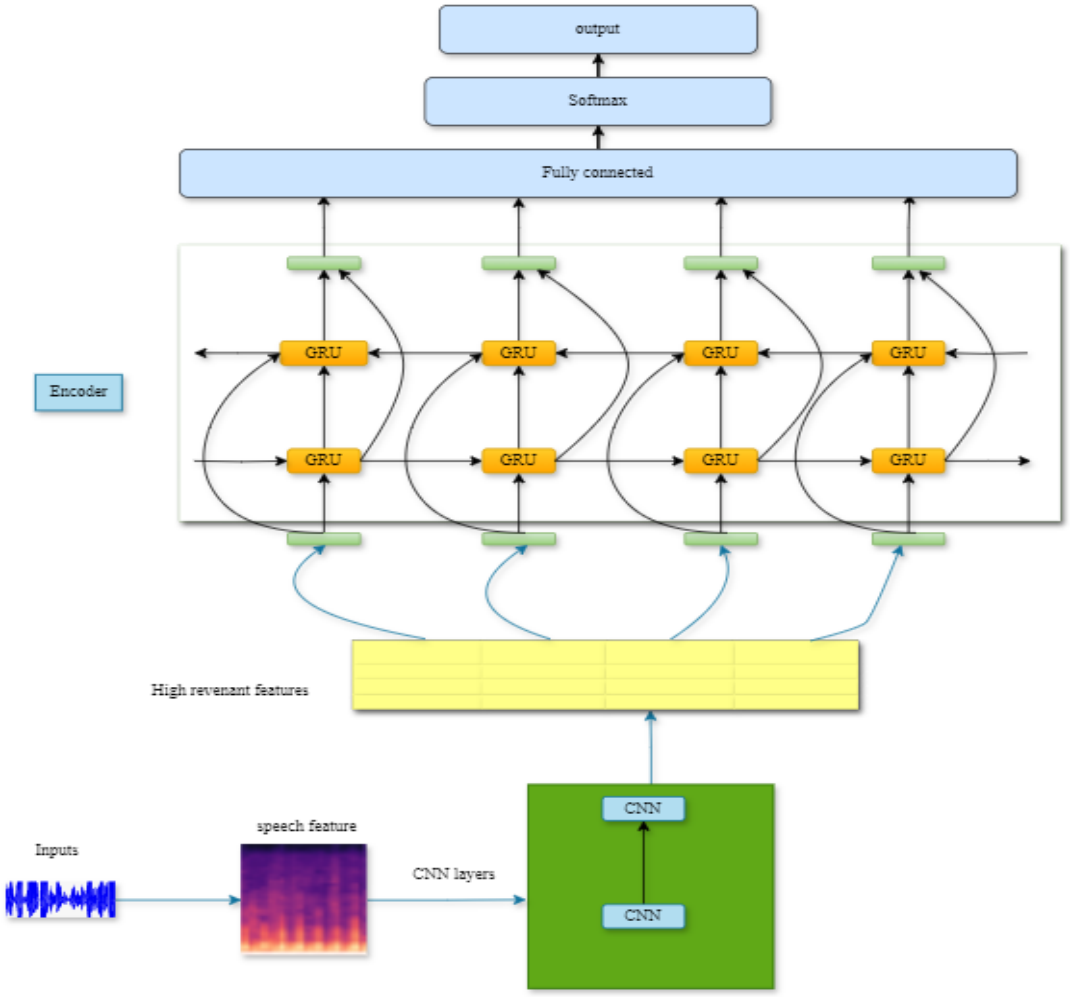



Figure 2.6. Bigru Architecture

## 2.8. Overview Of Guragigna Language

Guragigna has a phonological system that includes seven vowel phonemes and a wide variety of consonants. The language permits clusters of consonants in both start and ending places as well as complicated syllable patterns. The penultimate syllable usually receives the most stress, and intonation is crucial to meaning.

Guragigna has a primarily subject-verb-object word order noun and verb conjugation and adjective agreement with nouns. Originally written in the Geez script the language is now frequently written

in the Ethiopian script or abugida. For the Gurage people Guragigna is sociocultural significant since it is entwined with their identity customs and traditional stories [51].

Cheha is written using the Geez (Ethiopic) script which was initially created for the now-extinct Geez language and is currently used for Amharic and Tigrinya. Although there are limited texts in Cheha the script has been adapted with modified characters to represent palatalized consonants absent in Geez, Amharic and Tigrinya such as the addition of wedges at the tops of letters. This adaptation originated with the Ethiopian Bible Society's publication of the New Testament and later extended to the full Bible becoming widely accepted.

The phonetic structure of Cheha is typical for Ethiopian Semitic languages featuring plain voiceless and voiced consonants as well as the common ejective consonants. However, Cheha distinguishes itself from many other Ethiopian Semitic languages by having a larger inventory of palatalized and labialized consonants.

In addition to the standard seven vowels, Cheha includes open-mid front (ɛ) and back (ɔ) vowels. Some dialects have nasalized vowels while others exhibit both long and short vowel sounds [52]. The phonetic details of Cheha are complex and while this article uses a modified system for representing its sounds which is popular among linguists studying Ethiopian Semitic languages it does diverge slightly from traditional International Phonetic Alphabet (IPA) conventions. Any variations from the IPA symbols are noted in the accompanying charts.

Table 2.1 Cheha Consonants

| | | Labial | | Dental | Postalveolar | Palatal | Velar | | Glottal |
|---|---|---|---|---|---|---|---|---|---|
| | | plain | round | | | | plain | round | |
| Nasal | | m | $m^w$ | n | | | | | |
| Plosive/ Affricate | voiced | b | $b^w$ | d | d͡ʒ ⟨ǧ⟩ | ɟ ⟨$g^y$⟩ | g | $g^w$ | |
| | voiceless | p | $p^w$ | t | t͡ʃ ⟨č⟩ | c ⟨$k^y$⟩ | k | $k^w$ | |
| | ejective | | | t’ ⟨ṭ⟩ | t͡ʃ’ ⟨č̣⟩ | c’ ⟨ḳ$^y$⟩ | k’ ⟨ḳ⟩ | k’$^w$ ⟨ḳ$^w$⟩ | |
| Fricative | voiced | | | z | ʒ ⟨ž⟩ | | | | |
| | voiceless | f | $f^w$ | s | ʃ ⟨š⟩ | ç ⟨$x^y$⟩ | x | $x^w$ | h |
| Approximant | | β̞ | | l | | j ⟨y⟩ | | w | |
| Rhotic | | | | r | | | | | |

Table 2.2 Cheha vowel

| | Front<br>Central<br>Back | Front<br>Central<br>Back | Front<br>Central<br>Back |
|---|---|---|---|
| High | i | ɨ ⟨ə⟩ | u |
| High-mid | e | | o |
| Low-mid | ɛ | | ɐ ⟨ä⟩ |
| Low | | a | |

## 2.9. Related Works

End-to-end speech recognition is a technique where the whole pipeline from the acoustic signal to text output is modeled using a single neural network, without the need for any intermediate feature extraction or language modeling steps. This approach has gained significant popularity in recent years due to its simplicity and effectiveness. In this literature review, we have discussed about speech recognition generally and the recent advances in end-to-end speech recognition research.

### 2.9.1. ASR for non-Ethiopia language

According to [53]in this research, the authors demonstrate Automatic Speech Recognition (ASR) for low-resource languages, specifically focusing on Tamil. They present a method to develop a speech recognition model with minimal resources using the Mozilla Deep Speech architecture. Their results show that the ASR model achieves an impressive Word Error Rate (WER) of 24.7%, significantly outperforming Google's speech-to-text service, which has a WER of 55%. Additionally, they illustrate a semi-supervised approach to developing a speech corpus using their trained ASR model, highlighting a cost-effective method for building a large vocabulary corpus for low-resource languages. Furthermore, the trained Tamil ASR model and the associated training sets have been released into the public domain and are available on GitHub, promoting accessibility and further research in this area.

According to [54] presented a novel end-to-end speech recognition model named Conformer, which combines convolutional neural networks (CNNs) with transformers. This advanced architecture includes a convolutional layer before the transformer layers, improving the model's capacity to identify local audio patterns. The Conformer model attained state-of-the-art results on major benchmarks, such as LibriSpeech and WSJ, showcasing its efficacy in automatic speech recognition applications.

### 2.9.2. ASR for Ethiopian language

As mention in [11] In this study, the authors developed a syllable-based speech recognition system for the very low-resource language, Cheha, exploring the use of CV (consonant-vowel) syllables as acoustic modeling units. They employed Gaussian Mixture Models (GMM) alongside unilingual and transfer learning deep neural network (DNN) models. The experimental results indicate that the syllable-based unilingual DNN and transfer learning DNN models significantly outperform the corresponding GMM and unilingual DNN models, with absolute performance

improvements ranging from 2.8% to 3.09% and 1.07% to 4.94%, respectively. The best-performing syllable-based recognizer utilized a shared hidden layer (SHL) time delay deep neural network (TDNN) model, achieving a word error rate (WER) of 23.11%. These findings suggest that CV syllables are effective acoustic units for developing speech recognition systems for Cheha.

A key drawback of the syllable-based approach, compared to end-to-end speech recognition, is its reliance on more manual intervention in system design. The syllable-based method requires training separate acoustic models for each syllable in the Guragunga language, which involves significant effort in creating and labeling training data, as well as designing the system architecture to accommodate these individual models.

In contrast, end-to-end speech recognition systems are designed to learn directly from the raw audio waveform, eliminating the need for manual feature extraction or separate models for different linguistic units, such as syllables. This makes end-to-end systems more flexible and easier to adapt to various languages or speech tasks. Consequently, while the syllable-based approach may offer certain advantages in specific contexts, it is often less efficient and more labor-intensive than end-to-end methods.

According to [12] his paper investigates automatic speech recognition systems for the very low-resource language Cheha, utilizing multilingual deep neural network (DNN) modeling methods under sufficient training corpus conditions. The authors explored various basic and rounded phone unit-based speech recognizers, discovering that all multilingual models significantly outperformed their unilingual counterparts, with performance improvements ranging from 5.47% to 19.87% for basic phone units and 5.74% to 16.77% for rounded phone units. Notably, the rounded phone unit-based multilingual models surpassed the basic phone unit-based models, demonstrating relative performance gains of 0.95% to 4.98%. These findings suggest that multilingual DNN modeling methods are effective for developing Cheha speech recognizers, with both basic and rounded phone acoustic units being viable for constructing the ASR system, although the rounded phone unit-based models exhibited superior performance and faster recognition speeds.

Despite these advantages, the phone-based approach has limitations, particularly in its potential inability to adequately capture long-term dependencies and context in speech. In contrast, end-to-end systems can model the entire input sequence directly, allowing for more accurate context modeling, which may enhance the overall performance of speech recognition systems.

According to Azmeraw Dessalegn [32] The research titled "Syllable-Based Speaker Independent Continuous Speech Recognition for Afan Oromo" focuses on developing a Hidden Markov Model (HMM)-based automatic speech recognition (ASR) system using the HTK toolkit, incorporating both phone-based and syllable-based models. The phone system aids in creating a pronunciation dictionary and voice transcription. HMM-based ASR systems typically demand extensive feature engineering, which can be labor-intensive and require domain expertise, whereas end-to-end systems learn relevant features directly from the input data, enhancing efficiency and effectiveness. HMM systems model the probability distribution of state sequences based on the input, operating under the assumption of conditional independence, which may limit their ability to capture long-term dependencies and contextual nuances in speech. In contrast, end-to-end systems can model the entire input sequence directly, allowing for more accurate context modeling and improved overall performance in speech recognition tasks.

According to Tsegaye Abebe [22] The study investigated the feasibility of creating an automatic speech recognition system for the Ge'ez language using the Sphinx 4 trainer and hidden Markov modeling (HMM). Due to the absence of a prepared Ge'ez corpus, the researchers developed their own text and speech corpora, selecting audio from seven speakers to create 4,818 training sentences and 433 testing sentences. To evaluate the system, two experiments were conducted employing different language models and testing techniques (online and offline). However, the approach by Tsegaye Abebe has several drawbacks compared to end-to-end speech recognition. Firstly, HMMs for acoustic modeling require manual feature engineering, which is time-consuming and limits the system's adaptability to new data. Secondly, the lack of a large and diverse Ge'ez speech corpus could hinder accurate recognition across various contexts, such as different speakers or environments. Thirdly, using separate language models for each experiment may not be optimal, as an end-to-end system could simultaneously optimize the language and acoustic models. Additionally, the reliance on a pronunciation dictionary poses another limitation that the current research aims to avoid.

Table 2.3 Summary of Related Works

| Citation | Author | Title | Objectives | Tool | Dataset | Result | Research gaps |
|---|---|---|---|---|---|---|---|
| [55] | Mohamed Hashim Changrampadi1, A. Shahina, M. Badri Narayanan and A. Nayeemulla Khan (2022). | End-to-End Speech Recognition of Tamil Language. | They present a method to develop speech recognition model with minimal resources using Mozilla Deep Speech architecture. | they use six-layer RNN for deep speech, CTC for alignment and MFCC feature extraction. | they used 100 hours speech dataset. | 24.7% WER | They use public dataset rather than Tamil local dataset and they are not used Attention mechanism to train and test purpose. |
| [17] | Yohannes Ayana and Tesfa Tegegne (2024). | Large Scale Speech Recognition for Low Resource Language Amharic, an End-to-End Approach | To develop an end-to-end speech recognition model. | they use RNN, CTC for alignment and Spectrogram feature extraction, CNN to select high level features. | 20 hour read speech data. | achieving a WER of 2%. | They are not used encoder decoder with attention. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| [56] | Abdinabi Mukhamadiyev, Ilyos Khujayarov, Oybek Djuraev and Jinsoo Cho (2022). | Automatic Speech Recognition Method Based on Deep Learning Approaches for Uzbek Language. | End-To-End Deep Neural Network-Hidden Markov Model and a hybrid (CTC)-attention. | they use RNN, CTC for alignment and Mel-Spectrogram feature extraction, CNN to select high level features | Used Uzbek languag e corpus with 207 hours of recordin gs. | Achieved a word error rate of 14.3% using CTC+ Attention. | They use only LSTM from RNN models what about other models to compare and contrast the result of WER. |
| [57] | Alsayad, Abdelaziz, A. Abdelhamid, Islam Hegazy, and Zaki T. Fayed. (2021). | Arabic speech recognition using end-to-end deep learning. | Investigated end-to-end deep learning for discretized Arabic ASR. | They used Mel-Frequency Cepstral Coefficients Conventional ASR with Kaldi toolkit, RNN-LM. | 7 hours of modern standard Arabic speech. | CNN-LSTM with attention reduces WER by 5.24% End-to-end achieve 5.66 CER and 28.48 WER. DNN achieves 33.72 WER. | They use conventional ASR not fully RNN for End-to-end Approaches. |
| Our proposed work | | End-to-end Speech Recognition for Guragigna language in case of Cheha using Deep learning Approaches. | to develop a model that used to subscribe Guragigna language in cheha dialect into a computer-understandable text by using deep learning approaches. | We have used spectrogram feature extraction with CNN to select high level features, RNN model with Attention in CTC loss function architecture. | 15-hour speech corpus. | We have achieved CNN-BIGRU WER 2.5%. | We used RNN model like lstm, BILSTM, GRU, and BIGRU to enhance the model with Attention for experimental analysis. |

## 2.10. Summary Of Related Work

In conclusion, this review of the literature observed a number of aspects of speech recognition with an emphasis on both suggested end-to-end models and conventional speech recognition systems. The majority of the studies under review allocated with various kinds of language ASRs which usually use traditional pipelines to transcribe spoken language into text. Traditional ASRs do have some serious disadvantages though such as requiring a lot of human labor taking longer to process and having a higher chance of mistakes. Many of the problems with classic ASRs like alignment problems are intended to be addressed by our suggested paradigm. The end-to-end method increases overall efficiency by reducing time consumption and the need for additional human resources by allowing spoken words to be converted directly to text without the use of intermediary processes including Attention/CTC layer for decoding purpose not only CTC loss function. In the field of NLP this suggested approach exhibits a great deal of promise for raising the precision and effectiveness of speech recognition systems.

# CHAPTER THREE

# 3. MATERIALS AND METHODS

## 3.1. Introduction

In this chapter, we provide a comprehensive overview of the approaches, methods, tools, and techniques employed in the research on Cheha speech recognition. The study follows a structured framework to achieve both general and specific objectives, encompassing steps such as data collection of audio samples and transcripts, conducting a literature review of relevant studies, implementing specific methodologies tailored for Cheha speech processing, and preparing a comprehensive annotated corpus. Additionally, the research includes data preprocessing, training and testing the proposed speech recognition model, developing a working prototype, and validating the model's effectiveness. This chapter also offers an in-depth analysis of the techniques and algorithms utilized in constructing the speech recognition model while investigating the linguistic characteristics of the Cheha language to enhance our understanding of its phonetic and phonological structures in the realm of speech recognition.

## 3.2. Proposed Approach

To develop an end-to-end speech recognition model for the guragunga languages in case of cheha dialect first collect a speech and text corpus from various sources and then it is essential to review existing literature on automatic speech recognition to know the research gap and to give a solution for those problems. The next steps involve recording, segmenting, and aligning audio data with text transcriptions using software tools and custom code. Various data preprocessing techniques, including data cleaning, feature extraction, normalization, and labeling, should be applied, followed by converting the preprocessed audio data into waveform format. An end-to-end model must be designed and built to transcribe input audio signals directly into the corresponding target text which will then be trained using the prepared dataset. During this training process different hyperparameters should be experimented with learning hyperparameters such as learning rate, batch size, number of epochs and training/testing percentages should be evaluated. Finally, the best model for Cheha speech-to-text transcription will be selected, and results of the training must be documented.

## 3.3. Literature Review

This work does a thorough literature evaluation on speech recognition in several language domains. The ideas, methods, strategies, tools, and materials they employed are also used to search various related publications (text documents, books), journals, articles, conferences, and other literature on speech recognition, including IEE Explore, Science Direct, Spring Link, and others. Furthermore, a thorough analysis of the different implementation options will be conducted for Cheha speech recognition.

## 3.4. Corpus of cheha language

In this section, the text corpus and speech corpora, which we have used in our study the process followed to prepare them are discussed. Cheha does not have a readily available text corpus. Besides, it has limited presence on the web, and has limited hardcopy books [11]. We have compiled a substantial collection of texts from Wolkite FM as well as the Old and New Testaments. These texts have been merged followed by text cleaning processes that include correcting grammar and spelling errors expanding abbreviations removing foreign terms transcribing numbers and separating concatenated words. This process resulted in a text corpus comprising 15,000 sentences which includes 178,830 tokens and 12,500-word types, and serves as the foundation for training testing and evaluation of models. Similarly due to the absence of publicly available speech corpora for Cheha to facilitate speech recognition tasks we created a speech corpus by selecting 15,000 phonetically balanced sentences from the text corpus. We recorded 15 hours of speech in an office environment using a smartphone recorder involving 27 native speakers (16 male and 11 female) who read these sentences. Of this 15-hour corpus 12 hours and 40 minutes (12,400 sentences) were gathered from the same 27 speakers, each reading 460 sentences. This portion is designated as the training dataset. To prevent overlap between the training and testing datasets regarding speakers and sentences a separate corpus of 1 hour and 20 minutes (2,600 sentences) was recorded from 5 native speakers (5 male and 5 female), each reading 260 sentences. This testing dataset represents 20 percent of the total 15-hour corpus. However, in comparison to other speech corpora that consist of 20 hours or more of training data, our corpus is relatively small which may delay the performance of the models due to insufficient training data.

## 3.5. Tools

### 3.5.1. Hardware Tools

We used the Hardware tools such as dell laptop with Processor: Intel(R) Core (TM) i5-7320M CPU @ 2.90GHz, 4GB RAM, 1T hard disk, screen size: 15 inches, Wi Fi:4G lite.

### 3.5.2. Software Tools

Software tools are essential for implementing research, as they enhance efficiency and accuracy in various tasks. The choice of software tools depends on the subject of study and the type of research being conducted. Before making our selections, we considered several criteria to ensure the tools were appropriate. These included the availability of sufficient learning materials, such as free video tutorials and existing user experiences, as well as compatibility with machines that have limited resources, like those relying solely on CPU. For our research, we chose Python as the programming language and utilized TensorFlow and Keras libraries to implement the neural layers. These tools met all our criteria and were compatible with Python, a language we are familiar with. Additionally, if we were working with a specific corpus format, we ensured that the functions for reading and writing that format were organized together for ease of use.

**Google Collab**: is based in the cloud platform that offers a free environment for Jupiter notebooks for running Python code. It is a Google has provided that users to run Python code using Google computational resources including GPUs and TPUs without the requirement for any local hardware or software installation and Integration with Google Drive Google Collab and Google Drive are integrated enabling to access notebooks and data from anywhere. While Google Collab is popular platform for running python code online and provides basic python environment some commonly used libraries like Tensor flow, Pandas, NumPy and Librosa etc.

**TensorFlow:** The various network designs in this study are implemented using TensorFlow an open-source software library developed by the Google Brain Team Launched in 2015 TensorFlow has quickly gained popularity as one of the leading frameworks for machine learning and neural networks. The name TensorFlow is derived from tensors which refer to multi-dimensional matrices that traverse through the various mathematical computations carried out by the layers within a neural network [37].

**Keras**: is a user-friendly high-level Python library designed for deep learning that operates on top of TensorFlow (as well as Theano or CNTK). It enables developers to concentrate on core deep learning concepts such as building layers for neural networks while managing the complexities of tensors their shapes and associated mathematical operations. Keras requires one of these back-end libraries to function TensorFlow, Theano or CNTK allowing users to engage in deep learning tasks without needing to research into the more complex aspects of these frameworks. Keras offers two primary frameworks these are Sequential API and the Functional API. The Sequential API is centered around a linear stack of layers making it the most commonly used and the simplest aspect of Keras. This model allows users to create neural networks in a straightforward sequential manner [58].

**Librosa**: is a Python tool for music and audio analysis. It provides the essential building blocks needed to create music information retrieval systems [59].

**Diagrams.net**: is a no charge online software tool to draw system architecture, flowcharts and visual representations of information. It includes drag and drop capabilities and various formatting options to create a professional looking diagram. Diagrams created in the past can still be loaded and edited in the current version of the application. It also provides various integrations with popular platforms, such as google drive. In this section we are discuss software tools that used in this research work particularly tool used in the experiment section. In the following table tools used are listed.

Table 3.1 Tools and materials

| **Tools** | **Purpose** |
|---|---|
| Audacity | For data pre-processing such as segmentation and noisy reduction. |
| Python | Used to text file manipulation and such as text cleaning and tokenization. |
| Anaconda with integrated Python | To implement and run model code. |
| Microsoft office word | To write documentation. |
| Microsoft PowerPoint | Prepare presentation PowerPoint. |
| Zotero | For reference Citation. |
| Google Collab | To compile and execute Python code. |
| Google Drive | To upload dataset. |

## 3.6. Data collection and pre-processing

In this research data is collected from different source like bible text book and soon on after that I recorded each sentence from recorder the segmenting each audio parallel with sentence by using audacity tools. The first step to prepare speech corpus is selecting the source and scope of text, we use for preparing speech corpus. In this research, we prepare text corpus to include bibles, Fm radio and cultural information. This makes our speech recognizer inclusive and can be used for different purposes. The text corpus contains bibles, Fm radio and cultural information. After text corpus is prepared, the speakers read the text to prepare the speech corpus. The text corpus is prepared from chaha dialects. Text corpus is categorized based on the script content of words relating to these dialects. To record sound speech, we use smart phone recorder. The environment in which the speech is recorded in a semi open area with less noisy environment. Therefore, the recoded speech is not resistant from noise. The speakers are asked to read paragraph by paragraph; if they make mistakes while reading the text, they reread the paragraph again. In general, the recorded speech data is presented in table. In general, this speech data preparation phase for this research work is discussed in the following subsection.

Table 3.2. Source and size of the speech data collected for experiment

| **Training data** | | | | | | | | **Testing data** | |
|---|---|---|---|---|---|---|---|---|---|
| id | age boundary | | speech time duration | | | Total time duration | Total utterance | Male numbers | Female numbers |
| | | male | female | male | female | | | | |
| 1 | [13-20] | 5 | 5 | 3:01:55 | 2:00:35 | 5:02:30 | 3504 | 5 | 5 |
| 2 | [21-35] | 4 | 6 | 3:12:04 | 3:13:32 | 6:25:36 | 4042 | | |
| 3 | [36-50] | 3 | 4 | 2:33:14 | 1:30:11 | 4:03:25 | 4522 | | |
| 4 | Total speaker | 27 | | | | | | 10 | |
| 5 | Total time | 15:30:33s | | | | | | 3:30:33s | |
| 6 | Total utterance | 12,068 | | | | | | 3000 | |

### 3.6.1. Text Data Pre-preprocessing

Data preprocessing the first steps in speech recognition task to easily fetch data for deep learning model to train and test with the best and robustness of the experimental results. By lowering the noise in raw textual data text pre-processing has a major impact on the entire mining process and is an essential stage in many texts cleaning activities. Improper pre-processing execution can result in inconsistent and confusing outcomes. Thus careful text pre-processing is necessary to guarantee the dependability and quality of big corpora [60].

### 3.6.*2*. Text cleaning

Data cleaning is an important step in data pre-processing where errors and inconsistencies in the dataset are addressed. It involves tasks like handling missing values by imputing them dealing with

outliers to remove extreme values correcting inconsistent data formats, removing duplicate entries and standardizing data formats. By performing data cleaning researchers can improve the quality and reliability of the dataset ensuring that it is accurate complete and ready for analysis. This process helps to enhance the integrity of the data and prevents biases or errors that could affect the outcomes of subsequent analyses and modeling.

### 3.6.3. Sentences Tokenization

Tokenize sentences We divide a document or paragraph into sentences using the supplied tokenize () method. The re. split () method can be used to tokenize sentences. By introducing a pattern into the text, this will divide it into sentences [61].

## 3.7. Speech Corpus Recording

To record sound speech, we use smart phone recorder. The environment in which the speech is recorded in a semi open area with less noisy environment. Therefore, the recoded speech is not resistant from noise. The speakers are asked to read paragraph by paragraph if they make mistakes while reading the text they reread the paragraph again and labeling parallel with transcription of text file.

## 3.8. Speech pre-processing

In speech recognition gathering and analyzing data is the first significant effort. Audio utterances and their transcriptions make up the speech corpus utilized for training. The initial step in the speech signal processing technique is pre-processing. This research discusses preprocessing in speech recognition, which covers normalization, noise reduction and silence removal.

Figure 3.1 Original Audio

### 3.8.1. Silence Removal

The technique of eliminating unvoiced or silent speech signals is known as silence removal. Different people take varying amounts of time between words when speaking. A distinctive trait that distinguishes one speaker from another is crucial to the speaker recognition process. Silence is not necessary for recognition because it is a characteristic shared by all speakers [62]. Therefore, the speech signals should no longer include silence. Because understanding where speech and silence or unvoiced words belong not only helps the speech processing system handle information more efficiently but it also improves its accuracy. Software called an audio silence trimmer is used to eliminate the quiet at the start middle and finish of voice signals. The software was chosen due to its ease of use and free nature. In this case the duration is shortened by eliminating the silence. Each speech signal wav file was kept to a minimum in duration, ranging from 10 to 15 seconds, based on the length of the silence. Using an online audio trimmer 10 seconds of speech have been cut from each speaker's voice signals to make them all the same length.

Figure 3.2 After silence removal

### 3.8.2. Noise Reduction

Every speech has background noise since the speech signals are obtained from many environments and are public utterances. To obtain the speakers precise unique traits these background noises should be eliminated. Therefore, noise reduction and normalization come next once the speech signals silence has been eliminated [62].

Figure 3.3 After Noise reduction

### 3.8.3. Noise Normalization

Applying a fixed amount of gain to an audio recording in order to raise the amplitude to a desired level (the norm) is known as normalization.  because the voice signal's volume is reduced during noise reduction. Normalization restores the volume to its pre-noise reduction level. Software called Audacity is used to normalize and cut down on background noise in speech signals.  Audacity was chosen since it is open source and among the top ten (3rd) audio editing programs [63].

Figure 3.4 After normalization

### 3.8.4. Segmented Speech Parallel with Text Transcription

After recording and cleaning completed, we are going to segmenting each audio parallel with text transcription by using audacity software tool.

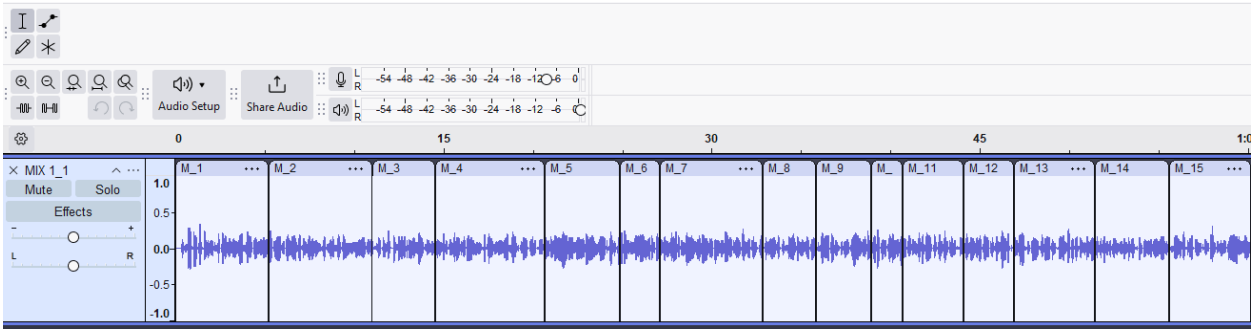



Figure 3.5 Audio segmentation techniques

Speech data after segmentation export in to single folder by give file name with extension just like this

"C:\Users\abdo\Desktop\1-600\segmented.wav\AbeM_00021.wav"

"C:\Users\abdo\Desktop\1-600\segmented.wav\BetF_00051.wav"

"C:\Users\abdo\Desktop\1-600\segmented.wav\DawM_00061.wav"

"C:\Users\abdo\Desktop\1-600\segmented.wav\DawM_00061.wav"

Every speech recording had a transcription made for it. This required paying close attention to the audio and turning it into text. The deep learning model can learn the connection between the audio attributes and the related text by using this text data as the ground truth.

A transcription file is a written document that summarizes the speakers' words from the audio recording. The audio file name is followed by the colon for each sentence in the transcript file. The matching transcription sentence appears after the filename.

Example:

NesF_00001.wav: "**የፍጥረት መጣፍ አለም የትፈጠሮ ኽማ የሰብ ዘር የቾነ ኽማ**"

NesF_00002.wav: "**የባጢርም የጅጓረም ንብረት የቄነሺኽማም እግዘር ተሰብ ዘር ጋሙ ያነን መራኽብ ዩድ**"

NesF_00003.wav: "**የፍጥረት መጣፍ በሑት ንቅ ንቅ መደር ይሸጅዬ ይኽር**"

NesF_00004.wav: "***ተመንሽ አት ቄነሽም አስራት ስን አለም የትፈጠሮ ኽማም የሰብ ዘር ይፍት ወረር ታሪክ ዩድ*".**

For this study 12,000 labelled cheha language sentences were used.

## 3.9. Feature Extraction

Feature extraction is a crucial step in the speech recognition process. Raw audio data is not directly used in the model training process due to the presence of noise and other unnecessary parameters that can affect the models accuracy [64].

### 3.9.1. The Short-Time Fourier Transform

A signal processing method called the Short-Time Fourier Transform (STFT) is used to examine a signal's frequency composition over time. STFT divides the signal into overlapping segments and calculates the Fourier transform for each segment, in contrast to the conventional Fourier transform, which analyzes the full signal at once. A spectrogram a time-frequency representation that shows how the signals frequency content changes over time is the end product of this procedures is crucial in end-to-end speech recognition as it transforms audio signals into spectrograms which serve as inputs for neural networks. These networks excel at processing numerical data, and spectrograms provide a compact and informative representation of an audio

signal's frequency content. By applying STFT we can extract important features relevant to speech recognition such as formants and pitch. These features can then be utilized to train neural networks for speech recognition either through direct mapping from spectrogram to text or via more complex architectures incorporating additional layers such as convolutional or recurrent layers. In summary, STFT is a vital tool for converting audio signals into spectrograms facilitating their use as inputs for neural networks in speech recognition tasks. It enables the extraction of relevant features from audio signals and offers a concise representation that can be efficiently processed by neural networks.

## 3.10. Padding

Padding is used to make sure that all input and output sequences are equal in length which mean that after feature extraction and selection, the next steps are padding values of variable length makes equal length by adding zeros pre and post of the value that is used to encode neural networks.

## 3.11. Encoder

The encoder architecture consists of a CNN-RNN combination. The use of a CNN as the initial layer enhances ASR performance by applying convolutions across both frequency and time domains on spectral input features. Additionally, employing RNNs layers in speech recognition allows for better context utilization by considering information from both past and future sequences which improves word prediction accuracy. The encoder receives normalized spectrograms of audio clips as input. Each audio clip is represented as a time series of length $T$ with a vector of audio features for each time slice. The input vectors $V1$, V2 …, V T are generated through the 2D convolution layers. The capabilities of the encoder are further enhanced using the CTC loss function [65].

## 3.12. Decoder

The last stage of the speech recognition pipeline decoding is in charge of translating the anticipated probabilities into the final output sequence. The Beam Search method is used in this models decoding process to determine the most likely output sequence given the input sequence [17]. The CTC loss function trains the model while the beam search maintains a limited number of top scoring sequences at each step reducing the search space and enhancing decoding speed and

accuracy. This combination makes the CTC beam search decoder especially when paired with attention mechanisms an effective approach for speech recognition and other sequence-to-sequence applications.

### 3.12.1. Attention Mechanism

This attention mechanism, which can be defined as a selection mechanism for allocating limited information processing capabilities can be applied to the field of deep learning because the human brain perceives the world through a process of selective concentration. It facilitates speedy analysis of the target data and works in tandem with the information screening and weight setting mechanisms to increase the model's computational capacity [66]. The attention-based encoder-decoder (AED) model is another component of E2E ASR models. The encoder network performs the same function as the encoder network in CTC, converting input feature sequences into high-level hidden feature sequences. A context vector is formed as a weighted sum of the encoder outputs after the attention module calculates attention weights between the preceding decoder output and the encoder output of each frame using attention functions like additive attention or dot-product attention [67]. An attention-based encoder-decoder network called Listen, Attend, and Spell (LAS) is frequently used to address mapping issues with variable-length input to variable-length output. An embedding or higher-level feature representation is taken from the input features by the encoder (the Listen module). A context vector is then produced when the attention mechanism (the Attend module) decides which encoder features need to be attended in order to forecast the subsequent output symbol. Lastly the decoder (the Spell module) creates a prediction for the subsequent output by using the attention context vector and an embedding of the prior prediction [68].

### 3.12.2. Connectionist Temporal Classification (CTC) Loss

The model is trained using Connectionist Temporal Classification (CTC) as the loss function. Applications such as speech and handwriting recognition use CTC an output and scoring function that solves sequence problems when the alignment between the input and the output is unknown. For each time step $t$ and a single input sequence $X$ of length $T$ the encoder gives a distribution over the vocabulary $pt(c|X)$ then CTC computes the probability for a single sequence $C$ of length $T$ as follows.

$$P(C|X) = \prod_{t=1}^{T} p\,(C_t|X)$$

Finding the most likely sequence involves adding up the probabilities of the various sequences that can represent the same word.

$$P(S|X) = \sum P(C|X)$$

Lastly, to expedite the computation, the dynamic programming approach is used to compute CTC loss which is the negative log probability of all valid sequences. Additionally, to compute its derivative which updates the encoder's parameters by using the backpropagation through time procedure of CTC loss function.

$$L_{CTC} = -\log P(S|X)$$

## 3.13. Optimization Algorithms

A machine's ability to learn from experience is based on optimization algorithms. In an effort to decrease the loss function they compute gradients. For our work end-to-end speech recognition, the Adam optimizer is recommended because of its efficiency in managing complex models and big datasets. By adjusting the learning rate for every parameter separately it combines the advantages of the AdaGrad and RMSProp methods. It is appropriate for voice recognition tasks because of its adaptive learning rate adjustment which facilitates effective optimization and convergence. The RNN was trained with using the Adam learning rule, End-to-end speech recognition training data usually comprises of text and audio data in pairs. Regularization and training hyper-parameters were altered during the experiment run and the experiment was restricted to training a single network due to the training process slow convergence. Therefore, rather than an exhaustive evaluation of the suggested model's performance the results should be regarded as an initial proof of concept validation.

## 3.14. Performance Measurement Metrics

### 3.14.1. Accuracy

The accuracy metric is used to monitor the model's performance during training. It gives an indication of how successfully the model can categorize or predict the right output given the input data. We also used this measurement of accuracy to assess the quality of the recognition in the transcription process. It is computed computationally by adding the errors for each sample

throughout the training set. Each batch of data is used to calculate training loss, which is then plotted as a curve.

### 3.14.2. Loss

A loss function is a vital component used to measure the dissimilarity between the predicted output of a speech recognition model and the ground truth annotations. It quantifies the discrepancy between the predicted transcriptions and the correct ones in the training data. This function guides the model in updating its internal parameters to optimize performance during training. Common loss functions in speech recognition include Connectionist Temporal Classification (CTC) loss and cross-entropy loss. These functions calculate the difference between predicted probabilities of phonemes or words and the true labels, encouraging the model to minimize errors and enhance its accuracy in recognizing speech. By minimizing the loss function, researchers aim to train the speech recognition model to produce transcriptions that closely align with the ground truth, enabling it to generalize effectively to unseen audio data and deliver accurate recognition results.

### 3.14.3. Word Error Rate (WER)

Our end-to-end speech recognition model performance is evaluated in WER and is often described in terms of speed and accuracy. While speed is measured using the real-time factor accuracy can be determined using performance accuracy which is often graded using Word Error Rate (WER). Command Success Rate (CSR) and Single Word Error Rate (SWER) are additional accuracy metrics [4]. When assessing the effectiveness of machine translation and speech recognition model one often used statistic is WER. The recognized word sequence may be shorter than the reference word sequence which presents a significant measurement issue. Working at the word level as opposed to the phoneme level WER is based on the Levenshtein distance. This problem is solved by first applying dynamic string alignment algorithms to align the recognized word sequence with the reference word sequence. This is how the WER can then be computed.

$$(\text{WER}) = \frac{\text{Inserted} + \text{Deleted} + \text{Substituteded}}{\text{Total words in Transcript}}$$

## 3.15. Regularization techniques

### 3.15.1. Dropout

The dropout rate parameter can be used to prevent overfitting. Nevertheless, an excessive dropout rate may result in the network losing significant weight values which would lower the performance of the final model. But just during the training phase, an excessively low dropout rate can result in models that are overly intelligent [69]. To ensure that no values are dropped during inference the Dropout layer only applies when training is set to True in call (). Training will be automatically set to true when model fit is used. When invoking the layer in other situations you can explicitly set the argument to True.

### 3.15.2. Early Stopping

Deep learning models have frequently employed early stop as a regularization strategy to avoid overfitting.  discovered that early stopping worked well to avoid overfitting when compared to other regularization strategies [70].

### 3.15.3. L1 and L2 Regularization

In our research we used regularization techniques to prone overfitting and underfitting in training and testing model Because RNNs are flexible and regularization is essential for optimal performance. This research employed L2 regularization also known as weight decay plays a crucial role in enhancing the performance of end-to-end speech recognition model [40].

# CHAPTER FOUR

# 4. RESEARCH DESIGN

## 4.1. Introduction

In this chapter, the methods used to develop deep learning-based end-to-end speech recognition model for the guragunga language in case of cheha dialect is described. Specifically, it covers the techniques employed to collect and assess the data required for the system, which includes both text and speech corpus. Moreover, it delves into the proposed speech recognition model for cheha, that uses deep learning, and details how the training process was conducted, along with the various testing and evaluation methods used to determine its effectiveness.

## 4.2. Proposed Model Architecture

The entity framework outlines a comprehensive pipeline for developing a speech recognition model, beginning with dataset preparation, which includes collecting data from sources like the Old and New Gurugram Testaments, followed by the Chaha Text Data. Preprocessing steps involve removing extraneous spaces and punctuation, splitting text into sentences, and recording audio from the text, ensuring high-quality input for the model. Noise reduction and normalization enhance audio quality, while feature extraction techniques like MFCC and spectrograms convert audio signals into usable formats. The data is then split into training and test sets, allowing for effective model training using architectures such as LSTMs, GRUs, and Bigru. Finally, the model is evaluated using metrics like Word Error Rate (WER), This structured approach ensures robustness and reliability in handling the complexities of speech recognition tasks. The general entity framework of the cheha speech recognition model that I proposed from starting from data collection, preprocessing, segmenting, labeling and also train and testing phase are illustrated in the below diagram.

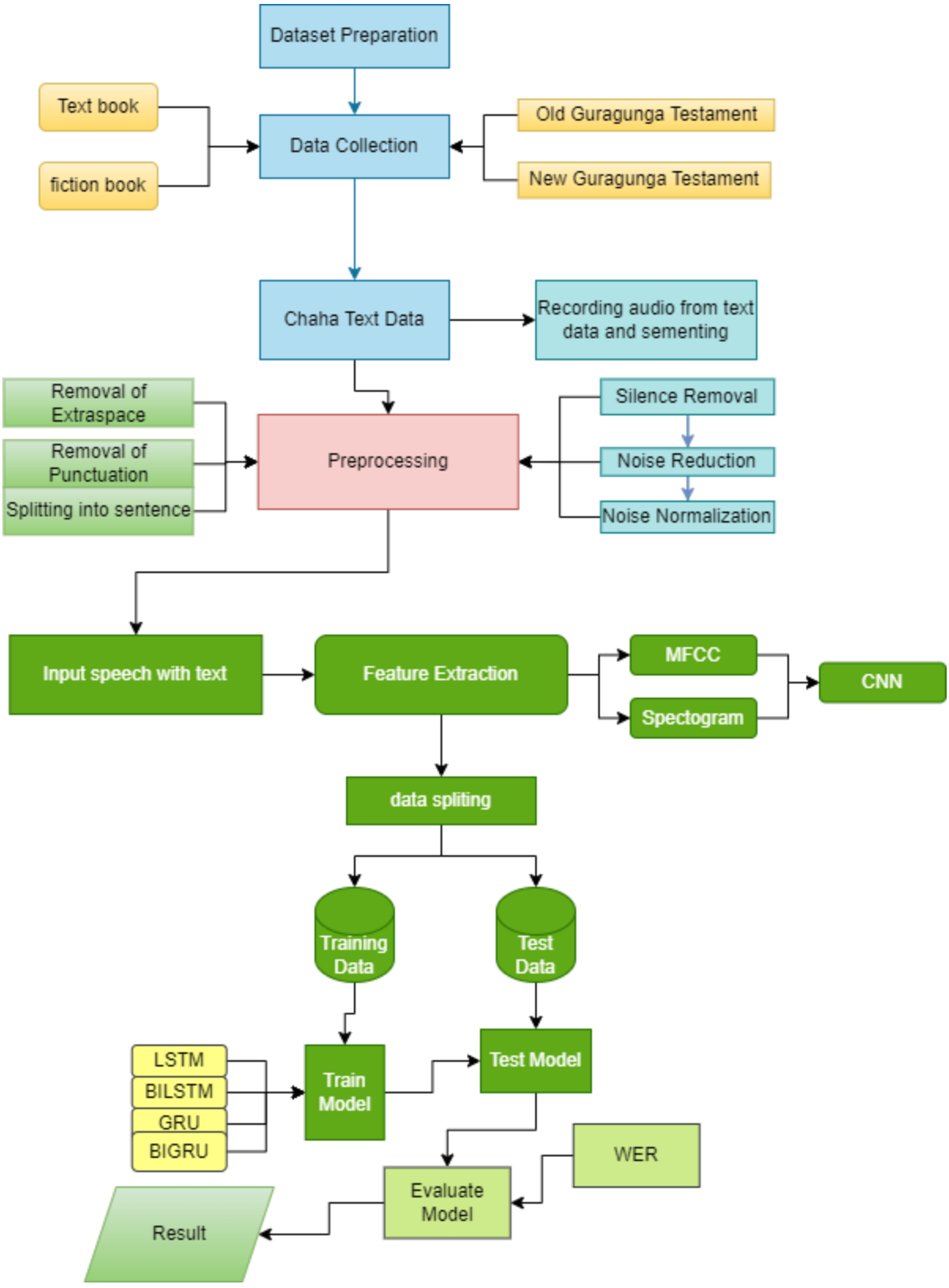



Figure 4.1 Proposed Architecture of Cheha Speech Recognition

## 4.3. Data Collection and Processing

Speech recognition is a growing field in artificial intelligence and machine learning, aiming to develop algorithms and models that can understand and interpret human speech. The development of speech recognition systems for under resource languages such as cheha, poses unique challenges. This thesis describes the process of building a quality deep learning model for cheha speech recognition, focusing on the importance of good quality data, the sources of data used, and the steps taken to ensure that the data is representative of the target population. By understanding the importance of data quality and the steps taken to collect and process the data, we can ensure that the final model is accurate, efficient, and robust.

Data is collected, processed and then fed to the deep learning algorithm to build the desired model. So, to build our model we understood that we should prepare good quality data for a quality model. The data that we have fed for the deep learning algorithm is a speech corpus of having a speech and its transcriptions. We have recorded speeches of different people in consideration of sex and age proportion. We have built speech corpus of recording from studio-based area.

## 4.4. Data Splitting

The training and testing data were should put separately using two folders namely training and testing for both speech and transcript data. Among the developed corpora, 15,600 sentences with their audio file for training, the rest 3400 sentences including their audio data were used for testing purpose and those testing data selected randomly using ten speakers, The dataset is divided as 80% train and 20%. test set.

## 4.5. Feature extraction and Engineering

One kind of characteristic parameter that can be seen and accurately captures every aspect of human speech is a spectrogram. The majority of speech recognition systems use MFCC parameters although their stability and noise resistance are limited. As a result, this research makes use of spectrogram-extracted features which provide increased stability and resilience and improve speech recognition accuracy [71]. High-level spatial information is extracted from the image using CNN. CNN can be used to capture high level features in the spatial domain since the spectrogram plot can be viewed as a transformed intensity of frequencies over time that mimics images. CNN can obtain more robust features by utilizing local filtering and maximum pooling approaches in

contrast to traditional speech features. One particular kind of CNN that works with 2D input data, such an image, is called a 2D Convolutional Neural Network (2D-CNN). The convolutional layers of a 2D-CNN carry out 2D convolutions by scanning the input data and extracting local features using 2D kernels. 2D-CNNs are well-suited for tasks like object recognition and segmentation because of the hierarchical representations created by the many convolutional and pooling layers, which enable them to learn progressively complicated and discriminative features Furthermore, because 2D-CNNs enable the automatic extraction of pertinent characteristics from an audio signals spectrogram, they can be an effective tool for speech recognition tasks. End-to-end speech recognition systems that can handle variable-length inputs and represent the intricate interactions between speech sounds can be produced by merging 2D-CNNs with other deep learning architectures such as RNNs [65].

# CHAPTER FIVE

# 5. EXPERIMENTATION RESULTS AND DISCUSSION

## 5.1. Introduction

The designed architecture is discussed in the chapter before chapter. Thus, this section provides detail on corpus source, experimental implementation, detail Explanation how dataset preprocessed and how it used, Variety each conducted model experiment, parameter selection of the thesis was done It also discusses how the preprocessing effect training was suggested based on the selected model.

In this chapter, we present our work on End- to- end speech recognition, including process for collecting and preprocessing a large dataset, parameter selection and training inputs for our model, and evaluating its performance on various metrics.

## 5.2. Dataset Pre-Processing

The analysis of audio length (in seconds) and frequency in the dataset reveals important insights into the characteristics of the audio samples. Each audio file's duration is measured in seconds, while its frequency content, analyzed through techniques like the Fourier Transform or spectrogram, indicates tonal qualities and pitch variations. By summarizing audio lengths alongside their average frequencies, we can observe trends, such as shorter clips typically having a limited frequency range compared to longer ones that may encompass a broader spectrum. visualizations like scatter plots and histograms can further illustrate these relationships, helping to inform feature extraction and model training for applications in speech recognition and music classification.

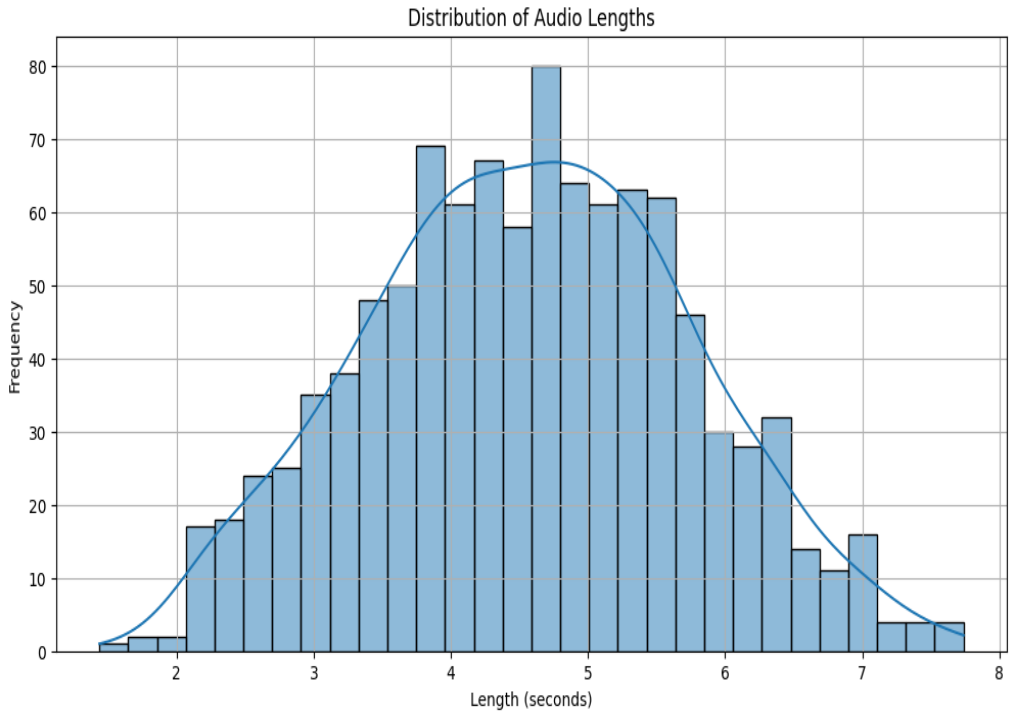



Figure 5.1 frequency distribution of Audio lengths

## 5.3. Implementation Environment

For corpus preparation, we drafted the research report on a Dell laptop. Our workflow primarily took place in Google Colab, starting with dataset preparation in .txt and .wav formats. We uploaded the dataset to Google Drive for easy access and mounted it in the Colab notebook for proper retrieval. We selected Python 3.8 as our primary programming language for model development due to its extensive library support and ease of use. Essential libraries included Keras for deep learning, TensorFlow 2.12 as the backend, Librosa for audio processing, and NumPy for numerical tasks. Google Colab was chosen for its access to GPUs with 32GB of RAM, significantly speeding up training compared to CPU usage. We trained RNN models on GPUs, processing multiple utterances in parallel to enhance training speed through matrix-matrix multiplication. Within each group, we padded utterances to match the longest one, excluding padding frames from gradient computation. To further optimize efficiency, we sorted training utterances by length, minimizing padding and ensuring similar lengths within groups.

## 5.4. Hyperparameter Selection

In our research, we explored various hyperparameters across different model types, including LSTM, BILSTM, GRU, BIGRU with Attention models. The batch sizes chosen for the models varied between 32 and 64, allowing us to assess their impact on training efficiency and performance, while each model utilized a hidden dimension of 256 for consistent comparison. We experimented with learning rates of 0.01 and 0.001 for LSTM, BILSTM, and GRU models, with the BIGRU and CTC/Attention models including an additional option of 0.1. All models employed the Adam optimizer, known for its effectiveness in training deep learning models.

Table 5.1. Hyper-parameters

| **No** | **Hyperparameter** | **Model TYPE** | | | |
|---|---|---|---|---|---|
| | | **LSTM** | **BILSTM** | **GRU** | **BIGRU** |
| 1 | Batch size | 64, 32 | 64, 32 | 32, 64 | 32, 64 |
| 2 | Hidden units | 1024 | 1024 | 1024 | 1024 |
| 3 | Learning rate | 0.01,0.001 | 0.01,0.01 | 0.001,0.01 | 0.01,0.01 |
| 4 | Optimization | Adam optimizer | Adam optimizer | Adam optimizer | Adam optimizer |
| 5 | Epoch | 50,100 | 50,100 | 50,100 | 50,100 |
| 6 | Drop rate | 0.4,0.5 | 0.4,0.5 | 0.4,0.5 | 0.4,0.5 |

## 5.5. Model Implementation

This research describes the distribution of an end-to-end speech recognition model. The solution uses CNN-RNN layers and dense layers to process the input spectrograms and produce character or word probabilities based on a TensorFlow/Keras model.

### 5.5.1. Spectrogram Input and Preprocessing

A variable-length series of spectrograms with input dim frequency bins for each spectrogram is the models input. The inputs none form indicates that the sequence length is flexible. In order to give the input spectrograms, the additional dimension needed for the convolutional layers they are reshaped. The symbol for this additional dimension is 1 which is a singleton dimension. The input layer receives a 3D tensor as input where the batch size is the first dimension the time steps (which are set to None to allow sequences of arbitrary length), and the input feature dimension (input dim) are the second and third dimensions respectively. The input layer's name is set to input. The input tensor is subsequently reshaped to include an extra dimension of size 1 which is necessary for the 2D convolutional layer. To automatically determine the amount of time steps, the second dimension is set to -1. A spectrogram with shape (None, input dim), where None denotes the variable-length time dimension, serves as the model's input. First, an extra channel dimension is added to the input, giving it the shape (-1, input dim, 1). Applying 2D convolutional layers to the input spectrogram requires this step.

```python
def encode_single_sample(wav_file, label):
    """Process a single audio file and its corresponding label."""
    file = tf.io.read_file(wav_file)
    audio, _ = tf.audio.decode_wav(file)
    audio = tf.squeeze(audio, axis=-1)
    audio = tf.cast(audio, tf.float32)
    # Get the spectrogram
    spectrogram = tf.signal.stft(audio, frame_length=frame_length,
    frame_step=frame_step, fft_length=fft_length)
    spectrogram = tf.abs(spectrogram)
    spectrogram = tf.math.pow(spectrogram, 0.5)
    # Normalize the spectrogram
    means = tf.math.reduce_mean(spectrogram, axis=1, keepdims=True)
    stddevs = tf.math.reduce_std(spectrogram, axis=1, keepdims=True)
    spectrogram = (spectrogram - means) / (stddevs + 1e-10)
    # Add a channel dimension to the spectrogram
    spectrogram = tf.expand_dims(spectrogram, axis=-1)
    # Process the label
    label = tf.strings.unicode_split(label, input_encoding="UTF-8")
    label = char_to_num(label)
    # Resize spectrogram to fixed height and width
    spectrogram = tf.image.resize(spectrogram, [desired_height, desired_width])
    # Permute dimensions to match model input shape
    spectrogram = tf.transpose(spectrogram, perm=[1, 0, 2])
    # Truncate or pad labels to fixed length
    label_length = tf.shape(label)[0]
    if label_length > desired_label_length:
        label = label[:desired_label_length] else:
        padding_amount = desired_label_length - label_length
        label = tf.pad(label, [[0, padding_amount]], constant_values=0)
    tf.debugging.assert_equal(tf.shape(label)[0], desired_label_length,
    "Label padding mismatch.")
    return spectrogram, label
```

Figure 5.2 Sample code to show Preprocessing

### 5.5.2. Convolutional Layers

The customized CNN architecture for speech recognition consists of two convolutional layers that extract features from the input audio spectrogram, which represents audio signals in a 2D format with time on the horizontal axis and frequency on the vertical axis. The first convolutional layer uses a kernel size of 11x41 and a stride of 2x2 to produce 32 feature maps, capturing local patterns. The second layer, applying a kernel size of 11x21 and a stride of 1x2, extracts additional features from the first layer's output, also generating 32 feature maps. This design enables the model to automatically learn relevant features without manual engineering. After feature extraction, the output is reshaped for input into recurrent layers, which model temporal dependencies in the data. This combination of convolutional and recurrent layers allows the model to effectively analyze speech signals and generate accurate transcriptions.

```
# Convolutional layers
    x = layers.Conv2D(filters=32, kernel_size=[11, 41], strides=[2, 2],
                      padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)

    x = layers.Conv2D(filters=32, kernel_size=[11, 21], strides=[1, 2],
                      padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
```

Figure 5.3 Sample code for CNN layer

To enhance neural network performance, batch normalization and ReLU activation functions are commonly used after convolutional layers. Batch normalization normalizes each batch's output, reducing internal covariate shift and stabilizing training. ReLU activation introduces non-linearity, allowing the network to capture more complex features from the input spectrogram.

### 5.5.3. RNN Layers

The model incorporates various RNN layers, including LSTM, Bidirectional LSTM (BILSTM), GRU, and Bidirectional GRU (BIGRU). For each layer, a recurrent neural network is created with a specified number of units, and dropout regularization is applied after each layer at a rate of 0.5 to prevent overfitting. The input data is reshaped to fit the RNN requirements, and following the RNN layers, a dense layer is added with twice the number of units, using the ReLU activation function for non-linearity. Finally, the output layer generates probabilities for each class using the SoftMax activation function, with the number of units set to the output dimension plus one. This structure allows the model to effectively learn complex patterns in sequential data while maintaining generalization capabilities.

```
# Reshape for RNN

x = layers.Reshape((-1, x.shape[-2] * x.shape[-1]))(x)

# RNN layers with Bidirectional GRU

for i in range(1, rnn_layers + 1):

recurrent = layers.GRU(units=rnn_units, return_sequences=True, name=f"gru_{i}")

x = layers.Bidirectional(recurrent)(x)

if i < rnn_layers:

x = layers.Dropout(rate=0.5)(x)
```

Figure 5.4 Sample code for RNN layer

### 5.5.4. Dens and Output Layer

The Dense layer is a fully connected layer that transforms outputs from previous layers into final predictions corresponding to the character vocabulary. It has units set to rnn_units * 2, allowing for a larger representation space, which aids in learning complex relationships. A ReLU (Rectified Linear Unit) activation function is applied, introducing non-linearity and enabling the model to capture intricate patterns. Following this, a Dropout layer with a rate of 0.4 helps prevent overfitting by randomly setting half of the inputs to zero during training. Finally, the output layer maps to the character vocabulary size plus one (for the CTC blank token) with Attention and uses a SoftMax activation function to convert scores into probabilities, indicating the likelihood of each character or word being predicted.

```
# Dense layer

x = layers.Dense(units=rnn_units * 2)(x)

x = layers.ReLU()(x)

# Output layer
output = layers.Dense(units=output_dim + 1, activation="softmax")(x)
```

Figure 5.5. Sample code for dens and output layer

## 5.6. RESULT

In this study, the process of speech recognition begins with extracting spectrogram features from raw audio. These spectrogram features are then fed into a Convolutional Neural Network (CNN) to extract high-level features, which are essential for building the acoustic model.

To optimize the model's performance, several experiments were conducted by varying hyperparameters such as batch size, learning rate, and the number of epochs. The CNN used a filter size of 41 and a kernel size of 11, with 1024 hidden units to enhance feature extraction.

We utilized Google Colab, a cloud-based platform that enables users to write and execute Python code directly in their browsers, for all training, testing, and evaluation tasks. This environment is particularly useful for data analysis and machine learning.

The focus of our research was on end-to-end speech recognition using a Connectionist Temporal Classification (CTC) and attention mechanisms in conjunction with various Recurrent Neural Network (RNN) algorithms, including LSTM, Bilstm, GRU, and BIGRU. Each model's performance was evaluated based on different hyperparameter configurations to assess their effectiveness in sequence modeling tasks. This comprehensive approach allowed us to identify the most effective configurations for speech recognition in the Cheha dialect.

Table 5.2. Experimental result of each RNN model

| Model Type | Hidden unit | Optimizer | Epoch | Drop rate | Learning rate | Batch size | Accuracy | loss |
|---|---|---|---|---|---|---|---|---|
| **LSTM** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 90.99 | 0.81 |
| | | | | | | 64 | 92.84 | 0.75 |
| | | | | | 0.001 | 32 | 89.35 | 0.19 |
| | | | | | | 64 | 95.29 | 0.36 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.856 |
| | | | | | | 64 | 94.02 | 0.57 |
| | | | | | 0.001 | 32 | 95.45 | 0.65 |
| | | | | | | **64** | **95.99** | **0.415** |
| **BILSTM** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 92.87 | 0.88 |
| | | | | | | 64 | 93.54 | 0.26 |
| | | | | | 0.001 | 32 | 95.35 | 0.49 |
| | | | | | | 64 | 95.29 | 0.376 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.95 |
| | | | | | | 64 | 94.52 | 0.62 |
| | | | | | 0.001 | 32 | 95.45 | 0.365 |
| | | | | | | **64** | **96.92** | **0.328** |
| **GRU** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 95.99 | 0.823 |
| | | | | | | 64 | 94.84 | 0.75 |
| | | | | | 0.001 | **32** | **96.25** | **0.375** |
| | | | | | | 64 | 95.29 | 0.36 |
| | | | | 0.5 | 0.01 | 32 | 93.19 | 0.856 |
| | | | | | | 64 | 94.02 | 0.57 |
| | | | | | 0.001 | 32 | 96.05 | 0.65 |
| | | | | | | 64 | 95.22 | 0.27 |
| **BIGRU** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 91539 | 0.83 |
| | | | | | | 64 | 91.62 | 0327 |
| | | | | | 0.001 | 32 | 92.39 | 0.49 |
| | | | | | | 64 | 95.29 | 0.523 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.726 |
| | | | | | | 64 | 94.12 | 0.617 |
| | | | | | 0.001 | 32 | 96.35 | 0.525 |
| | | | | | | **64** | **97.50** | **0.251** |

### 5.6.1. Experiment in LSTM Model

In our first experiment, we selected Long Short-Term Memory (LSTM) as our model type, utilizing 1024 hidden units and the Adam optimizer. The training process was divided into two sets of epochs: 50 and 100. For the 50 epochs, we set the drop rate to 0.4 and experimented with learning rates of 0.01 and 0.001, using batch sizes of 32 and 64 for each learning rate. Implementing a CNN-LSTM model with a CTC/Attention architecture was essential for our analysis. This model incorporated LSTM cells on the encoder side and utilized label encoders, ensuring a comprehensive approach. Consistent parameters were applied across all deep learning models. The results for the LSTM model showed that various configurations were tested to optimize performance. Across the experiments, a lower learning rate of 0.001 consistently led to better accuracy, particularly when combined with a larger batch size of 64. The drop rate also influenced performance, with a value of 0.5 generally yielding more favorable results. Ultimately, the highest accuracy achieved was 95.99, indicating that this combination effectively balanced learning and overfitting prevention. Overall, the findings suggest that careful tuning of the learning rate, drop rate, and batch size can significantly enhance model performance, with the optimal configuration demonstrating strong accuracy and reasonable training loss.

Table 5.3. Lstm experimental result with different hyper-parameter

| Model Type | Hidden unit | Optimizer | Epoch | Drop rate | Learning rate | Batch size | Accuracy | loss |
|---|---|---|---|---|---|---|---|---|
| **LSTM** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 90.99 | 0.81 |
| | | | | | | 64 | 92.84 | 0.75 |
| | | | | | 0.001 | 32 | 89.35 | 0.19 |
| | | | | | | 64 | 95.29 | 0.36 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.856 |
| | | | | | | 64 | 94.02 | 0.57 |
| | | | | | 0.001 | 32 | 95.45 | 0.65 |
| | | | | | | **64** | **95.99** | **0.415** |

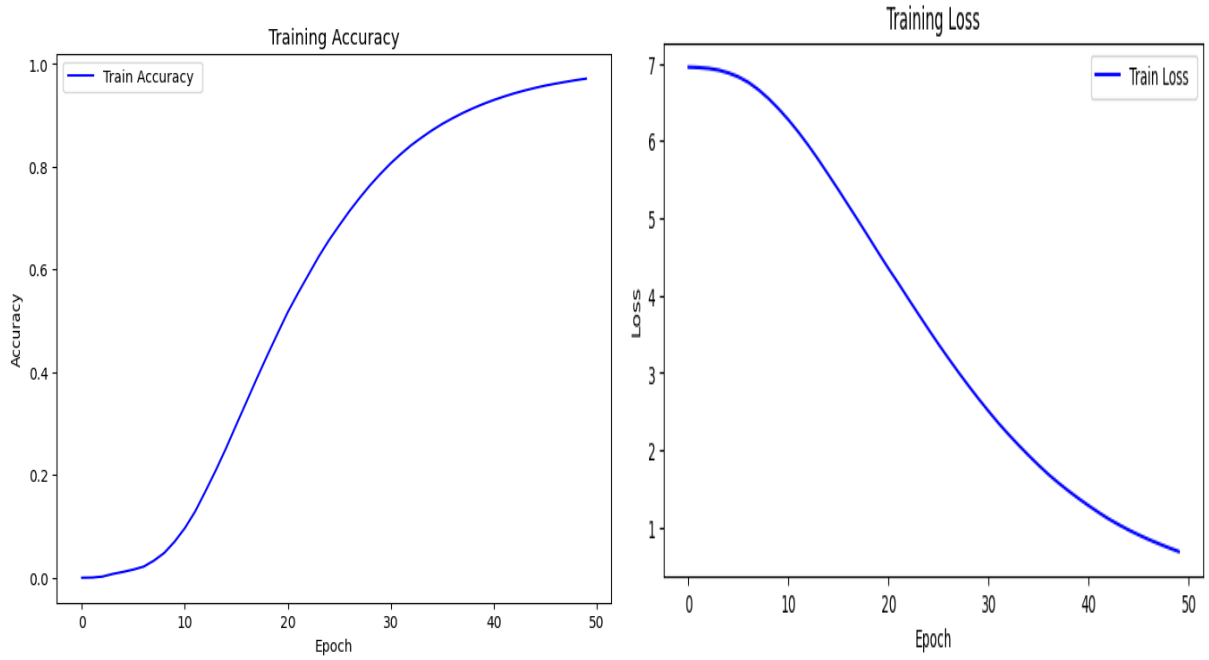



Figure 5.6. CNN- LSTM model Accuracy and Loss Graph

### 5.6.2. Experiment in BILSTM Model

In our second experiment, we implemented a Bidirectional Long Short-Term Memory (Bilstm) model, which leverages two LSTM cells on the encoder side, allowing for enhanced context understanding but requiring more memory than standard LSTM models. Configured with 1024 hidden units and the Adam optimizer, the model was trained over 50 epochs, maintaining consistent parameters from our previous LSTM experiment. This model train in a remarkable accuracy of 96.92%, showcasing the superior performance of the Bilstm architecture in capturing complex patterns in the data. This high accuracy underscores the effectiveness of our parameter tuning and the potential of Bilstm models for tasks requiring sequential data analysis.

Table 5.4. Bilstm experimental result with different hyper-parameter

| Model Type | Hidden unit | Optimizer | Epoch | Drop rate | Learning rate | Batch size | Accuracy | loss |
|---|---|---|---|---|---|---|---|---|
| **BILSTM** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 92.87 | 0.88 |
| | | | | | | 64 | 93.54 | 0.26 |
| | | | | | 0.001 | 32 | 95.35 | 0.49 |
| | | | | | | 64 | 95.29 | 0.376 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.95 |
| | | | | | | 64 | 94.52 | 0.62 |
| | | | | | 0.001 | 32 | 95.45 | 0.365 |
| | | | | | | **64** | **96.92** | **0.328** |

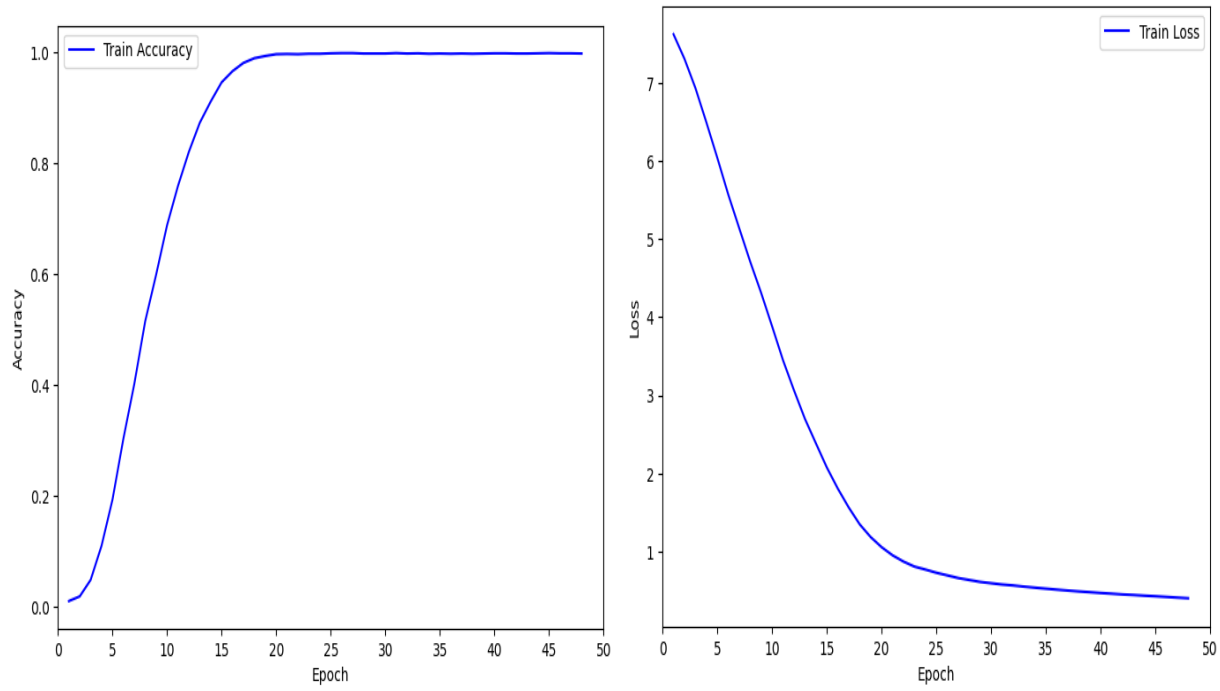



Figure 5.7. CNN-BILSTM model Accuracy and Loss Graph

### 5.6.3. Experiment in GRU Model

In this phase, we implemented the Gated Recurrent Unit (GRU) model, configured with 1024 hidden units and the Adam optimizer, training for 50 epochs. The model's performance was influenced by various hyperparameters, including the drop rate, learning rate, and batch size. We tested drop rates of 0.4 and 0.5, alongside learning rates of 0.01 and 0.001. Our analysis revealed that the highest accuracy achieved was 96.25% with a batch size of 32 at a learning rate of 0.001, highlighting the model's effective learning capacity and the significance of careful hyperparameter tuning.

Table 5.5. Gru experimental result with different hyper-parameter

| Model Type | Hidden unit | Optimizer | Epoch | Drop rate | Learning rate | Batch size | Accuracy | loss |
|---|---|---|---|---|---|---|---|---|
| **GRU** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 95.99 | 0.823 |
| | | | | | | 64 | 94.84 | 0.75 |
| | | | | | 0.001 | **32** | **96.25** | **0.375** |
| | | | | | | 64 | 95.29 | 0.36 |
| | | | | 0.5 | 0.01 | 32 | 93.19 | 0.856 |
| | | | | | | 64 | 94.02 | 0.57 |
| | | | | | 0.001 | 32 | 96.05 | 0.65 |
| | | | | | | 64 | 95.22 | 0.27 |

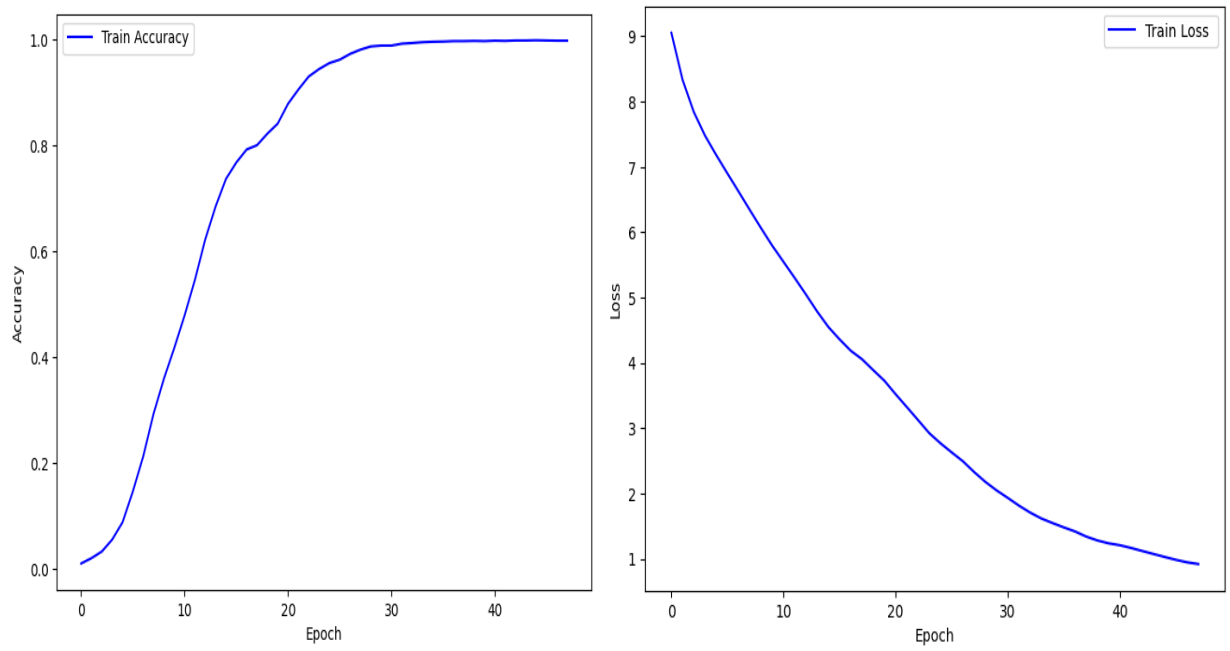



Figure 5.8. CNN-GRU model Accuracy and Loss Graph

### 5.6.4. Experiment in BIGRU Model

In our fourth experiment, we implemented the CNN-Bigru model, which features double GRU cells on the encoder side, utilizing significantly more memory than a standard GRU. We retained the same parameters used in our previous GRU experiment to ensure consistency. This approach yielded remarkable results, achieving an accuracy of 97.50%. Specifically, this accuracy was attained with the following parameters: 1024 hidden units, Adam optimizer, 50 epochs, a drop rate of 0.5, a learning rate of 0.001, and a batch size of 64. These findings highlight the effectiveness of the CNN-Bigru architecture in capturing complex patterns within the data.

Table 5.6. Bigru experimental result with different hyper-parameter

| Model Type | Hidden unit | Optimizer | Epoch | Drop rate | Learning rate | Batch size | Accuracy | loss |
|---|---|---|---|---|---|---|---|---|
| **BIGRU** | 1024 | Adam | 50 | 0.4 | 0.01 | 32 | 91539 | 0.83 |
| | | | | | | 64 | 91.62 | 0327 |
| | | | | | 0.001 | 32 | 92.39 | 0.49 |
| | | | | | | 64 | 95.29 | 0.523 |
| | | | | 0.5 | 0.01 | 32 | 90.19 | 0.726 |
| | | | | | | 64 | 94.12 | 0.617 |
| | | | | | 0.001 | 32 | 96.35 | 0.525 |
| | | | | | | **64** | **97.50** | **0.251** |

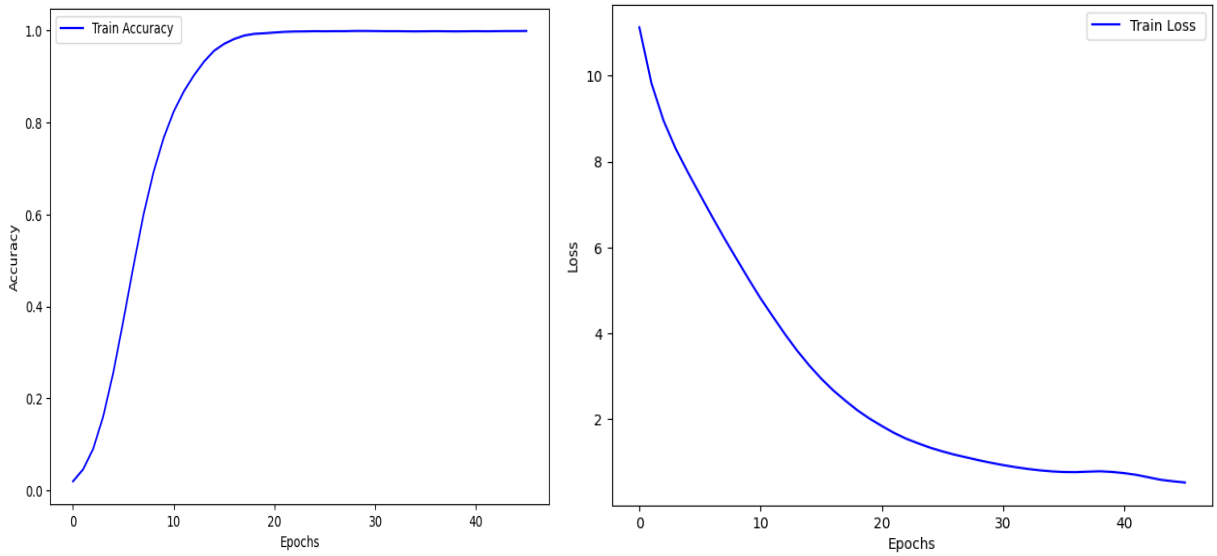



Figure 5.9. CNN-BIGRU model Accuracy and Loss Graph

**Word Error Rate (WER)** is a key metric for evaluating the performance of RNN model in speech recognition measuring the accuracy of transcribed text by calculating the ratio of errors insertions, deletions, and substitutions compared to the total number of words in the actual transcription so that in our low resource language cheha dialect the WER rate in graph Bigru model is better for cheha speech recognition purpose for transcribing speech in to text.

A WER of 0.05 is generally considered to be a good result in the field of speech recognition. It suggests that the system is capable of accurately transcribing spoken words with a high degree of accuracy, even in the presence of background noise or other audio disturbances. As speech recognition technology continues to evolve, it is likely that WERs will continue to decrease, leading to even more accurate transcription of spoken language.

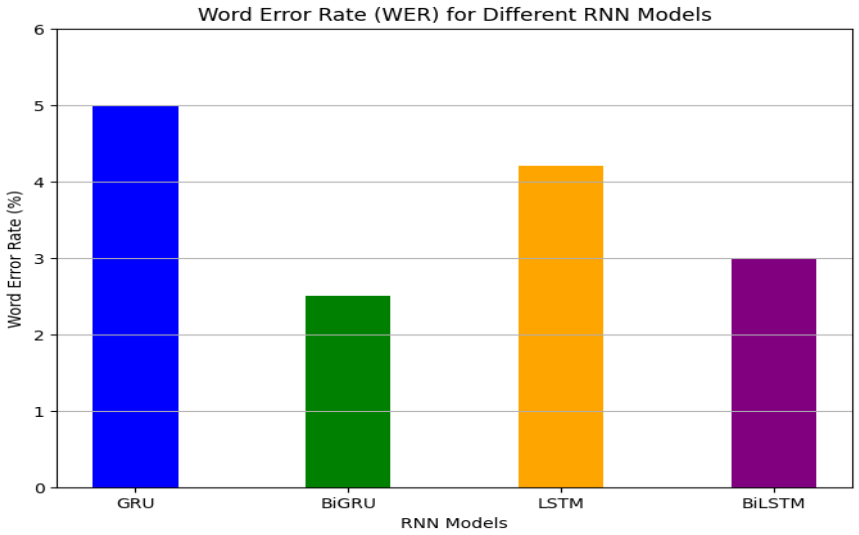



Figure 5.10. Word error rate comparison

## 5.7. Discussion of the Result

In our experiments, we evaluated four types of recurrent neural network models: LSTM, Bilstm, GRU, and Bigru, each configured with 1024 hidden units and the Adam optimizer over 50 epochs. The models were tested across various hyperparameters, including drop rates, learning rates, and batch sizes. The LSTM model achieved a maximum accuracy of 95.99% with a batch size of 32 at a learning rate of 0.001 and a drop rate of 0.5. The Bilstm model demonstrated strong performance, reaching an accuracy of 96.92% with a batch size of 64 at a learning rate of 0.001 and a drop rate of 0.5, indicating its effectiveness in capturing complex data patterns. The GRU model also performed well, achieving its highest accuracy of 96.25% with a batch size of 32 at a learning rate of 0.001 and a drop rate of 0.4. Finally, the Bigru model outperformed the others, achieving an impressive accuracy of 97.50% with a batch size of 64 at a learning rate of 0.001 and a drop rate of 0.5. These results underscore the varying strengths of each model architecture, with the Bigru showing the best overall performance in this set of experiments. Our experiments highlight the outstanding performance of our end-to-end speech recognition model for the Guragunga language. The model performs exceptionally well in clean environments achieving low error rates but demonstrates reduced effectiveness in noisy conditions. In speech recognition modeling, the x-axis of a plot typically represents the number of training epochs, which indicates how many times the model sees the entire training dataset. The y-axis usually shows the loss a metric that evaluates the model's ability to predict the correct output for a specific input. The loss function quantifies the discrepancy between the predicted output and the actual output in speech recognition tasks. Accurate transcription of spoken words is crucial for the effectiveness of speech recognition technology. One important metric for assessing accuracy is the Word Error Rate (WER) which measures the frequency of errors in the transcription of spoken words or character. A lower WER signifies higher accuracy. In our analysis we used WER to evaluate the accuracy of our transcriptions against the ground truth. Our result indicate that our model has achieved an impressive transcription accuracy for Cheha speech with a WER of 2.5%. This marks a significant improvement over previous speech recognition systems for Guragunga which typically exhibited much higher error rates. The high accuracy of our model is particularly significant as it effectively tackles one of the main challenges in developing reliable speech recognition systems.

## 5.8. Summary

Recurrent Neural Networks (RNNs) are designed to process sequential data by maintaining a hidden state that captures information from previous time steps, making them suitable for tasks like time series prediction and natural language processing. Long Short-Term Memory (LSTM) networks address the vanishing gradient problem, incorporating memory cells and gates that allow them to learn long-term dependencies effectively. Bidirectional LSTMs (Bilstm) enhance this capability by processing sequences in both forward and backward directions, improving context understanding. Gated Recurrent Units (GRUs) simplify LSTMs by combining gates, resulting in fewer parameters and faster training while still capturing important temporal patterns. Bidirectional GRUs (Bigru) offer similar advantages as Bilstm but retain the efficiency of GRUs. The CNN-Bigru architecture combines Convolutional Neural Networks with Bigru, enabling enhanced feature extraction from spatial and temporal data, making it particularly effective for speech recognition and achieving a remarkable Word Error Rate (WER) of 2.5%. Additionally, Connectionist Temporal Classification (CTC) provides a training framework for RNNs in sequence-to-sequence tasks, allowing the model to handle unaligned input-output pairs, while attention mechanisms help models focus on relevant parts of the input, further improving performance in tasks like machine translation and text summarization. Together, these models represent significant advancements in the ability to handle complex sequential data across various applications.

## 5.9. Answering Research Questions

**RQ1. Which pre-processing tools and algorithms best in cheha speech and text dataset preparation for model training, testing and evaluations?**

For the preparation dataset some tools are recommended to use. In audio pre-processing, noise reduction techniques using tools like Audacity and Librosa such as noise reduce can help eliminate background noise. Normalization of volume levels across recordings ensures consistency while segmentation Audacity can break long audio files into manageable chunks for easier handling during training. For text pre-processing methods such as tokenization using libraries like NLTK or spaCy can split text into sentences or words.

**RQ2. Which feature extraction techniques best to extract relevant combination of features from cheha dialects for model building?**

When it comes to feature extraction from Cheha dialects several techniques be obvious particularly the use of spectrograms and CNN for spatial feature selection purpose. In speech recognition audio signals are transformed into spectrograms using techniques like Short-Time Fourier Transform which captures frequency information over time. These spectrograms enable models to visualize how frequencies change essential for understanding speech patterns. CNN are then used to extract features from these spectrograms. CNNs apply convolutional layers with filters that detect phonemes and other speech characteristics followed by pooling layers that down sample the feature maps to enhance efficiency and robustness. The high-level features produced by the CNN are passed to fully connected layers for classification into specific phonemes words or character.

**RQ3. Which deep learning Models best performer to build cheha speech recognition model?**

In this research we investigate deep learning models for speech recognition highlighting the effectiveness of BIGRU are used in various sequence-based tasks including speech recognition natural language processing and time series prediction due to their ability to effectively model temporal dependencies without the vanishing gradient problem common in standard RNNs which enhances contextual understanding particularly for the guragunga language. While GRUs provide simplicity and efficiency they can struggle with managing long-range dependencies. Therefore, we select the CNN-BIGRU architecture for an effective Cheha speech recognition model. This architecture combines CNNs for extracting spatial features from spectrograms with BIGRU for handling sequential data. Additionally, we incorporate Attention layers to focus on relevant parts of the input with CTC loss function for aligning variable-length sequences. This CNN-Bigru model significantly enhances performance in speech recognition tasks allowing for better contextual understanding and enhanced accuracy model in transcribing Cheha speech into text.

.

.

# CHAPTER SIX

# 6. CONCLUSTION AND RECOMENDATION

## 6.1. Overview

This chapter finalizes the whole work and gives a general conclusion about the study contribution and findings of the work and suggests some feature works for the upcoming researchers.

## 6.2. Conclusion

In this study we developed an end-to-end speech recognition model for the low-resource Guragunga language by combining CNN and RNN architectures with CTC/Attention mechanisms. For the Cheha dialect we achieved a word error rate of 2.5%. However, due to Cheha being a purely spoken language without a rigid grammatical structure developing a language model presented challenges. Overall, the research successfully achieved its objective of creating an efficient speech recognition system for the Guragunga language especially the Cheha dialect thus advancing the field and improving communication technologies for diverse linguistic communities. We chose BIGRU as the ideal deep learning algorithm for creating an end-to-end voice recognition model for Cheha. This choice was made in light of the finding that Bilstm necessitated a greater investment of computational resources and needed around twice as much processing time than BIGRU.

## 6.3. Contributions

The primary goal of this study, which was to use deep learning to create an end-to-end speech recognition model for the Guragigna language in the case of Cheha, was accomplished. The model design to use Attention with CTC loss function and mergers a convolutional neural network (CNN) and a recurrent neural network (RNN) it leads best achieved and important advances in a number of fields. This study presents an end-to-end speech recognition model for the Cheha language, addressing a gap in research using deep learning techniques. The model combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) utilizing CTC/Attention mechanisms. Key contributions include enhanced model performance efficiency and applicability to low-resource languages achieving automatic phoneme alignment without external tools. We no longer require alignment tools because we are using the end-to-end approaches for autonomous phoneme alignment rather than traditional ASR. To the best of our knowledge no end-to-end deep

learning method has yet been used to address the problem of Cheha speech recognition this work aims to close that gap.

## 6.4. Recommendations

Several suggestions might be made to further enhance and investigate end-to-end speech recognition in the Cheha dialects in light of the research findings. First combining the predictions of several models through the investigation of joint techniques can greatly improve the overall accuracy and robustness of end-to-end speech recognition model for guragigna language in case of cheha dialect. Improved performance across a range of speech inputs may result from utilizing strategies like model averaging and advancing which capitalize on the distinct advantages of various model architectures. Second, there can be significant advantages to tailoring the speech recognition models to certain Cheha dialect domains including legal or medical settings. The accuracy and relevance of the transcriptions can be greatly increased by using domain specific datasets to train the models. This focused strategy guarantees that the models are more capable of managing the distinct lexicon and speech patterns seen in many domains. Thirdly, exploring cross lingual speech recognition by training the models on multilingual data can facilitate better communication and understanding across different languages. This approach can enable applications such as translation services and assist in integrating guragigna language speakers into broader linguistic contexts thereby enhancing accessibility and usability. It is also critical to increase the quantity and variety of the guragigna language dataset that is utilized for assessment and training. The models generalization abilities can be enhanced with a larger and more diverse dataset which will enable them to function better across a greater variety of speech patterns accents and subjects. The deep grammatical subtleties of the Cheha dialectal can be better conveyed with this expansion. Finally, by focusing on these suggestions, researchers and developers can improve the performance and applicability of end-to-end speech recognition systems in the Cheha dialectal framework. Using the developed speech recognition models in real-world applications such as voice-activated assistants transcription services and educational tools can provide practical use cases that not only validate the model's effectiveness but also show their usefulness in everyday scenarios thereby contributing to the advancement of speech recognition technology in the guragigna language.

## 6.5. Future work

Future research can build on the current findings and investigate the following areas when assessing end-to-end speech recognition for the Guragigna language in the case of Cheha.

- ❖ the goal of our research is to develop end-to-end speech recognition for Guragigna language in case of cheha only for speech recognition, we suggest that additional studies conduct speech synthesis.
- ❖ Adding a variety of accents and dialects to the training dataset will enhance the system's inclusivity and effectiveness. This diversity allows the model to better comprehend a wide range of speech patterns, improving its ability to recognize and transcribe speech accurately across different speakers.
- ❖ Explore integrating visual sign such as lip movements in video alongside speech inputs to enhance recognition accuracy and robustness particularly in noisy environments.

# REFERENCE


[1] Ambuj Mehrish, NAVONIL Majumder, and Rishabh Bhardwaj, Rada Mihalcea, Soujanya Poria, "A Review of Deep Learning Techniques for Speech Processing," May 30, 2023. doi: 10.48550/arXiv.2305.00359.

[2] Richard P. Lippmann, "Review of Neural Networks for Speech Recognition," pp. 1–38, 1989.

[3] JANE ORUH, SERESTINA VIRIRI, (Senior Member, IEEE), AND ADEKANMI ADEGUN, "Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition," *IEEE Access*, vol. 10, pp. 30069–30079, 2022, doi: 10.1109/ACCESS.2022.3159339.

[4] Praphulla A. Sawakare, Ratndeep R. Deshmukh, Pukhraj P. Shrishrimal, "Speech Recognition Techniques: A Review," vol. 6, no. 8, Art. no. 8, 2015.

[5] Ashok Kumar, Vikas Mittal, "Speech Recognition: A Complete Perspective," vol. 7, no. 6, Art. no. 6, Apr. 2019, doi: F90340476C19.

[6] Zhang Leini, Sun Xiaolei, "Study on Speech Recognition Method of Artificial Intelligence Deep Learning," 2021. doi: doi:10.1088/1742-6596/1754/1/012183.

[7] V. A. Kherdekar and S. A. Naik, "Speech Recognition System Approaches, Techniques and Tools for Mathematical Expressions A Review," vol. 8, no. 08, Art. no. 08, 2019.

[8] Chenfeng Miao, Kun Zou, Ziyang Zhuang, Tao Wei, Jun Ma, Shaojun Wang, Jing Xiao, "Towards Efficiently Learning Monotonic Alignments for Attention-Based End-to-End Speech Recognition," pp. 1051–1055, 2022.

[9] Da-Hee Yang, Joon-Hyuk Chang, "Attention-based latent features for jointly trained end-to-end automatic speech recognition with modified speech enhancement," pp. 202–210, 2023, doi: https://doi.org/10.1016/j.jksuci.2023.02.007.

[10] Andrew Rosenberg, Kartik Audhkhasi, Abhinav Sethy, Bhuvana Ramabhadran, Michael Picheny, "END-TO-END SPEECH RECOGNITION AND KEYWORD SEARCH ON LOW-RESOURCE LANGUAGES." [Online]. Available: https://github.com/rizar/attention-lvcsr

[11] T. G. Fantaye, J. Yu, and T. T. Hailu, "Syllable-based Speech Recognition for a Very Low-Resource Language, Chaha," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, Sanya China: ACM, Dec. 2019, pp. 415–420. doi: 10.1145/3377713.3377794.

[12] T. G. Fantaye, J. Yu, and T. T. Hailu, “Investigation of Automatic Speech Recognition Systems via the Multilingual Deep Neural Network Modeling Methods for a Very Low-Resource Language, Chaha,” *J. Signal Inf. Process.*, vol. 11, no. 01, Art. no. 01, 2020, doi: 10.4236/jsip.2020.111001.

[13] Fitsum Gizachew Deriba, “DEVELOPING PART OF SPEECH TAGGER FOR GURAGIGNA LANGUAGE,” BAHIR DAR UNIVERSITY BAHIR DAR INSTITUTE OF TECHNOLOGY, 2023. [Online]. Available: http://hdl.handle.net/123456789/10761

[14] Jan Chorowski,Dzmitry Bahdanau, Yoshua Bengio, “End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results.” Dec. 2014.

[15] Rohit Prabhavalkar , Member, IEEE, Takaaki Hori , Senior Member, IEEE, Tara N. Sainath , Fellow, IEEE, and Ralf Schlüter , Senior Member, IEEE, and Shinji Watanabe , Fellow, IEEE, “End-to-End_Speech_Recognition_A_Survey.pdf,” vol. VOL. 32, 2024.

[16] Takaaki Hori, Jaejin Cho, Shinji Watanabe, “END-TO-END SPEECH RECOGNITION WITH WORD-BASED RNN LANGUAGE MODELS,” *arXiv:1808.02608v1 [cs.CL] 8 Aug 2018*, Aug. 2018.

[17] Yohannes Ayana Ejigu and Tesfa Tegegne Asfaw, “Large Scale Speech Recognition for Low Resource Language Amharic, An End-to-End Approach,” 2024. doi: 10.20944/preprints202402. 0813.v1.

[18] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, “JOINT CTC-ATTENTION BASED END-TO-END SPEECH RECOGNITION USING MULTI-TASK LEARNING,” pp. 4835–4839, 2017.

[19] J. Levis and R. Suvorov, “Automatic Speech Recognition,” in *The Encyclopedia of Applied Linguistics*, 1st ed., C. A. Chapelle, Ed., Wiley, 2012. doi: 10.1002/9781405198431.wbeal0066.

[20] K. S and C. E, “A Review on Automatic Speech Recognition Architecture and Approaches,” *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 9, no. 4, Art. no. 4, Apr. 2016, doi: 10.14257/ijsip.2016.9.4.34.

[21] T. G. Tohye, “Towards Improving the Performance of Spontaneous Amharic Speech Recognition,” Oct. 2015.

[22] T. A. Altaye, “Designing automatic speech recognition for Ge’ez language,” Mar. 2020, [Online]. Available: http://hdl.handle.net/123456789/10545

[23] Mamyrbayev Orken, Oralbekova Dina, Alimhan Keylan, Turdalykyzy Tolganay & Othman Mohamed, “A study of transformer-based end-to-end speech recognition system for Kazakh language,” 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-12260-y

[24] Zillay Huma, Areej Mustafa, “Graph-Based and End-to-End Model Integration for Low-Resource.pdf,” vol. Vol 4, 2024, [Online]. Available: https://universe-publisher.com/index.php/jcit

[25] Song Wang and Guanyu Li, “Overview of end-to-end speech recognition,” in *Journal of Physics: Conference Series*, 2019. doi: doi:10.1088/1742-6596/1187/5/052068.

[26] P. Wang, “Research and Design of Smart Home Speech Recognition System Based on Deep Learning,” in *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, Chongqing, China: IEEE, Jul. 2020, pp. 218–221. doi: 10.1109/CVIDL51233.2020.00-98.

[27] Yerlan Karabaliyev, Kateryna Kolesnikova, Nurkhan Batyrkhan, “Review of methods of end-to-end automatic recognition of Kazakh speech,” 2024, pp. 615–620. [Online]. Available: www.sciencedirect.com

[28] Sendong Liang, Wei Qi Yan, “Multilingual Speech Recognition Based on The End-to-End Framework”.

[29] Vimala.C, “A_Review_on_Speech_Recognition_Challenge.pdf,” vol. 2, pp. 1–7, 2012.

[30] Yadeta Gonfa Gutu and M. Yifiru, “A Large Vocabulary, Speaker-Independent, Continuous Speech Recognition System for Afaan Oromo: Using Broadcast News Speech,” 2022, doi: 10.13140/RG.2.2.26635.98089.

[31] “Kherdekar and Naik - 2019 - Speech Recognition System Approaches, Techniques A.pdf.”

[32] A. D. Fantahun, “Syllable based speaker independent continuous speech recognition for Afan oromo”, [Online]. Available: http://hdl.handle.net/123456789/10343

[33] S.Karpagavalli, R.Deepika, P.Kokila, K.Usha Rani, “Automatic Speech Recognition: Architecture, Methodologies and Challenges - A Review,” *Int. J. Adv. Res. Comput. Sci.*, vol. 2, No. 6, Dec. 2011.

[34] T. Zeng, “Deep Learning in Automatic Speech Recognition (ASR): A Review,” in *Proceedings of the 2022 7th International Conference on Modern Management and Education Technology (MMET 2022)*, C. F. Peng, H. S. Du, T. S. Yin, J. Prabhu, and H. Li, Eds., Paris: Atlantis Press SARL, 2023, pp. 173–179. doi: 10.2991/978-2-494069-51-0_23.

[35] A. Loubser, “END-TO-END AUTOMATED SPEECH RECOGNITION USING A CHARACTER BASED SMALL SCALE TRANSFORMER ARCHITECTURE”.

[36] Hamza Kheddara, Mustapha Hemisb and Yassine Himeurc, “Automatic Speech Recognition using Advanced Deep Learning Approaches: A survey,” Apr. 2024.

[37] Johan Hagner, “Recurrent Neural Networks for End-to-End Speech Recognition,” p. 37, Jan. 2018.

[38] E. Morris, “Automatic Speech Recognition for Low-Resource and Morphologically Complex Languages”.

[39] Sin-Horng Chen, Yuan-Fu Liao, Song-Mao Chiang, and Saga Chang, “An RNN-based preclassification method for fast continuous Mandarin speech recognition,” *IEEE Trans. Speech Audio Process.*, vol. 6, no. 1, Art. no. 1, Jan. 1998, doi: 10.1109/89.650315.

[40] Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton, “Speech Recognition with Deep Recurrent Neural Networks”.

[41] H. K. Hamarashid, “Modified Long Short-Term memory and Utilizing in Building sequential model,” vol. Vol.9, May 2021, doi: DOI: 10.14741/ijmcr/v.9.3.2.

[42] Van Huy Nguyen, “An End-to-End model for Vietnamese speech recognition”.

[43] Farhad Mortezapour Shiria, Thinagaran Perumala, Norwati Mustaphaa, and Raihani Mohameda, “A Comprehensive Overview and Comparative Analysis on Deep Learning Models”.

[44] Gaofeng Cheng, Daniel Povey, Lu Huang, Ji Xu, Sanjeev Khudanpur, Yonghong Yan, “Output-Gate Projected Gated Recurrent Unit for Speech Recognition”.

[45] Sakib Ashraf Zargar, “Introduction to Sequence Learning Models: RNN, LSTM, GRU,” 2021. doi: DOI: 10.13140/RG.2.2.36370.99522.

[46] Manpreet Singh Bhatia, Alok Aggarwal, Narendra Kumar, “Speech-to-text conversion using GRU and one hot vector encodings.” Jan. 2021.

[47] Bharat Bhatta, Basanta Joshi, Ram Krishna, “Nepali Speech Recognition Using CNN, GRU and CTC.” Sep. 24, 2020.

[48] Burak TOMBALOGLU, Hamit ERDEM, “Turkish Speech Recognition Techniques and Applications of Recurrent Units (LSTM and GRU),” pp. 1035–1049, 2021, doi: DOI: 10.35378/gujs.816499.

[49] Markus Nussbaum-Thom, Jia Cui, Bhuvana Ramabhadran, Vaibhava Goel, “Acoustic Modeling using Bidirectional Gated Recurrent Convolutional Units,” pp. 8–12, 2016, doi: http://dx.doi.org/10.21437/Interspeech.2016-212.

[50] “End-to-end indonesian speech recognition with convolutional and gated recurrent units.”

[51] G. Degemu, “Developing Semantic Textual Similarity for Guragigna Language Using Deep Learning Approach,” *Int. J. Adv. Sci. Comput. Appl.*, vol. 5, no. 1, Art. no. 1, Nov. 2024, doi: 10.47679/ijasca. v4i2.106.

[52] “Chaha_language.” [Online]. Available: https://en.wikipedia.org/wiki/Chaha_language

[53] “Hashim Changrampadi et al. - 2022 - End-to-End Speech Recognition of Tamil Language.pdf.”

[54] A. Gulati *et al.*, “Conformer: Convolution-augmented Transformer for Speech Recognition,” May 16, 2020, *arXiv*: arXiv:2005.08100. doi: 10.48550/arXiv.2005.08100.

[55] Mohamed Hashim Changrampadi, A. Shahina, M. Badri Narayanan and A. Nayeemulla Khan, “End-to-End Speech Recognition of Tamil Language,” vol. vol.32, 2022, doi: DOI:10.32604/iasc.2022.022021.

[56] A. Mukhamadiyev, I. Khujayarov, O. Djuraev, and J. Cho, “Automatic Speech Recognition Method Based on Deep Learning Approaches for Uzbek Language,” *Sensors*, vol. 22, no. 10, Art. no. 10, May 2022, doi: 10.3390/s22103683.

[57] H. A. Alsayadi, A. A. Abdelhamid, I. Hegazy, and Z. T. Fayed, “Arabic speech recognition using end-to-end deep learning,” *IET Signal Process.*, vol. 15, no. 8, Art. no. 8, Oct. 2021, doi: 10.1049/sil2.12057.

[58] N. K. Manaswi, *Deep Learning with Applications Using Python*. Berkeley, CA: Apress, 2018. doi: 10.1007/978-1-4842-3516-4.

[59] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, Oriol Nieto, “librosa: Audio and Music Signal Analysis in Python,” presented at the Python in Science Conference, Austin, Texas, 2015, pp. 18–24. doi: 10.25080/Majora-7b98e3ed-003.

[60] M. Lamba and M. Madhusudhan, “Text Pre-Processing,” in *Text Mining for Information Professionals*, Cham: Springer International Publishing, 2022, pp. 79–103. doi: 10.1007/978-3-030-85085-2_3.

[61] Shubham singh, “How to Get Started with NLP – 6 Unique Methods to Perform Tokenization.” Dec. 08, 2024. [Online]. Available:

https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/

[62] GIZACHEW BELAYNEH GEBRE, "Artificial Neural Network Based Amharic Language Speaker Recognition," *Turk. J. Comput. Math. Educ. TURCOMAT*, vol. 12, no. 3, Art. no. 3, Apr. 2021, doi: 10.17762/turcomat. v12i3.2043.

[63] A Free, Open Source, Cross-Platform Audio Editor, "audacity.pdf." [Online]. Available: http://audacity.sourceforge.net/onlinehelp-1.2/reference.html

[64] Ali Nasr-Esfahani, Mehdi Bekrani, Roozbeh Rajabi, "Robust Recognition of Persian Isolated Digits in Speech using Deep Neural Network," Dec. 2024.

[65] Ahmad Al Harere, Khloud Al Jallad, "Quran Recitation Recognition using End-to-End Deep Learning."

[66] M. Khairi Ishak, D. Ovind Madsen, and F. Ahmed Al-Zahrani, "An Optimal Method for Speech Recognition Based on Neural Network," *Intell. Autom. Soft Comput.*, vol. 36, no. 2, Art. no. 2, 2023, doi: 10.32604/iasc.2023.033971.

[67] Jinyu Li, "Recent Advances in End-to-End Automatic Speech Recognition.pdf." 2022. [Online]. Available: (http:// creativecommons.org/ licenses/ by-nc/ 4.0/)

[68] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-Based End-to-End Speech Recognition on Voice Search," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB: IEEE, Apr. 2018, pp. 4764–4768. doi: 10.1109/ICASSP.2018.8462492.

[69] Suyanto, Anditya Arifianto, Anis Sirwan, Angga P. Rizaendra, "End-to-End Speech Recognition Models for a Low-Resourced Indonesian Language," in *2020 8th International Conference on Information and Communication Technology (ICoICT)*, Yogyakarta, Indonesia: IEEE, Jun. 2020, pp. 1–6. doi: 10.1109/ICoICT49345.2020.9166346.

[70] B. M. Hussein and S. M. Shareef, "An Empirical Study on the Correlation between Early Stopping Patience and Epochs in Deep Learning," *ITM Web Conf.*, vol. 64, p. 01003, 2024, doi: 10.1051/itmconf/20246401003.

[71] Jia-nan Chen, Shuang Gao, Han-zhe Sun, Xiao-hui Liu, Zi-ning Wang, Yan Zheng, "An End-to-end Speech Recognition Algorithm based on Attention Mechanism," in *2020 39th Chinese Control Conference (CCC)*, Shenyang, China: IEEE, Jul. 2020, pp. 2935–2940. doi: 10.23919/CCC50068.2020.9189026.

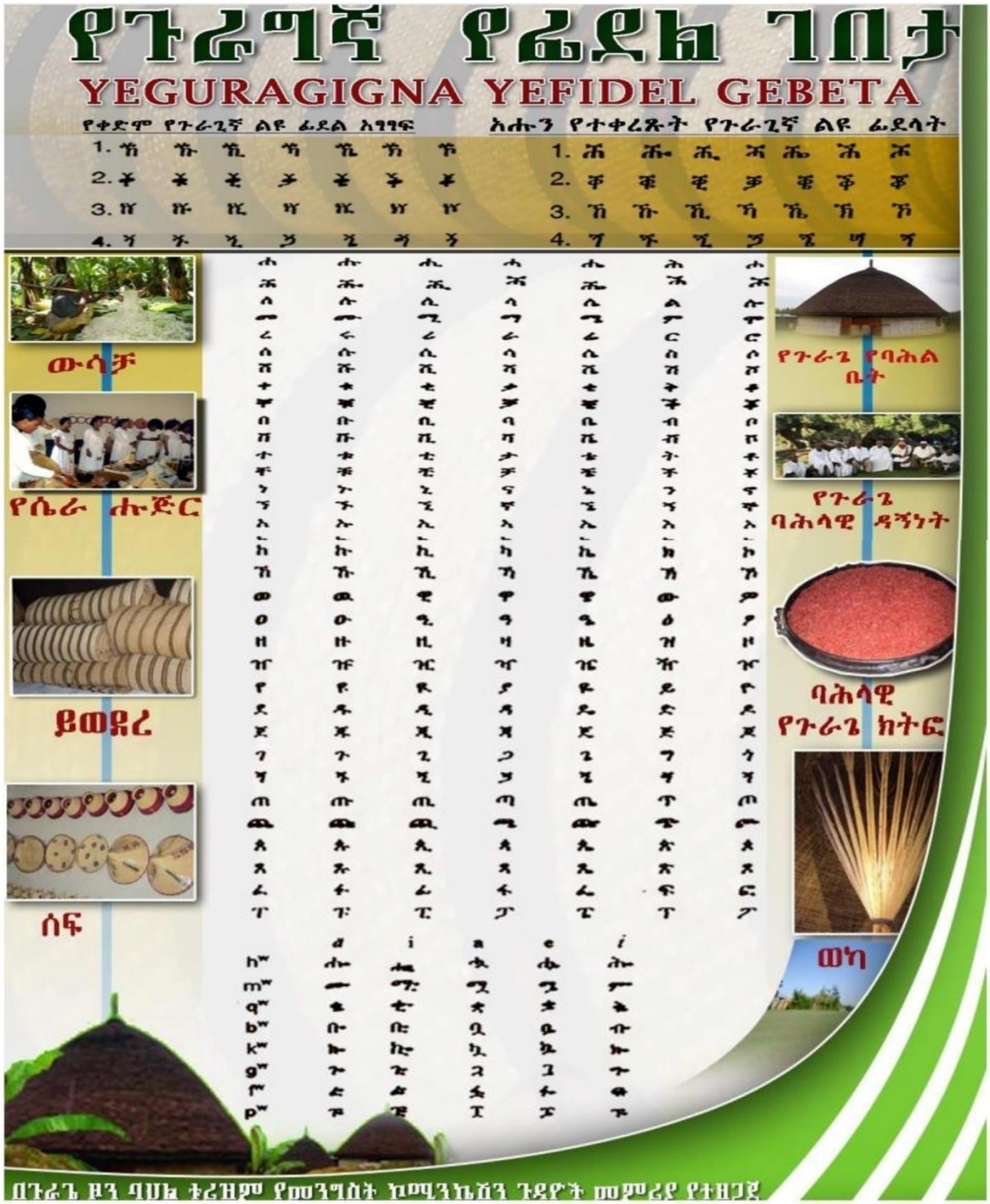
# APPENDIX

## Appendix A

Appendix Table 1. Sample of text Corpus

| | |
|---|---|
| NesF_00001.wav | "የፍጥረት መጣፍ አለም የትፈጠሮ ኽማ የሰብ ዘር የቸነኽማ" |
| NesF_00002.wav | "የባጢርም የጅጓረም ንብረት የቄነሺኽማም እግዘር ተሰብ ዘር ጋመ ያነን መራኽብ ዩድ" |
| NesF_00003.wav | "የፍጥረት መጣፍ በሑት ንቅ ንቅ መደር ይሸጅዬ ይኽር" |
| NesF_00004.wav | "ተመንሽ አት ቄነሽም አስራት ስን አለም የትፈጠሮ ኽማም የሰብ ዘር ይፍት ወረር ታሪክ ዩድ" |
| NesF_00005.wav | "የአዳምም የኼዋን የቃየልም የአቤልም የኖኽም የጥፋት እኻ" |
| NesF_00006.wav | "የባቢሎን ግምብም ያነ በዝ ደኑ" |
| NesF_00007.wav | "ተመንሽ አስሑት ቄነሽም አምሳ ስን ዝኽታ የእስራኤል የድረ አባሹና ታሪክ ዩድ" |
| NesF_00008.wav | "ተኽኖም ግብት ይፍትወረር በመሮትመታ" |
| NesF_00009.wav | "ይእግዘርም በታዘዞተታ የትሸረ አብርኻም ባነ" |
| NesF_00010.wav | "ተኽም አንቄ የርቸታ ይስኻቅ የመቸበያመታ" |
| HawF_00011.wav | "የያእቆብም አስረሔት የያእቆብ ደንጓም ታሪክ ዩድ" |
| HawF_00012.wav | "ተዝም አስረሔት የያእቆብ ደንጓም ግብት" |
| HawF_00013.wav | "በዮሴፍ የኽክም ያእቆብም የተነፎም ደንጓታ ታበሩስመሔና" |
| HawF_00014.wav | "ግብፅ ገነ በቸነቦቹ ዘንጋ ንቃር ፍታታም ዩድ ሚኒም አኽር" |
| HawF_00015.wav | "ዝኽ መጣፍ የሰብ ታሪክ ዩድ ቃር ቢኽርም ትንም በፅር ዩድ ያነ" |
| HawF_00016.wav | "እግዘር የቾተን ሜናው እግዘር አለም በፈጠረወ ታሪ ዘንጎት ይቀርስምታነ" |
| HawF_00017.wav | "እግዘር የሰበታ እስቦት ኤችኽማ ያትየሽ ተስፋ ቦዶት ይጀፕር" |
| HawF_00018.wav | "በመጣፈታ ደን የቄነሽቹ ቲጀቹርቹ ስን ባነ በታሪክ ኤነቃር" |
| HawF_00019.wav | "ቢቾቶ ሰብ ፍርድም ቅጣትም ያሰራ ሰብመታ ይመራም ያግዝ" |
| HawF_00020.wav | "የታሪክመሔና ዘንጋ ይሸካክት እግዘር ገገታው" |

## Appendix B




Appendix Figure 1. Guragigna Fidel gebeta

# Appendix C

| | A | B | C |
|---|---|---|---|
| 1 | file_name | spectrogram_mean | mel_spectrogram_mean |
| 2 | NarF_00613.wav | [0.04493793845176697, 0.04716997966170311, 0.05819473788 | [0.0017138600815087557, 0.006127681117504835, 0.018050264567136765, 0.016980813816189766, 0.05313039571046829, 1.326073646545 |
| 3 | NarF_00614.wav | [0.04362718388438225, 0.05266258493065834, 0.06634532660 | [0.00245320214889943 6, 0.0069107236340641975, 0.017765840515494347, 0.01958228088915348, 0.06266395002603531, 3.469192504882812 |
| 4 | NarF_00615.wav | [0.04600328207015991, 0.05458929017186165, 0.07546646147 | [0.0032209919299930334, 0.015098252333700657, 0.044665001333275986, 0.04195816442370415, 0.06959647685289383, 1.9800596237182617 |
| 5 | NarF_00616.wav | [0.04138633608818054, 0.0514095239341259, 0.071064941585 | [0.002676500240340829, 0.0092890849333698177, 0.022323628887533414, 0.019296890124678612, 0.04388796538114548, 2.222532033920288, |
| 6 | NarF_00618.wav | [0.04422646015882492, 0.04386393335334301, 0.053780410438 | [0.002171882661059499, 0.004283609800040722, 0.006137018091976643, 0.009249940514564514, 0.02117220312356949, 0.4895990788936613 |
| 7 | NarF_00619.wav | [0.04841528832912445, 0.05856611207127571, 0.08330163359 | [0.003822537139058113, 0.011099717579782009, 0.023626461625099182, 0.015286778099834919, 0.03907328471541405, 1.167595863342285 |
| 8 | NarF_00617.wav | [0.03367104008793831, 0.04630972072482109, 0.05888016521 | [0.0029841698706150055, 0.008371937088668346, 0.023504348471760 75, 0.03651648759841919, 0.03237705677774 77264, 1.7339674234390257 |
| 9 | NarF_00620.wav | [0.04319200664758682, 0.04934106767177582, 0.06396492570 | [0.0025838746223598 72, 0.008521975018084 05, 0.020698573440313 34, 0.017891095951199 53, 0.029430760070681572, 1.447400450706482, 2 |
| 10 | NarF_00621.wav | [0.04784737154841423, 0.05623394995927811, 0.07676048576 | [0.004726708400994539, 0.012493850663304329, 0.037271738052368164, 0.025324935093522072, 0.10074496269226074, 3.625276327133178 |
| 11 | NarF_00622.wav | [0.04390756785869 5984, 0.05563218891620636, 0.0748260985 | [0.003460181411355734, 0.009798732586205006, 0.025378346443176 27, 0.024029497057199478, 0.05651903897523 88, 2.36015248298645, 12 |
| 12 | NarF_00623.wav | [0.04118264093995094, 0.04956220835447 3114, 0.0683170706 | [0.0030347681604325 77, 0.008268409408628 94, 0.022474743426 58461, 0.018102023750543594, 0.03219628334045 41, 1.884689450263977, 13.3 |
| 13 | NarF_00624.wav | [0.04504125565290451, 0.05543386563658714, 0.07725827395 | [0.003461384912952 7807, 0.012253138236701488, 0.036264751106500626, 0.03048691712319851, 0.06349118053913116, 1.8033996820449 83 |
| 14 | NarF_00626.wav | [0.04621629044413 5666, 0.05636724084615 7074, 0.064222842 | [0.003309267805889249, 0.010567773133516312, 0.024697788059711456, 0.02427484653890133, 0.06064468622207 6416, 0.882239639759063 |
| 15 | NarF_00625.wav | [0.03479912132024765, 0.04667405039072037, 0.06429309397 | [0.0024226149544119835, 0.008548086509108543, 0.017972648143768 31, 0.016760213300585747, 0.03455235436558 7234, 1.276000380516 05 |
| 16 | NarF_00627.wav | [0.04651889204978943, 0.05699811130762 1, 0.0800661146640 | [0.003301146440207 9582, 0.010043565183877945, 0.029756801202893257, 0.022395333275198936, 0.04305683076381 6833, 0.90986168384 55 |
| 17 | NarF_00628.wav | [0.03761031106114 3875, 0.04818165302276611, 0.0608577914 | [0.0020809234119951725, 0.007711171172559261, 0.024067834019660 95, 0.016842279583215714, 0.03714607283473015, 1.346090793609619 |
| 18 | NarF_00632.wav | [0.04378909617662 43, 0.05426185578107834, 0.073919072747 | [0.003736378159373 9986, 0.008904733695089817, 0.021655755117535 59, 0.018753858283162117, 0.06709601730108261, 2.416436672210693 |
| 19 | NarF_00629.wav | [0.04376563057303 4286, 0.05475788936018944, 0.0682711079 | [0.0022893105633556843, 0.007422498427331448, 0.024814328178763 39, 0.034121330827474594, 0.06066840887069702, 1.650936603546142 |
| 20 | NarF_00631.wav | [0.03958610445261 0016, 0.05136389657855034, 0.0760420784 | [0.003439253196120262, 0.009563707746565342, 0.023986067622900 01, 0.020467733964323997, 0.04893614351749 42, 1.831753134727478, 1 |
| 21 | NarF_00630.wav | [0.04715939611196518, 0.05632819980382919, 0.07139164205 | [0.004167569335550 07, 0.013611611910164356, 0.047004967927932 74, 0.038206882774829865, 0.06470233947038 65, 2.022927522659 3018, 1 |
| 22 | NarF_00635.wav | [0.03007290326058 8646, 0.04369660466909 4086, 0.067714914 | [0.003587074112147093, 0.014169863425195217, 0.028905453160405 16, 0.027608828619122505, 0.02683025971055 0308, 1.538285136222839 |
| 23 | NarF_00637.wav | [0.03733012825250 6256, 0.04564083740115166, 0.0649532154 | [0.0029500625096261 5, 0.007784092333167791, 0.018985604867339134, 0.011883041821420193, 0.03116525709629059, 1.772376775741577 1 |
| 24 | NarF_00634.wav | [0.04105965793132782, 0.05230164900422096, 0.07334198057 | [0.003687146585434675, 0.01144316140562296, 0.025622844696044922, 0.016815025359392166, 0.06700053811073303, 2.552418708801269 5 |
| 25 | NarF_00636.wav | [0.04462469741702 08, 0.05549229681491852, 0.070563070476 | [0.003525343956425786, 0.014203186146914959, 0.030544776469469 07, 0.023416467010974884, 0.05245300382375717, 2.519829750061035, |
| 26 | NarF_00633.wav | [0.04749775677919388, 0.05761115998029709, 0.07523944228 | [0.003813969902694 2253, 0.017208334058523178, 0.037354610860347 75, 0.026158181950449944, 0.06055987998843193, 2.169073581695556 |
| 27 | NarF_00638.wav | [0.04848947003483772, 0.05623853206634 5215, 0.0759067684 | [0.0029300642199814 32, 0.015264094807207584, 0.087905079126358 03, 0.048433445394039154, 0.05409266054630 2795, 5.133649349212646 |

Appendix Figure 2. sample feature from speech corpus

## Appendix D

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | (None, None, None, 129) | 0 |
| expand_dim (Reshape) | (None, None, 129, 1) | 0 |
| conv2d_18 (Conv2D) | (None, None, 65, 32) | 14,432 |
| batch_normalization_18 (BatchNormalization) | (None, None, 65, 32) | 128 |
| re_lu_27 (ReLU) | (None, None, 65, 32) | 0 |
| conv2d_19 (Conv2D) | (None, None, 33, 32) | 236,544 |
| batch_normalization_19 (BatchNormalization) | (None, None, 33, 32) | 128 |
| re_lu_28 (ReLU) | (None, None, 33, 32) | 0 |
| reshape_9 (Reshape) | (None, None, 1056) | 0 |
| bidirectional_45 (Bidirectional) | (None, None, 256) | 910,848 |
| dropout_45 (Dropout) | (None, None, 256) | 0 |
| bidirectional_46 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_46 (Dropout) | (None, None, 256) | 0 |
| bidirectional_47 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_47 (Dropout) | (None, None, 256) | 0 |
| bidirectional_48 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_48 (Dropout) | (None, None, 256) | 0 |
| bidirectional_49 (Bidirectional) | (None, None, 256) | 296,448 |
| dense_18 (Dense) | (None, None, 256) | 65,792 |
| re_lu_29 (ReLU) | (None, None, 256) | 0 |
| dropout_49 (Dropout) | (None, None, 256) | 0 |
| dense_19 (Dense) | (None, None, 236) | 60,652 |

Total params: 2,474,316 (9.44 MB)
Trainable params: 2,474,188 (9.44 MB)
Non-trainable params: 128 (512.00 B)

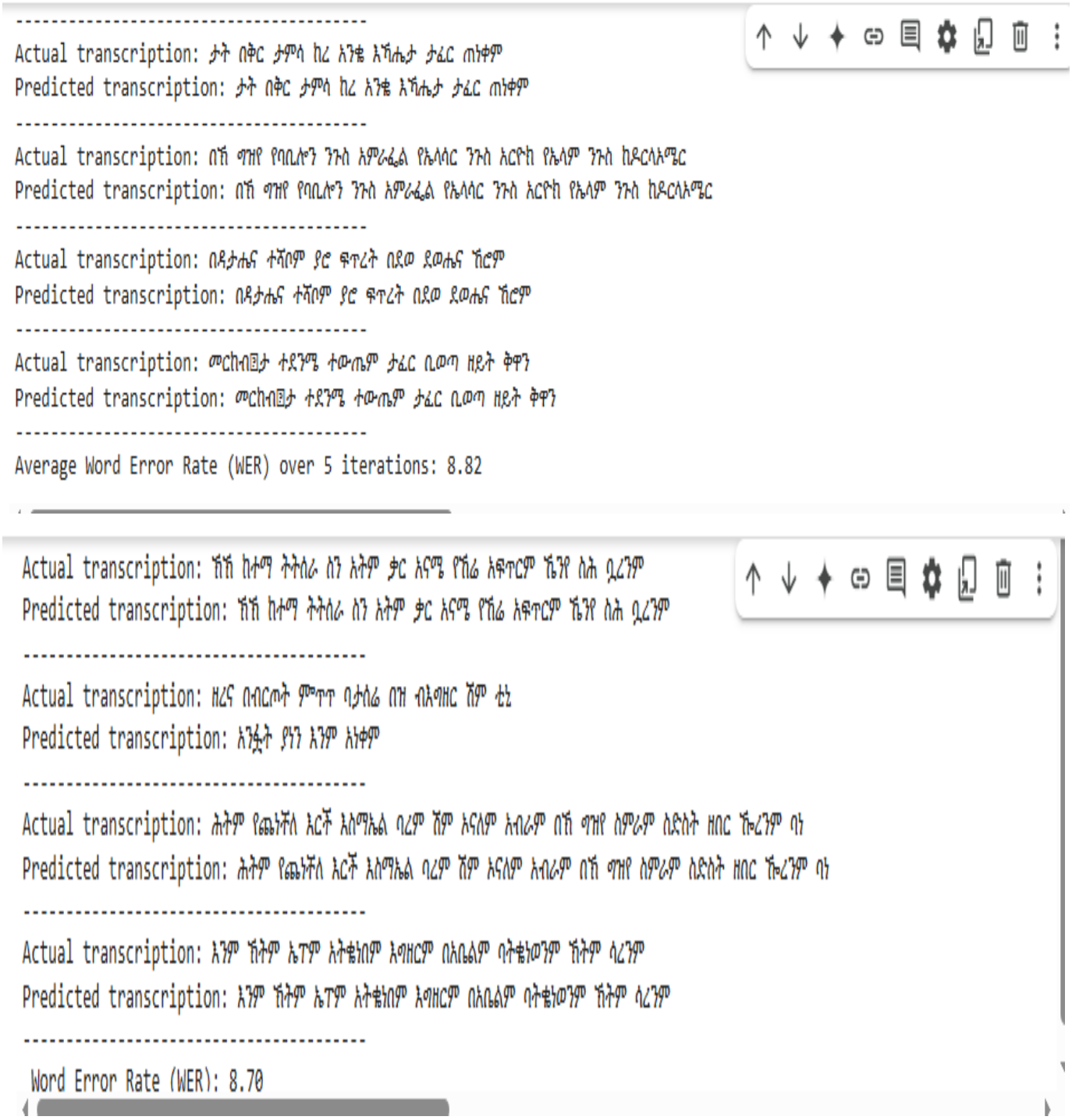
Model: "DeepSpeech_2"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | (None, None, 129, 129) | 0 |
| reshape_1 (Reshape) | (None, None, 129, 129) | 0 |
| conv2d_1 (Conv2D) | (None, None, 65, 32) | 1,861,760 |
| batch_normalization (BatchNormalization) | (None, None, 65, 32) | 128 |
| re_lu (ReLU) | (None, None, 65, 32) | 0 |
| conv2d_2 (Conv2D) | (None, None, 33, 32) | 236,576 |
| batch_normalization_1 (BatchNormalization) | (None, None, 33, 32) | 128 |
| re_lu_1 (ReLU) | (None, None, 33, 32) | 0 |
| reshape_2 (Reshape) | (None, None, 1056) | 0 |
| bidirectional (Bidirectional) | (None, None, 256) | 910,848 |
| dropout (Dropout) | (None, None, 256) | 0 |
| bidirectional_1 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_1 (Dropout) | (None, None, 256) | 0 |
| bidirectional_2 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_2 (Dropout) | (None, None, 256) | 0 |
| bidirectional_3 (Bidirectional) | (None, None, 256) | 296,448 |
| dropout_3 (Dropout) | (None, None, 256) | 0 |
| bidirectional_4 (Bidirectional) | (None, None, 256) | 296,448 |
| dense (Dense) | (None, None, 256) | 65,792 |
| re_lu_2 (ReLU) | (None, None, 256) | 0 |
| dense_1 (Dense) | (None, None, 236) | 60,652 |

Total params: 4,321,676 (16.49 MB)
Trainable params: 4,321,548 (16.49 MB)
Non-trainable params: 128 (512.00 B)

Appendix Figure 3. Number of parameters used for training

## Appendix E




Appendix Figure 4. sample output for proposed model

## Appendix F

```
import os
import numpy as np
import librosa
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Dropout,
     Input, Attention, Conv2D, MaxPooling2D, Reshape, Bidirectional
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# Load audio data and preprocess
def load_audio_files_and_labels(audio_dir, label_file):
    audio_data = []
    labels = []
    audio_files = [f for f in os.listdir(audio_dir) if f.endswith('.wav')]

    with open(label_file, 'r') as f:
        transcriptions = f.readlines()

    for audio_file, transcription in zip(audio_files, transcriptions):
        audio_path = os.path.join(audio_dir, audio_file)
        transcription = transcription.strip()

        if os.path.exists(audio_path):
            signal, sr = librosa.load(audio_path, sr=None)
            mfccs = librosa.feature.mfcc(y=signal, sr=sr, n_mfcc=13)
            audio_data.append(mfccs)
            labels.append(transcription)

    return audio_data, labels
```

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, backend as K
# Define the CNN-BiGRU with Attention model
def build_model(input_shape, num_classes):
    inputs = layers.Input(shape=input_shape)
    # CNN layers
    x = layers.Conv2D(32, kernel_size=(3, 3), activation='relu')(inputs)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.Conv2D(64, kernel_size=(3, 3), activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.Reshape(target_shape=(-1, x.shape[-1]))(x)  # Reshape for RNN
    # BiGRU layers
    x = layers.Bidirectional(layers.GRU(128, return_sequences=True))(x)
    x = layers.Bidirectional(layers.GRU(128, return_sequences=True))(x)
    # Attention mechanism
    attention = layers.Attention()([x, x])
    x = layers.Concatenate()([x, attention])
    # Output layer
    outputs = layers.Dense(num_classes, activation='softmax')(x)
    model = models.Model(inputs, outputs)
    return model
# CTC loss function
def ctc_loss(y_true, y_pred):
    y_true_len = tf.cast(tf.reduce_sum(y_true != -1, axis=1), tf.int32)
    y_pred_len = tf.fill(tf.shape(y_true_len), tf.shape(y_pred)[1])
    return K.ctc_batch_cost(y_true, y_pred, y_true_len, y_pred_len)
# Build and compile the model
model = build_model(input_shape, num_classes)
model.compile(optimizer='adam', loss=ctc_loss)
# Print model summary
model.summary()
```

```python
# Plot training loss and accuracy with different sizes
def plot_history(history):
    # Plot training loss
    plt.figure(figsize=(8, 4))  # Larger size for loss plot
    smoothed_loss = smooth_curve(history.history['loss'])
    plt.plot(smoothed_loss, label='Train Loss ', color='blue')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # Plot training accuracy
    plt.figure(figsize=(8, 6))  # Smaller size for accuracy plot
    smoothed_accuracy = smooth_curve(history.history['accuracy'])
    plt.plot(smoothed_accuracy, label='Train Accuracy ', color='blue')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

# Call the plot function
plot_history(history)
```

```
Epoch 44/50
26/26 [==============================] - 1s 41ms/step - loss: 0.1519 - accuracy: 0.9710
Epoch 45/50
26/26 [==============================] - 1s 45ms/step - loss: 0.0818 - accuracy: 0.9819
Epoch 46/50
26/26 [==============================] - 1s 49ms/step - loss: 0.0757 - accuracy: 0.9782
Epoch 47/50
26/26 [==============================] - 1s 50ms/step - loss: 0.0689 - accuracy: 0.9867
Epoch 48/50
26/26 [==============================] - 1s 50ms/step - loss: 0.0614 - accuracy: 0.9879
Epoch 49/50
26/26 [==============================] - 1s 43ms/step - loss: 0.0624 - accuracy: 0.9891
Epoch 50/50
26/26 [==============================] - 1s 36ms/step - loss: 0.0383 - accuracy: 0.9879
```

Appendix Figure 5. sample codes