2021-01

Multi-Script Ethiopic Document Images Recognition Using Deep Learning Approaches

Birhanu, Asrade

http://ir.bdu.edu.et/handle/123456789/12377

Downloaded from DSpace Repository, DSpace Institution's institutional repository
BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES
FACULTY OF COMPUTING
Information Technology Extension Program

Msc. Thesis:
Multi-Script Ethiopic Document Images Recognition Using Deep Learning Approaches

By
Birhanu Asrade Mengistu

Bahir Dar, Ethiopia
January, 2021
BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES
FACULTY OF COMPUTING

Multi-Script Ethiopic Document Images Recognition Using Deep Learning Approaches

By
Birhanu Asrade Mengistu

Thesis Submitted to The School of Research and Graduate Studies of Bahir Dar, Institute of Technology, BDU In Partial Fulfilment of the Requirements for the Degree of Master of Science the Information Technology

Advisor Name: Gebeyehu Belay (Dr.of Eng) Asct Professor

Bahir Dar, Ethiopia
January, 2021
DECLARATION

This is to certify that the thesis entitled “Multi-Script Ethiopic Document Images Recognition Using Deep Learning Approaches”, submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology under the Faculty of Computing, Bahir Dar Institute of Technology is a record of original work carried out by me and has never been submitted to this or any other institution to get any other degree or certificates. The assistance and help I received during the course of this investigation have been duly acknowledged.

Name of the student: Birhanu Asrade Signature

Date of submission: 03/03/2021

Place: Bahir Dar

This thesis has been submitted for final examination with my approval as a university advisor.

Advisor Name: Gebeyehu Belay (Dr. of Eng.)

Advisor’s Signature: Date 04/03/2021
© 2021

Birhanu Asrade Mengistu

ALL RIGHTS RESERVED
BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF RESEARCH AND POSTGRADUATE STUDIES
FACULTY OF COMPUTING

Thesis Approval Sheet for defense result

I hereby confirm that the changes required by the examiners have been carried out and incorporated into the final thesis.

Name of Student Birhanu Asrade Mengistu Signature Date 03/03/2023

As a member of the board of examiners, we examined this thesis entitled “Multi-Script Ethiopic Document Images Recognition Using Deep Learning Approaches” by Birhanu Asrade. We hereby certify that the thesis is accepted for fulfilling the requirement for the Degree of Master of Science in “Information Technology”.

Board of Examiners

Name of Advisor
Getaye Abele (Ad)
Name of External Examiner

Signature Date 04/03/2021

Adane Letta (PhD)
Name of Internal Examiner

Signature Date March 03, 2021

Name of Chairperson
Mekonnen Wogay (Ad)
Name of Chair Holder

Signature Date 04/03/2021

Demegew Lako
Name of Faculty Dean

Signature Date 04/03/2021

Belete Bitesen
ACKNOWLEDGMENTS

First of all, I would like to give my thanks to almighty God. For He is the Source of Wisdom and the beginning of knowledge. I praise and thank Him for His great help throughout this work. I would also thank the Ever-Virgin, St. Marry, Mother of our Lord.

Next, I would like to express my deepest gratitude to my advisor, Dr. Gebeyehu Belay for his commitment, constructive comments, and suggestions right from the moments of problem formulation to the completion of the work. Many thanks and appreciations go to him for showing important directions.

Besides my advisors, I would like to special thank Birhanu Belay (PhD) for his insightful comments, encouragement, and his guidance at the time of problem preparation and design the methodology, and encouraged me succeed in my academic attempt.

Finally, I want to Acknowledge my Wife Alem Birhanu and beloved son Yonatan, and Elesa Birhanu for all their support and love during my stay in the program.
# TABLE OF CONTENTS

DECLARATION .......................................................................................................................... iii
LIST OF TABLES ..................................................................................................................... x
LIST OF FIGURES .................................................................................................................. xi
LIST OF ABBREVIATIONS .................................................................................................... xii
ABSTRACT ............................................................................................................................... xiii

## CHAPTER ONE: INTRODUCTION ......................................................................................... 1

1.1 Background ......................................................................................................................... 1

1.2 Statement of the Problem .................................................................................................... 4

1.3 Objective ............................................................................................................................. 5
  1.3.1 General Objective ......................................................................................................... 5
  1.3.2 Specific Objectives ....................................................................................................... 5

1.4 Significance of the thesis ..................................................................................................... 6

1.5 Scope and Limitation of the Study ..................................................................................... 7

1.6 Methodology of the study ................................................................................................... 7
  1.6.1 Dataset Preparation .................................................................................................... 7
  1.6.2 Preprocessing Data .................................................................................................... 8
  1.6.3 Segmentation ............................................................................................................. 8
  1.6.4 System Modelling and Design .................................................................................. 8
  1.6.5 Implementation Procedure and Tools ..................................................................... 8
  1.6.6 Performance evaluation ............................................................................................ 8

1.7 Organization of The Thesis ............................................................................................... 9

## CHAPTER TWO: LITERATURE REVIEW ............................................................................. 10

2.1. Introduction ....................................................................................................................... 10

2.2. Overview of Writing System .......................................................................................... 10
  2.2.1 Evolution of Ethioptic script ..................................................................................... 12
  2.2.2 Type Feature of Ethioptic script .............................................................................. 13
  2.2.3 Amharic Script ......................................................................................................... 14
  2.2.4 Geez Script .............................................................................................................. 17
  2.2.5 Tigrigna Script ......................................................................................................... 19
  2.2.6 Afaan Oromo Script .............................................................................................. 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3</td>
<td>Resizing Normalization</td>
<td>63</td>
</tr>
<tr>
<td>4.3</td>
<td>Recognition of Proposed Optimal Model Architecture</td>
<td>64</td>
</tr>
<tr>
<td>4.4</td>
<td>Experimental Result</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.4.1 Compile and Train the Proposed Model</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>4.4.2 Test the Proposed Model</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>Discussion of the Result</td>
<td>73</td>
</tr>
<tr>
<td>CHAPTER FIVE: CONCLUSION AND RECOMMENDATION</td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusions</td>
<td>76</td>
</tr>
<tr>
<td>5.2</td>
<td>Contribution of the Study</td>
<td>77</td>
</tr>
<tr>
<td>5.3</td>
<td>Recommendation</td>
<td>78</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>79</td>
</tr>
<tr>
<td>Appendices</td>
<td></td>
<td>87</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1 Total Number of Characters in Amharic Alphabet - ሪል (Fidel) ...................... 15
Table 2 Amharic script core characters (Fidel) “አለ” ...................................................... 16
Table 3 Geez script of characters (Fidel) “አለ” ................................................................. 18
Table 4 Total Number of Characters in Tigrigna Alphabet ሪል (Fidel) ......................... 19
Table 5: Afaan Oromo (Qubee) alphabets and sounds ...................................................... 22
Table 6. Some important pre-processing operations .............................................................. 25
Table 7. The detail analyses dataset .................................................................................... 59
Table 8: Configuration summary in the proposed model layers. ‘k’, ‘s’, and ‘p’ stand for kernel size, stride, and padding size respectively. ......................................................... 69
LIST OF FIGURES

Figure 1. Sample multi-script Ethiopic Bible document contained Amharic, Qubbe, Geez, and Tigrigna text ................................................................. 2
Figure 2. The CNN architecture ........................................................ 33
Figure 3. The convolution operation ................................................. 34
Figure 4. Max-pooling is a convolution neural network ....................... 35
Figure 5. An architecture of LSTM cell Block diagram adopted from (Safarzadeh & Jafarzadeh, 2020) ......................................................... 37
Figure 6: The workflow diagram proposed scheme for text image recognition technique ........................................................................ 45
Figure 7: Sample collected text file image document from Bible with various languages, size and fonts style ................................................................. 58
Figure 8:- The ground truth uses dataset unique Ethiopic characters and punctuation marks ............................................................. 60
Figure 9: To generate Text-line image and GT text file from multi-script data ................. 62
Figure 10:- Sample text line image taken from dataset generated before resize. ........... 63
Figure 11: Sample Ethiopic text line images normalized to a size 32 by 256 pixels. ...... 63
Figure 12: A designed proposed model for multi-script Ethiopian text-line image recognition ........................................................................... 64
Figure 13:- Proposed Model Summary .............................................. 70
Figure 14: Training progress report ..................................................... 71
Figure 15. The training versus validation loss of the proposed was recorded for 10 epochs. .................................................................................. 72
Figure 16: Sample misrecognized multiple text-line images .................... 74
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>ADAptive Momentum Estimation</td>
</tr>
<tr>
<td>BSTM</td>
<td>Bi-directional Long Short-Term Memory</td>
</tr>
<tr>
<td>CER</td>
<td>Character Error Rate</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Networks</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRNN</td>
<td>Convolutional Recurrent Neural Networks</td>
</tr>
<tr>
<td>CTC</td>
<td>Connectionist Temporal Classification</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>MDLSTM</td>
<td>Multidimensional Long Short-Term Memory</td>
</tr>
<tr>
<td>OCR</td>
<td>Optical Character Recognition</td>
</tr>
<tr>
<td>ReLu</td>
<td>Rectifier Linear Unit</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Networks</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>
ABSTRACT

In recent years, deep learning has widely applied to pattern classification, object detection, image segmentation, speech recognition, and another field. Printed text image recognition is one of the most challenge tasks since decades. The text recognition plays an important role in document image processing. It is a type of techniques for document image analysis to recognize the useful content in the text documents to be archived in softcopy for different purposes. The technique involves the conversion of the given image of text to its most probable similar character in a given domain language scripts.

Therefore, traditional machine learning approach deals with time-dependent data. The Ethiopic script uses a large number of characters in the writing and existing visually similar characters, which results in a challenge for OCR development. The study prepared a new dataset contains 20,000 text-line images and 517,610 characters synthetic image. The performance of the proposed multiple writing scripts Ethiopic OCR model is tested by a printed artificial generated dataset. The proposed model addresses the text image recognition problem on the text-line level. In extracting the important feature values of text-line images of the scripts, preprocessing activities such as, binarization, text line segmentation, size normalization activities are performed. Then we proposed a model combine CNN and LSTM with together CTC which was integrated single framework of CNN is used for automatic feature extract from raw images, LSTM is used to learn sequential data, and CTC is used for of transcription of the output of the LSTM to character labels without any post-processing module it can directly decode the input sequence to the output. The experimental results were reported as character error rates (CERs) as results of insertion, deletion, and substitution of the characters in predicted output, and that achieves state-of-the-art result is 9.5% CER.

**Keywords:** multi-Ethiopic script, OCRopus, CNN, LSTM, CTC, OCR and text-line image
CHAPTER ONE: INTRODUCTION

1.1 Background

Nowadays, Digitization is the incorporation of emerging technologies into daily life by the digitization of everything that can be digitized. It can provide a way of retaining the material content by providing an available item facsimile to place less pressure on already fragile original such as printed books. The document analysis groups were formed to address this by digitizing the content, making it easy to exchange, search and allow language translation on the internet.

Optical Character Recognition (OCR) system convert a large number of documents without any transformation, noise, resolution variations, and other variables, either the typed alphabet or handwriting into machine-encoding text. Also, OCR refers to the mechanical or digital conversion into computer code of image of handwriting characters or printed text without any variation (Bhatia, 2014). OCR involves several steps, such as pre-processing, segmentation, extraction of characteristics, classification, and identification. One step’s input is the output of the next step. The preprocessing activity relates to noise reduction and variance in printed documents. Feature extraction is a technique for extracting and capturing certain pieces of information from data. In the classification phase, the portion of the divided text in the document image will be mapped to the equivalent textual representation. Offline handwriting recognition systems and pattern recognition are needed in many areas where OCR is used to include mail sorting, bank processing, document reading, and postal address recognition. OCR is among the oldest and most popular pattern recognition and application technology focused on machine learning that is being enhanced until today. OCR has been developed for many languages so far, such as English (Latin), Devanagari (Hindi), Bangla, etc. But not much work has been done for the Ethiopic script.

According to (A. Graves, 2009), the deep neural networks of the document content recognition approaches can be divided into two broad categories, those were segmentation based or analytical approach and segmentation free or holistic approaches. The
segmentation-based approaches first extract segments, for example connected components, and then perform the recognition on the individual linked components. The segmentation-free approaches work directly at the next text-line level and evaluate the output of the complete text line. The segmentation-free approach method performs an average of five up to ten percent better than segmentation based because manages to have a lower per query cost on unprocessed images.

In Ethiopia, there are different scripts that were multiple writing documents that use multiple scripts for example bus reservation forms, query papers, language translation books, religious books, and money order forms, which may contain text lines in more than one type of script. It is harder than single-script OCR development to develop a generalized OCR system for multi-script documents. This is because the character recognition features are necessary depending on the structural characteristics, style, and nature of writing, which usually vary from one script to another. A document can be categorized as either a single or multi-script document depending on its content. If the document contains just one script text, then we call it a single script document. If there are two or more scripts in a text, we call it bi-script or multi-script. Several countries use two or more scripts. For example, in Ethiopia a multi-script like Amharic, Geez, Tigrigna, and Qubbe Ethiopic document. Multi-script recognition is a process, which associates various script objects (words) drawn on an image i.e., Multi-script recognition techniques associate a word identified with the image of a multi-script (Nabin Sharma U. Pal, and R. Jayadevan, 2012).

Figure 1. Sample multi-script Ethiopic Bible document contained Amharic, Qubbe, Geez, and Tigrigna text.

Deep learning is a powerful set of techniques for learning in neural networks. It is about learning multiple levels of representation and abstraction that help to make sense of data
such as images, sound, and text. It requires a larger number of training datasets to be used. It contains a fast and reliable set of algorithms consisting of multiple layers that enhance image recognition accuracy. Such deep learning algorithms such as Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), and Connectionist temporal classification (CTC). Convolutional Neural Network (CNN) is a multi-layer feed-forward neural network that extracts features and properties from the input data (images or sounds). CNN is trained with a neural network back-propagation algorithm. CNN has the ability to learn from high-dimensional complex inputs, nonlinear mappings from a large number of data images or sounds (Durjoy Sen Maitra, 2015). The Long Short-Term Memory (LSTM) (S. Hochreiter & J. Schmidhuber, 1997) is used to encode a sequence of inputs to a single output. When as Bi-directional LSTM summarizer uses a forward LSTM followed by a backward LSTM to aggregate the sequence. Additionally, the Connectionist Temporal Classification (CTC) model is a free alignment algorithm and higher performance sequence prediction and labeling tasks. CTC works by summing over the probability of all possible alignments between the input and the label. CTC consists of two steps: by taking the most probable character per time-step, it determines the best path and undoes encoding by first removing repeated characters and then removing all blanks from the path (Zhang, Pezeshki, & Bengio, 2016).

To develop a script Ethiopian text OCR system that can recognize handwritten or machine-printed documents, we have to extract a sequence of features from the input automatically. Classification and prediction techniques are applied to extended features of the input data using traditional approach deal with time-dependent data such as Neural Network (NN), SVM and Hidden Markov Model (HMM). In addition, NN and HMM combinations are also tested and show a good performance in OCR tasks. However, recent works on OCR tasks using Convolutional Neural Network (CNN), Long Short-term Memory (LSTM), and Connectionist Temporal Classification (CTC) methods are more effective, typically faster train and remove time consumed than the previous techniques (Yousefi, Breuel, & Stricker, 2015). Therefore, CNN automatically extracted features from raw images and BLSTM followed by CTC for sequence labeling. Then, in several tasks of pattern recognition and
machine learning, the combination of CNN and BLSTM has already achieved excellent results.

The goal of this paper is to prepare the printed dataset for training and testing purposes using Amharic, Geez, Qubee, and Tigrigna of mixed writing documents and then to apply a deep learning approach to achieve very character low error rates. Therefore, in this paper, we developed a new proposed model using a combination of CNN and BLSTM together with CTC architecture and to get high-performance multiple writing Ethiopian documents image recognition.

1.2 Statement of the Problem

Nowadays, to understand multi-script recognition, several kinds of the research has been performed. But the issue of data sharing between humans and computing machines is a challenging one. Even at present, many algorithms have been proposed by many researchers so that these multi scripts like Hindi, English, Urdu, other scripts can be easily recognized (Israr Uddin, 2017). However, multi-script Ethiopian printed documents are not the efficiency recognition OCR system.

Since the Ethiopian multi-script is a document that contains text data, it is used in more than one script. It is also difficult to automatically recognize in a text document containing several scripts and fonts because it is not only related to the shape, size, and design of the characters and symbols used to render the text but also mixed with more separate variables, such as the page's shape and size, the structure of the written text, the information density, the directionality of the composition of the text, etc.

The various language of text recognition used under the time limitation. Since, text image has many problems such as text-line detecting issues, noise, skew, variance size and font, the brightness of color, etc. Therefore, text line recognition is still challenging parts and needs to be studied about some multiple documents, which would need to be improved OCR algorithms about different languages like Amharic, Geez, Oromia, and Tigrigna. The
training with mixed recognition scripts has challenges because of, its complexity and loss of structural information, and multi-script Ethiopic text recognition is difficult to implement with a machine. The process must be done automatic multiple texts at the same time, without select the single text image. The performance of character recognition is depending upon the quality of scanned documents. The pre-processing steps are used to removing low-frequency background noise, normalize the intensity of individually scanned documents. For the existing algorithm there are several factors are depending on character level including’s for the requirement of pre-segmented training data, vanishing gradient problem and for existing OCR is time-consuming for segmentation-based steps (Kaur S & PS, 2016). The research proposed to uses one of the deep learning algorithms to study multiple writing Ethiopic documents recognition.

The research issue formulated as the following research questions:

- How to develop text image recognition model for multiple writing Ethiopic documents.
- Which deep learning algorithms are better for multi-script Ethiopic document image recognition?
- To what extent deep learning can recognition of multi-script Ethiopic documents image recognition?

1.3 Objective

1.3.1 General Objective

The main objective of the study is to design and implement an OCR system that recognizes printed Ethiopic documents written using multiple scripts so as to enable users to edit documents written in multiple scripts.

1.3.2 Specific Objectives
To study the general and basic step of the character recognition system

➢ To apply an appropriate recognition algorithm model for multi-script Ethiopic documents image recognition.
➢ To develop a multi-script Ethiopic text recognizer model and demonstrate the proposed algorithm works for such OCR.
➢ To implementation and developing a program to test the algorithms and make the analysis of the result.
➢ To evaluate the performance of the proposed model using a testing dataset.

1.4 Significance of the Thesis

The outcome of the study would have a better success for Multi-script Ethiopic documents image recognition. There are many OCR systems available for handling the clean printed documents in Devanagari, Gurumukhi, English, Japanese, and Oriya, etc. But till now there is no software available for the recognition of multiple writing documents of Ethiopic text. As such, there is a need for printed OCR for Ethiopic multi-script that can help the people for converting the text to computer processable format. The software would be developed can be used for image recognition of multiple Ethiopic documents. There is a need to develop OCR. The set of features extract about the text line image can be utilized to reconstruct the text for editing purposes.

• A reusable component that can be used with other applications or devices which depends on extracting textual content from images.
• To save space in replacing huge collections of documents like in the National Library of Ethiopia and Conversion of books to digital libraries
• Sorting of the large document (legal, historical, security)
• Enhance search engine capabilities in dealing with images.
• Reading books and documents for the visual impaired.
• Enable people with limited vision to read Ethiopic text from the image.
• Make dealing with softcopy easier, such as reading, copy, cut, delete, search and print.

1.5 Scope and Limitation of the Study

The scope of the study to analyze and design the approaches for multi-script Ethiopic printed documents image recognition using a combination of CNN and BLSTM with the CTC approach. Even though the work only Ethiopic scripts that were used for Amharic, Geez, Tigrigna and Qubee alphabets, regarding the dataset, the present research focused only on writing documents on “Visual Geez” font type for Geez, Amharic, Tigrigna, and Affan Oromo scripts, the test considered common font size and style.

Therefore, the work not considered other Ethiopic scripts however, the study would be extended easily to cover other languages that use Ethiopic script for their writing system. Also, the promising findings are experimental in the study. The following major limitations are recorded; the study would not have included shapes/symbols, tables, and mathematical formulas, and others. The main challenges for doing this research were constraints in the platform for doing processing large-scale datasets because of high GPU and high CPU need processing beyond using huge datasets to classify and labeled.

1.6 Methodology of the Study

To achieve the general, specific objectives and address the research problem would be use different research methodology including tools and algorithms that used in this research work and approaches followed at each step.

1.6.1 Dataset Preparation

Two types of dataset would be prepared for training and testing of Ethiopic OCR system the following. Data collection forms would be prepared in text documents. Collecting the images of scripts of mixed different characters (Amharic, Geez, Qubee, Tigrigna) for
training as well as testing purpose, for the training dataset, Ethiopic Script in commonly used font types, sizes, and styles would be prepared and for the test dataset a collection of real-life documents which contain multi-Ethiopic scripts from historical documents, and others would be synthetic dataset prepared.

1.6.2 Preprocessing Data

Pre-processing techniques using OCROPUS tool publicly available to convert the document to ground truth and text line image corresponding would be carried out.

1.6.3 Segmentation

Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). Segmentation used for text-based images aim in retrieval of specific information from the entire image. The paper proposed method to segment a text-based image at various levels of segmentation. At this stage the page of the input images is decomposed into text line segmentation would be used.

1.6.4 System Modelling and Design

Design system model the CNN and BLSTM network with CTC used, selecting training algorithm and modelling the system feature extracted, sequence learner and prediction at each time-step as an input script respectively.

1.6.5 Implementation Procedure and Tools

Python programing would be used to implement for the proposed system recognition. Python would be used to develop the system. To operate on the Linux Ubuntu operating system of HP, Intel(R) Core (TM) i7-8700U CPU @ 4.6GHZ processor, with 16 GB DDR4-2666 SDRAM.

1.6.6 Performance Evaluation
Performance would be measured using character error rate. Also, to evaluate the performance of our proposed method can be calculated as the ratio of correctly predicted values to total number of prediction while error rate is the ratio of wrongly predicted value out of the total number of predictions.

1.7 Organization of The Thesis

The thesis is organized into five chapters. The first chapter of the thesis includes the background, the statement of the problem, and its contribution. It also includes the objectives of the work and discusses the methodology used to accomplish the research. The second chapter discusses the basic methods of preprocessing techniques of character images along with the target Ethiopic writing system of this research. Different character recognition techniques reviewed from the literature are also discussed. The third chapter develop the proposed model OCR system and discusses how to work the proposed model with each technique for the multi-script Ethiopic characters recognition. Chapter four deals with the experimentation activity undertaken to configure and implement the model and tries to implement the technique to the problem domain of interest are presented. The section also presents the recognition rate achieved for different test cases after the design and training of the neural network classifier. Based on the results of the experiment, chapter five presents the conclusion and recommendations of the research.
CHAPTER TWO: LITERATURE REVIEW

2.1. Introduction

In this chapter the overview of each Ethiopic script writing system for Amharic, Geez, Tigrigna and Qubee, the basic concept for techniques of OCR phase systems, and the most commonly used text recognition techniques, the architectures of deep learning type of CNN, LSTM, and Connection Temporal Classification, and related works are discussed.

2.2. Overview of Writing System

This research focuses on identifying the Ethiopic script writing system of a document containing mixed Amharic, Geez, Tigrigna, and Afaan Oromo language scripts for OCR purposes. Ethiopia had more than 80 nations and nationalities and follows the federal government system (Wang.K & Wang.Q. , 2009). The situation allows regional governments to use their own language and their choice of the script for the official work. The understanding the evolution, and the feature of these four scripts is an important input in the development of script identifier which handles these four language scripts. This section debates different concepts related to writing systems. The historical development, evolution and features of Amharic, Geez, and Tigrigna language scripts and Qubee features of Afaan Oromo language scripts were well discussed here.

A writing system, sometimes called a scriptwriting system has a set of rules universally understood by a group of people who systematically order and use symbols, each represents a certain term or word. The generic term for the symbol in a writing system is a character. That is a set of traceable symbols used to visually embody units of language in an orderly way, and it is a means of communicating through symbols that represent whole words or sentences in a given spoken language (Lawrence , 2010), (Sertse, 2011). Many scholars agreed on the suggestion that by the end of the 4th millennium BC, the first writing system originated among the Sumerians; the alphabetic writing system, however, began in Egypt
and the Indus valley around 2000 BC. Since then, writing has appeared several times separately, affiliated with different cultures (Million, 2008), (Assabie & Bigun, 2006).

There are many writing system families that support different users of the language. According to (Bernard, 2010), world scripts can be grouped in one of the following writing system families: Alphabetic, Consonantal (abjad), Syllabic (Abugida), Alpha-syllabic, and Logographic.

**Alphabetic**: is a simple writing form in which the phoneme is loosely represented by of the characters mixed up. A mixture of consonant and vowel characters is represented by phonemes. For example, for the representation of a phoneme, usual English spellings can be represented with consonants and vowels.

**Consonantal (Abjads)**: a consonantal writing system is a variation of the added element of an alphabetic writing system where only consonants, not vowels, are added. From the consonant combination sense, users are left to guess the vowels. Hebrew and Arabic may be examples for this family,

**Syllabic (Abugida)**: a syllabic writing system is an ideal version of the writing system. In this writing system family, a theoretically distinct symbol represents each syllable of the language in question. The Japanese hiragana syllabify comes close to an ideal example for this family, according to discussed (Bernard, 2010). Some studies have grouped the writing systems of Ethiopia and Devanagari into this family. However, instead of producing a distinct type of character in these writing systems, they supported adding diacritic on the base character to create the vowel for that base character.

**Alpha-syllabic**: Between the alphabetic and syllabic forms, there is an alpha-syllabic forms. The fundamental character indicates a consonant in alpha-syllabic writing systems, while a diacritic is added, or often some other transformation is introduced to signify a combination of that consonant with the following vowel. An example of an alpha-syllabic writing system is an Ethiopian script defined below the sample alpha-syllabic writing system of Ethiopian scripts. For example, Amharic scripts like this,
The alpha-syllabic Ethiopic script is used in Ethiopia for most indigenous languages, including Amharic, Geez, and Tigrigna, the working languages of the Federal Government. In the combination of consonant and vowel, Ethiopia has several anomalies, thereby moving them into the syllabic family more. Ethiopic scripts were also classified in the syllabic fam by a large number of other researchers (Bender M., 1976).

**Logographic:** The linguistic unit which forms the basic representational unit of the writing system has been recognizable in phonetic terms in all the writing systems discussed so far. The basic unit of representation in a logographic writing system is the morpheme. The best examples of this writing system may be Chinese scripts.

### 2.2.1 Evolution of Ethiopic script

The Ethiopic script has its origin in the same family writing systems as those of European alphabets, namely the Semitic scripts that proliferated in the Middle East more than three thousand years ago (Coulmas, 1989). Ge’ez language is a Semitic language that was widely used in Ethiopia and Eritrea until the 10th to the 12th century. Ethiopic (Ge’ez) script developed for the writing system of the Ge’ez language. As of nowadays, in excess of 43 Ethiopian and Eritrean Semitic languages, for example, Ge’ez, Amharic, Tigrinya, and so on use Ethiopic script as a writing system.

As debated in (Thomas B., 1995), the Ethiopic script was originally used in the representation of Ge’ez language inscriptions. The first Ge’ez inscriptions in the Ethiopic script can be traced back to the 4th century AD. Geez was the language of the empire of Aksum (a flourishing Semitic civilization based in what is now Northern Ethiopia). Amharic has been the political language of Ethiopia for many hundreds of years. Though no clear pieces of evidence are available, whether Amharic is descended from Geez or not, it is fairly certain that the Ethiopic writing system was passed on from Geez to Amharic.
In the Amharic language, the oldest Ethiopian script documents are songs and poems dated from the 14th century. Important literature in any quantity, however, did not begin until the 19th century. Since then, the Ethiopic script is the normal medium for newspapers, magazines, novels, poetry, primary school texts, official, and legal documents and other printed matter as well as for private correspondence of Amharic language users, Ethiopic script set was first presented to computerization in 1991 by Ethiopian Science and Technology Commission (ESTC). Since then, different software developers developed different font faces for Ethiopic scripts independently (Wasu, 1991).

2.2.2 Type Feature of Ethiopic script

There are disputes between scholars in the syllabic family as to whether or not to group Ethiopic scripts. In theory, an alphabet has individual symbols with the individual vowel and consonant symbols representing phonemes; a consonant system represents only consonants, leaving the reader to guess the vowels; and for each syllable, a syllabic has individual signs. According to (Bender, S.W., & Cowley, 1976), (Gragg, 1996), grouped Ethiopic script to Syllabic since it uses one character per syllable. However, (Sampson, 1995), explicitly rules it out of the syllabic category. The argument is that it should have unrelated symbols for phonologically similar syllables in order to be a true syllabic. Then again, the Ethiopian system can be analyzed as 33 fundamental consonant forms with relatively systematic variations to indicate vowels and/or labialization, also grouped to alpha syllabic family Ethiopic script. A number of characters: the Ethiopic script uses more than 500 characters, as a writing system. In this number of characters, developing a recognizer for Ethiopic script is challenging in addition memory and computational requirements are very demanding. So, discusses the following type of Ethiopic language script for used the research. Based on the similar nature of Amharic, Geez, and Tigrigna Characters of their shape, according to discussed (Taddeesse, 2000) grouped base Amharic characters into five categories

1. Symbols that have one straight ‘leg’.

   e.g., ዓ, ኃ, ከ, ኮ, ኯ
2. Symbols that have two straight 'legs'

   e.g., እ, ኡ, ኢ, ኣ, ኤ, እ, ኦ, ኧ

3. Symbols that have three 'legs'.

   e.g., ኧ, ከ, ዲ

4. Symbols that have rounded bottom.

   e.g., ኰ, ኱, ዐ

5. Symbols that have horizontal bottom line.

   e.g., ኲ, ኳ.

According to (Bender, S.W., & Cowley, 1976), in two ways, Ethiopic script vowels are formed from consonants. By shortening or lengthening one of its main strokes, the fourth and seventh orders mostly take on a modified form of the base character. While vowels derive from the second, third, and fifth order of the core characters by adding small appendages, such as strokes, loops to the right, left, top, or bottom of each form of base characters. However, there are irregularities among appendages of the same order. For instance, the fourth, sixth, and seventh orders are highly irregular.

Ethiopic scripts also have size differences both in length and width. Examined, short characters like ኴ, ኵ and ኶ and long characters ኸ and ኹ. Widthwise differences are also discussed among character like ኴ, ኵ, and ኶. Ethiopic is written from left to right. According to (Bender M., 1976), (Thomas B., 1995), the Ethiopic system makes no division between upper- and lower-case letters and has no conventional cursive form, though, rapid handwriting can result in an ad hoc cursiveness and often a lack of clear distinctions. Finally, for other languages including Geez, Argobba, Gurage, and Tigre, the Ethiopic script used for Amharic is often used which are chiefly liturgical languages.

**2.2.3 Amharic Script**
The official language of the Ethiopian government and the 2nd largest Semitic language next to Arabic is the Amharic language, also known as “አማርኛ”. It is spoken primarily in Ethiopia's central highlands. It is the Afro-Asian language of the Southwest Semitic group and is linked to the liturgical language of the Ethiopian Orthodox Church; it is also linked to Tigre, Tigrinya and the dialects of South Arabic, discussed by (Britannica., 2008). Amharic is written in a slightly modified form of the alphabet used to write the Geez language called Ethiopic script. The Ethiopic system is used on a large scale in the representation of three Semitic languages, all confined to Ethiopia and Eritrea. These are Ge’ez (the liturgical language of the Ethiopian Orthodox Church), Amharic, and Tigrigna.

According to (Assabie & Bigun, 2009), Over the past two millennia, the Ethiopian script has been used effectively as a writing system for languages spoken in Ethiopia, with a population currently in excess of 80 million. There are about 356 distinct alphabets in the Amharic script, including 238 core characters, 89 labialized characters, 9 punctuation marks, and 20 numerals written and read from left to right, as in English (Endalamaw, 2016) (Halefom & Chen, 2018). All vowels and labialized characters are, with a minor modification, derived from the 34 consonant characters in the Amharic script. The modification involves the composition of the characters by inserting and elongating a straight line or shortening their main legs. It also requires inserting small diacritics to the right, left, top, or bottom of the characters, such as strokes or loops/circles. Table 1 a summary of the number of Amharic characters in each group.

<table>
<thead>
<tr>
<th>Type of Amharic character</th>
<th>Number of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Core character</td>
<td>238</td>
</tr>
<tr>
<td>2 Labialization character</td>
<td>89</td>
</tr>
<tr>
<td>3 Punctuation mark</td>
<td>9</td>
</tr>
<tr>
<td>4 Numerals</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>356</td>
</tr>
</tbody>
</table>

Table 1 Total Number of Characters in Amharic Alphabet - እምርኛ (Fidel)
Ethiopic scripts have their own features that are unique to the writing system. Ethiopic has additions, conventionally added to the base character to give a derived vocal sound. Amharic writing system consists of thirty-four characters (called Fidel/“ፊደል”) as a core character. The 34 core characters occur in seven orders, according to (Assabie & Bigun, 2006) the Amharic language alphabet is suitably written in a tubular from seven-column, where each column corresponding to vocal sounds in the ä, u, i, a, e, ə, and o as shown in Table 2.

Table 2 Amharic script core characters (Fidel) “ፊደል”.

<table>
<thead>
<tr>
<th></th>
<th>1stä order</th>
<th>2nd u order</th>
<th>3rd I order</th>
<th>4th a order</th>
<th>5th e order</th>
<th>6th o order</th>
<th>7th o order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
</tr>
<tr>
<td>2</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
</tr>
<tr>
<td>3</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
</tr>
<tr>
<td>4</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
</tr>
<tr>
<td>5</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
</tr>
<tr>
<td>6</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
</tr>
<tr>
<td>7</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
</tr>
<tr>
<td>8</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
</tr>
<tr>
<td>9</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
</tr>
<tr>
<td>10</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
</tr>
<tr>
<td>11</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
</tr>
<tr>
<td>12</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
</tr>
<tr>
<td>13</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
</tr>
<tr>
<td>14</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
</tr>
<tr>
<td>15</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
</tr>
<tr>
<td>16</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
</tr>
<tr>
<td>17</td>
<td>ኪ</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
</tr>
<tr>
<td>18</td>
<td>ካ</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
</tr>
<tr>
<td>19</td>
<td>ኬ</td>
<td>ኦ</td>
<td>ኧ</td>
<td>ከ</td>
<td>ኩ</td>
<td>ኪ</td>
<td>ካ</td>
</tr>
</tbody>
</table>
2.2.4 Geez Script

Geez is written with characters from Ethiopia or Geez abugida, a script originally developed specifically for this language. The script is called Fidel (አፈል), which means script or alphabet, in languages that use it, such as Amharic and Tigrigna. From left to right, geez is written/read. Geez, there are 26 main characters in the writing system (called “Fidel”/ “አፈል”) and seven orders. The transformation is from Geez to Amharic. Taken from (Worku Alemu, 1997) research’s Amharic took all the 26 symbols which were used in the Geez language and added several new symbols to represent sounds not found in Geez. These symbols are ቧ, ቫ, ቭ, ቯ, ቴ, እ and እ. It must be acknowledged also that there are no upper- or lower-case distinctions in Geez. The total Geez characters are 182 without Geez numbers and diacritics.
Table 3 Geez script of characters (Fidel) “ፇፇፏ”

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ዊ</td>
<td>ዊ</td>
<td>ዊ</td>
<td>ዊ</td>
<td>ዊ</td>
<td>ዊ</td>
</tr>
<tr>
<td>2</td>
<td>እ</td>
<td>እ</td>
<td>እ</td>
<td>እ</td>
<td>እ</td>
<td>እ</td>
</tr>
<tr>
<td>3</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>4</td>
<td>ዤ</td>
<td>ዤ</td>
<td>ዤ</td>
<td>ዤ</td>
<td>ዤ</td>
<td>ዤ</td>
</tr>
<tr>
<td>5</td>
<td>ዞ</td>
<td>ዞ</td>
<td>ዞ</td>
<td>ዞ</td>
<td>ዞ</td>
<td>ዞ</td>
</tr>
<tr>
<td>6</td>
<td>ዖ</td>
<td>ዖ</td>
<td>ዖ</td>
<td>ዖ</td>
<td>ዖ</td>
<td>ዖ</td>
</tr>
<tr>
<td>7</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
</tr>
<tr>
<td>8</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>9</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
</tr>
<tr>
<td>10</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>11</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
</tr>
<tr>
<td>12</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
<td>዗</td>
</tr>
<tr>
<td>13</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>14</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>15</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>16</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>17</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>18</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>19</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>20</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>21</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>22</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>23</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>24</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>25</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
<tr>
<td>26</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
<td>ዘ</td>
</tr>
</tbody>
</table>
2.2.5 Tigrigna Script

The Semitic (Afro-Asiatic) language is Tigrigna. A language family that includes modern languages such as Tigrinya, Tigre, Amharic, Hebrew, Arabic, Maltese (from Malta), and Aramaic belongs to the Semitic (derived from the Biblical "Shem") languages. Approximately 7 million people around the world speak Tigrinya. In Eritrea and in the northern part of Ethiopia, it is a widely spoken language. In Eritrea, it is a working language in offices alongside Arabic. Tigrinya is also spoken by many immigrant communities worldwide, such as in Sudan, Saudi Arabia, the United States, Germany, Italy, the United Kingdom, Canada, Australia, Uganda, etc. Tigrigna is written in the Geez script, originally created for the now extinct Ge'ez language.

The Ge'ez script is an abugida: each symbol is a consonant plus a vowel syllable, and the symbols are arranged on the basis of both the consonant and the vowel in groups of similar symbols. written by (Rehman, 2017). Like the English language, it is written and read from left to right. In the Alphabet of Tigrigna, there are 32 sets of characters. At the beginning, the Ge'ez script looks complicated, but learning how to read in Tigrigna doesn't take long. Pronunciation is very simple and straightforward in Tigrigna. When it is necessary to represent a consonant with no following vowel, the consonant + ø form is used. For example, the word ’ǝntay ‘what?’ is written “እንታይ”, literally ’ǝ-nǝ-tǝ-yǝ’. In table 4 below the columns are assigned to the seven vowels and script of Tigrigna they appear order rows are assigned to the consonants. The total Tigrigna characters are 245 basic and 30 Labialization character without Geez numbers.

Table 4 Total Number of Characters in Tigrigna Alphabet (Fidel)

<table>
<thead>
<tr>
<th>Å</th>
<th>u</th>
<th>i</th>
<th>a</th>
<th>e</th>
<th>(ə)</th>
<th>o</th>
<th>wä</th>
<th>Wi</th>
<th>wa</th>
<th>we</th>
<th>wə</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>u</td>
<td>u</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Labialization characters symbols to represent special features, a dataset also include most of which are used to represent two vocal sound example \( \digamma \) for \( \digamma \) list of the following Such as \( \lambda \distinct \), \( \nu \distinct \), \( \delta \distinct \digamma \), \( \theta \distinct \), \( \varphi \distinct \), \( \eta \distinct \), \( \xi \distinct \), \( \zeta \), etc.

Punctuation consisting of word divider Hulet netib (:), end of the sentence indicator arate netib (: :), drib serez (፣) netela serez (ፉ), and other symbols inherited from the Latin language like a question mark (❓), the exclamation mark (!), quotes (“”) and parenthesis ( ).

Amharic, Geez and Tigrigna Numerals (‘Kutroch’) which consists of symbols from 1 to 9 (፩, ዘ, ዠ, ዡ, ዢ, ዣ, ዤ, ዥ, ዦ) for multiples of 10 (10 – 90) it has ዧ, የ, ዩ, ዪ, ያ, ዬ, ይ, 100 (፱), 1000 (፲፱) Numerals.

### 2.2.6 Afaan Oromo Script

According to (Tullu Guya, 2003) discussed Afan Oromo, also called Oromiffaa or Afaan Oromoo, is a member of the Afro-Asian language family's Cushitic branch. After Hausa and Arabic, it is the third most widely spoken language in Africa. The region's initial homeland covers most of what is now Ethiopia, Somalia, Sudan, and northern Kenya, as well as parts of other East African countries. (Abera, 1988). It is officially the official language of the regional state of Oromia (which is the biggest region among the current federal states in Ethiopia). It is used by the Oromo people, Ethiopia's largest ethnic group, which in 2007 accounted for 50% of the total population (2015 Census statistic of Ethiopia). Qubee (Latin-based alphabet) was adopted with regard to the writing system and became the standard Afan Oromo script from 1991 on (Debela, 2010).

According to (Gragg, 1996) language structure, Afan Oromo has a very rich morphology like other African and Ethiopian languages. With regard to the writing system, Qubee (Latin-based alphabet) has been adopted and become the official script of Afan Oromo.
since 1842. The writing system of the language is straightforward, which is designed based on the Latin script.

Afan Oromo uses Qubee (Latin alphabets) consisting of 33 letters, five of which are vowels, 24 which are consonants, seven of which are paired letters, and fall together a combination of two consonant characters like ‘ch’. The Afan Oromo alphabets are characterized by capital letters and small letters, as with English alphabets. In the Afan Oromo language, as in the English language, the vowels are sound makers and are sound by themselves. Vowels in Afan Oromo are characterized as short and long vowels. The complete list of the Afan Oromo alphabets is found on the manuscript by (Tullu Guya, 2003). Qubee uses Arabic numeric styles included in this study like 1-9 and 10-1000 etc., numbers also recognized. Qubee’s characters show in table 5, of capital and small alphabets.

Table 5: Afaan Oromo (Qubee) alphabets and sounds

<table>
<thead>
<tr>
<th>Aa</th>
<th>Bb</th>
<th>Cc</th>
<th>Ch ch</th>
<th>Dd</th>
<th>Dh dh</th>
<th>Ee</th>
<th>Ff</th>
<th>Gg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Ba</td>
<td>ca</td>
<td>cha</td>
<td>da</td>
<td>dha</td>
<td>e</td>
<td>fa</td>
<td>ga</td>
</tr>
<tr>
<td>Hh</td>
<td>Ii</td>
<td>Jj</td>
<td>Kk</td>
<td>Ll</td>
<td>Mm</td>
<td>Nn</td>
<td>Ny</td>
<td>Oo</td>
</tr>
<tr>
<td>Ha</td>
<td>I</td>
<td>ja</td>
<td>ka</td>
<td>la</td>
<td>ma</td>
<td>na</td>
<td>nya</td>
<td>o</td>
</tr>
<tr>
<td>Pp</td>
<td>Ph</td>
<td>ph</td>
<td>Qq</td>
<td>Rr</td>
<td>Ss</td>
<td>Sh</td>
<td>sh</td>
<td>Tt</td>
</tr>
<tr>
<td>Pa</td>
<td>Ph</td>
<td>ph</td>
<td>Qq</td>
<td>Rr</td>
<td>Ss</td>
<td>Sh</td>
<td>sh</td>
<td>Tt</td>
</tr>
<tr>
<td>Wa</td>
<td>Xa</td>
<td>ya</td>
<td>za</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aa</td>
<td>Ee</td>
<td>ii</td>
<td>oo</td>
<td>uu</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**2.3 Basic Optical Character Recognition (OCR) Techniques**

Optical Character Recognition (OCR) is a process that enables printing (typewritten, printout as well as handwritten) to be optically and translated into machine readable format. The terms OCR is involved approaches such as scanning documents through scanning machines (handheld or flatbed scanners), segmenting each character in the scanning
documents, extracting features of a character (which may help to differentiate the character from others), and match these features to the ones already stored to identify the character. OCR are carried out system of automated optical character readers. OCR can be described as the methods of into printed or handwriting images of machine numbers, letters, and symbolic to a format capable of computer processing. The long history of this area’s study, commercial success, and the continuing need and ability to handle less restricted forms of text make OCR the most important application area in machine perception to date (Genovese, 1970).

The first OCR was started in the 1960s as a major initiative in postal number recognition. Gradually, there came success in the reading of postal numbers. At first, high performance of correct reading was not required, although further research and development was expected to improve the reading performance. This fortunate situation can be contrasted with speech recognition, in which essentially continuous speech recognition was required from the beginning. This level of difficulty corresponds to cursive script character recognition, which is a current topic of investigation. Thus, OCR has grown up as an industry, aided by the rapid development of computer technology. According to discussed (Ralston, 2000), OCR technology is still at a stage of development. The points out that an OCR system usually consists of modules for preprocessing, segmentation, and recognition. Digitization, skew detection, noise removal, binarization, slant correction, thinning, normalization, and others belong to the preprocessing stage of the overall recognition system. In the segmentation step, the text image is separated into individual characters. Two essential components in a character recognition step are feature extraction and classification.

For multilingual script OCR, whereas the OCR problems look almost solved, in several research papers multilanguage scripts recognition from a single document text image is still under investigation. According to (Kunte & Samuel, 2007) there is no OCR technique that can work in the same document simultaneously for two or more scripts. To emphasize that the multi-script OCR mentioned in bilingual and multilingual OCR is current research focused that involves many complexities within itself. To developing a multi-script OCR
two types of approaches are defined. The first approach to script recognition is achieved in this approach before the final multi-script OCR. Using the result information from script recognition, the corresponding OCR build for that particular script is invoked. In this approach the search space in the datasets is decreases when the script recognition process is involved. Several types of researcher are published in the script recognition method. Script recognition is a proposed task for documents image that contains more than one script, according to the model proposed (Sertse, 2011) Amharic and English from the paper for multi-scripts and (Singh & Kiran, 2016) proposed model multi-script printed text. The second mixed dataset approach character from all the participating scripts are handled identically regardless of scripts in the combined data approach. However, the search space in the data increase because it includes alphabets from all of the script involved. Therefore, script identification processes may vary from approach to approach.

Optical character recognition is the process of classifying optical patterns found in a digital image (OCR). In this section, we describe the main significant phases and architecture of optical character recognition. In these steps, image acquisition, pre-processing, extraction of features, classification, and post-processing. We must recognize the difficulties that can occur in each stage in order to achieve a high rate of character recognition in order to design an efficient application relevant to the OCR system.

2.3.1 Image Acquisition Phase

Image acquisition to capture the image from an external source like a scanner or a camera etc. Image acquisition is the initial phase of OCR that involves acquiring and transforming a digital image into an appropriate form that can be easily processed by a computer. This can include both quantization and image compression (Lazaro, J., Martín, J.L, Arias, J., Astarloa, A., & Cuadrado, C., 2010). Binarization, which includes only two levels of the image, is a special case of quantization. In certain instances, a binary image is adequate for the image to be described.

2.3.2 Pre-processing Phase
Preprocessing first step in the deep learning workflow to prepare raw data in a format that the network can accept. The aim of pre-processing is to eliminate the undesired characteristics of an image's noise without losing any important details. Preprocessing techniques are required for color, gray-level, or binary document images containing text and graphics. Since color image processing is much more expensive, binary or gray images are used in character recognition system for most applications. Inconsistent information and noise are minimized by preprocessing. Therefore, decreasing the noise that causes the reduction in the character recognition rate is the main important issue in preprocessing phase taken from (Bhatia, 2014).

Therefore, because preprocessing controls the suitability of the input for the successive stages, the preprocessing stage is a primary stage prior to the extraction stage of the function. In preprocessing stage and operations, binarization, noise reduction, skew correction, morphological operations, slant elimination, filtering, thresholding, and thinning, this segment used to be discussed. Some important preprocessing issues with a short description are shown in Table 6.

Table 6. Some important pre-processing operations

<table>
<thead>
<tr>
<th>Processes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digitization</td>
<td>Paper image documents converted into digital form by a process of scanning and digitization or to digital image</td>
</tr>
<tr>
<td>Binarization</td>
<td>Separates image pixels as text or background.</td>
</tr>
<tr>
<td>Noise Reduction</td>
<td>Better improvements of image acquisition devices produced by the advancements in technology.</td>
</tr>
<tr>
<td>Skew Correction</td>
<td>Because of the possibility of rotation of the input image through captured image device, document skew should be corrected.</td>
</tr>
<tr>
<td>Morphological Operations</td>
<td>Adding or removing pixels to the characters that have holes or surplus pixels.</td>
</tr>
<tr>
<td>Thresholding</td>
<td>For an image, separating information from its background.</td>
</tr>
<tr>
<td>Thinning and Skeletonization</td>
<td>Thinning process is the Skeletonization, which regularize the map of the text until reaches most medial one-pixel width</td>
</tr>
</tbody>
</table>
2.3.3 Segmentation Phase

The critical and major component of an Optical Character Recognition (OCR) system is the segmentation of text lines from images. Segmentation is a mechanism which determines the components of an image. Figures and graphs are the most important point that is needed to identify the regions of the document where data is printed and distinguished. To better reorganize the editable text lines from the known characters, segment the text line first, then segment the words from the segmented line, and then from that the characters are segmented. As according discussed by (Kaur S & PS, 2016) document segmentation is a major pre-processing phase in implementing an OCR system. It is the process of classifying a document image into homogeneous zones, i.e., that each zone contains only one kind of information, such as text, a figure, a table, or other images.

The separation of characters or phrases or lines is text segmentation. Several segmentation algorithms in which segment terms are used in personally recognized isolated characters. This segmentation process is carried out by isolating each part that is related. In several instances, the precision rate of OCR-related systems depends heavily on the accuracy of the page segmentation algorithm used. As follows, there are three types of the record documentation algorithm.

- Top-down methods
- Bottom-up Methods
- Hybrid Methods

The top-down method recursively divides wide areas into smaller sub-regions in a text. When the criteria are met, the process of document segmentation will stop and the obtained ranges constitute the results of final segmentation at that point. But, bottom-up approaches start by looking for pixels of interest and then grouping pixels of interest. These interest pixels are then managed into linked components that constitute characters that are then combined into words and lines or blocks of text. Hybrid techniques are called the synthesis of both top-down and bottom-up processes. With respect to various aspects of the OCR
system, many approaches to segmentation have already been suggested over the past decades.

According to (Million, 2000), there are basically two commonly used segmentation algorithms: stage by stage and recursive segmentation. In the stage-by-stage segmentation algorithm, a text is segmented in three successive steps: line segmentation, word segmentation, and character segmentation. Line segmentation relates to the process of separating text lines from a given scanned image of a document. Word segmentation, on the other hand, involves separating words from a given line of text. Finally, character recognition is the identification of characters from a given word. Such an algorithm is originally proposed (Chaudhuri & Pal, 1995) for use in the character recognition of printed Bangla script and not affected by noise. It shows the better results if the scanned image is not skewed. The segmented components comprise the complete set of Amharic, Geez, Tigrigna, and Qubee text line segmentation in this work. All segmented paragraphs into text line image segmentation are sent to the normalization module for scaling the text image to a manageable resize for the recognizer.

2.3.4 Normalization Phase

Isolated characters that are ready to pass through the extraction step of the function are obtained by the segmentation procedure, so the isolated characters are minimized to a specific size depending on the algorithms used. As it transforms the image into the form of a m*n matrix, the segmentation process is crucial. By decreasing the size and removing the unnecessary information from the image without losing any influential information taken from the image, these matrices are then usually normalized (Trier ØD, Jain AK, & Taxt T, 1996).

2.3.5 Feature Extraction Phase

Extraction follows the segmentation phase of OCR where the individual characters are considered and extracted for features. Image features are unique characteristics that can
represent a specific image. Image features are meaningful, detectable parts of the image. Meaningful means the features are associated with interesting elements via the image formation process. Feature extraction is the method of extracting the appropriate characteristics from objects or alphabets to create vectors of features. In order to associate the input unit with the objective output unit, these feature vectors are then used by classifiers. By looking at these characteristics, it becomes easy for the classifier to distinguish between different classes as it becomes reasonably easy to define (Pradeep J, Srinivasan E, & Himavathi S, 2011).

There are two main characteristic classes: statistical characteristics and structural characteristics. In a character matrix, statistical characteristics such as zoning, moments, crossings, Fourier transformations, and projection histograms are obtained from the statistical distribution of each point (Rehman A. & Saba T., 2014). As global features, statistical characteristics are also noteworthy as they are normally averaged and extracted in sub-images such as meshes. Statistical characteristics are initially given to identify machine-printed characters. Structural or topological characteristics, on the other hand, involve the geometry of the character set to be considered. Some of these features are convexities and concavities in the characters, number of holes in the characters, number of endpoints, etc. But nowadays we used modern machine learning approaches used for statistical feature extraction we mostly rely on deep learning models like using a of convolutional Neural network (CNN), Recurrent Neural Networks (RNN), and LSTM (Long Short-term memory) network together with CTC layers.

2.3.6. Classification Phase

Classification is the method by which a given image unit is categorized into one of the predefined categories based on a different image feature analysis. Since, once the image feature is extracted, the next step in multi-script recognition is to group each image unit based on its feature magnitude to its proper mark. OCR systems broadly use pattern recognition methodology, which assigns a predefined class to each example. Classification is the method of distributing inputs to their comparative class with regard to detected
information in order to produce collected with homogeneous characteristics while separating different inputs into a different class. The OCR classification techniques can be classified as Template Matching, Statistical Techniques, Neural Networks, Kernel Methods, and Classifier Combination, according to the discussed (Dongre VJ. & Mankar VH., 2011).

**a. Template matching**

In light of matching the stored models with the word or character to be interpreted, this is the least complex approach for character recognition. The matching operation defines the degree of similarity between two vectors by gathering forms, pixels, and so on. With a regular arrangement of stored templates, a gray-level or binary input character is contrasted. This strategy's recognition rate is highly vulnerable to noise and input disfigurement.

**b. Statistical Techniques**

The statistical decision suggestion deals with statistical decision capabilities and an arrangement of optimality parameters that can amplify the probability of the observed pattern for a given model of a particular class. According to (Dongre VJ. & Mankar VH., 2011), Nearest Neighbor (NN), Likelihood or Bayes classifier, Clustering Analysis, Hidden Markov Modelling (HMM), Fuzzy Set Reasoning, and Quadratic classifier are the key statistical methods conducted in the OCR area.

**c. Neural Networks**

Character classification issue is identified with heuristic rationale as people can perceive characters and records by their learning and experience. Thus, in discussed the work (Dongre VJ. & Mankar VH., 2011), the neural networks which are pretty much heuristic in nature are greatly appropriate for this type of issue. A neural network is an ascertaining architecture that includes the enormously parallel interconnection of flexible node
processors. The output from one node is reinforcing to the next one in the network and an official choice relies on the complicated collaboration of all nodes. As a result of its similar character, it can apply calculations at a rate higher contrasted with the traditional strategies. Feed-forward neural networks and feedback neural networks can be thought of as categorizations of neural network architectures. Feed-forward neural networks types of the organization referred to a bottom-up or top-down and feedback (Recurrent) neural networks are very powerful and can get extremely complicated. Feedback neural network architectures are also referred to as interactive or recurrent and it requires appropriate learning algorithms.

Therefore, the analysis we proposed uses feature extraction from input images of the Convolution Neural Network (CNN) and Bidirectional LSTM Neural Networks along with CTC methods. A modern version of Recurrent Neural Networks is LSTM Networks (RNN). Traditional RNNs suffer from the issue of gradients disappearing and exploding, which means that the gradient becomes either too small (vanishing) or too high (exploding) during the training resulting process. The study proposed the processing of feature extraction and contextual information in both forward and backward directions, the bidirectional LSTM (BLSTM) network and the addition of a layer in an LSTM network that performs a forward-backward algorithm called Connectionist Temporal Classification (CTC), output and allows the use of LSTM networks as sequence learning machines, i.e., the input sequence does not need to be segmented. To use the CNN and LSTM recognizer model for the historical multiple text training data of the Ethiopic script.

d. Kernel Methods

Although supporting Vector Machines is the most imperative kernel strategy, the kernel approach is often used by techniques such as Kernel Fisher Discriminant Analysis (KFDA) and Kernel Principal Component Analysis (KPCA). One of the most widely used and most efficient supervised learning techniques for binary or multi-class categorization is support vector machines (SVM). In classification techniques, the work discussed (Verma R. & Ali DJ., 2015), The data set is first partitioned into training and testing sets by convention.
SVM's aim is to produce a model that forecasts the test set's performance. The enhancement rule is the width of the edge between the groups, i.e., the unfilled zone around the decision boundary characterized by the interval to the nearest training example (Jain AK, 2000).

### e. Combination Classifier

Different classification strategies have their own particular advantages and shortcomings. Thus, ordinarily, various classifiers are consolidated together to solve a given classification problem (Matei O, Pop PC, & Vălean H, 2013) by utilizing neural networks and k-Nearest Neighbor, proposed Optical character recognition in real environments such as electricity meters and gas meters.

#### 2.3.7 Post-Processing Phase

There are different methods that can be used to enhance the accuracy of OCR results after the character has been classified, and a dictionary can also help to improve OCR results. The use of a dictionary to amend the minor errors of the OCR frameworks is the least complicated method for consolidating context details. The basic idea is to spell check the OCR yield and give a few different options in the dictionary for the recognizer's yields.

This study discusses a method to developing CRNN text image recognition and classification model we have considered the CNN for feature extraction, LSTM for classification for predicting sequential output per time-step task, and CTC for calculates loss value and decodes into the final text free segmented and without preprocessing phase. Since we discuss below review deep learning technique with related the study.

#### 2.4 Deep Learning Networks

Deep learning is one of the machine learning (ML) approaches that have a sound reputation in solving a huge number of classification problems that include character recognition, speech recognition, machine translation, writer identification, explained by (Rehman A. &
Razzak M., 2018) named entity recognition, protein sequences classification, etc. Deep learning is a powerful feature extraction method applied to extract the feature of printed characters. Deep learning provides the task-specific method, which inherits features from machine learning methods based on learning data representation. Feature extraction is the process of retrieving meaningful information from an image that is used for the classification of images to different categories. Feature extraction of segmented characters are the processes to extract different features. Some of the well-known deep learning models CNN are AlexNet, GoogLeNet, Visual Geometry Group (VGG), and for RNN are Bi-directional Long Short-Term Memory (BLSTM), MDLSTM, Hierarchical LSTM, and Extreme Learning Model (ELM).

2.4.1 Convolutional Neural Network (CNN)

As one of the most common deep neural network architectures, which comes in various variations, the Convolution Neural Network (CNN) was introduced in 1980. There are many features, such as convolution operation, parameter sharing characteristics and shift-invariant, making them a typical computer vision deep learning model. The Convolution Neural Network (ConvNets or CNNs) is one of the key groups for image recognition, image classification, in neural networks. The neural convolution network is a type of artificial neural feed-forward network in which the animal visual cortex organization influences the pattern of communication between its neurons. Simple ConvNet is a set of layers, and each layer of a convolution neural network converts one volume of activations through a distinct function to another. As discussed by (Li & W. Chen, 2017) Convolutional Layer, Pooling Layer, and Fully-Connected Layer are three key layer types for building CNN architectures. The dense layer is activated by soft-max to classify an entity with probabilistic values between 0 and 1. CNN processes an input image and classifies the objects on the basis of the values given.
According to discussed by (Yan, Zhang, & Vignesh, 2019), Figure 2 explains the structure of the convolutional neural network (CNN). A neural convolution network stands for a hierarchical model that inputs raw information such as audio and images. Using a series of actions such as convolution, pooling, nonlinear mapping of activation functions, high level semantics are abstracted or extracted layer by layer.

![Figure 2. The CNN architecture](image)

**Convolutional layer:** As the main module of the convolutional neural network, the convolutional layer is the core operation object of the convolutional neural network (Jia, Jeff, Karayev, & Darrell, 2014). The network uses the convolution operations mainly to extract features from the images, which is also the source of the convolutional neural network. The convolution operation uses a matrix that becomes the convolution kernel from top to bottom, slides on the images from left to right, multiplies the weight parameters in the convolution kernel with the elements at the corresponding positions covered by the image, and eventually sums up the pixel values that need to be output. As follows, the equation is shown (Verma R. & Ali DJ., 2015)).

$$x_j^l = f \left( \sum_{i \in M_j} x_{i-1}^{l-1} \ast w_{ij}^l + b_j^l \right)$$  \hspace{1cm} (2.1)

where $f(\cdot)$ represents the activation function of the convolution layer, which increases the nonlinearity of the algorithm and improve the network’s representation capability. There is $x_j^l$ is the $j$ feature map of the $i$-th convolution layer; $b$ is the bias parameter. $w$ is the parameter weight in the convolution kernel; $M_j$ is the currently selected feature map. Figure 3. showing the operation.
The activation functions the activation functions are necessary components of the network, the nonlinear representation potential of the entire network increases with the implementation of it. The function of activation includes sigmoid function formulated by (Han & Moraga, 1995), tanh function, and rectified linear activation (ReLU) (Browne, 2011), Sigmoid function is an activation function.

\[
f(x) = \frac{1}{1 + e^{-x}}
\]

2.2

\[
\tan(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]

2.3

Tanh’s function based on sigmoid function is called the hyperbolic tangent function. The range of these function values is fallen in (-1, 1). The mean of the outputs is 0. The convergent speed is faster than the sigmoid function.

Rectified linear units (ReLU) function is one of the key activation functions of the existing CNN network discussed by (Nair & Hinton, 2010). His term is \( f(x) = \text{max} \ (0, x) \). A comparison is made between the input value and zero, and the larger value is used as the ReLU function output. If the input value is less than zero, the output value is zero; if the input value is greater than zero, then the output value is equal to the input value. The ReLU function will speed up the training phase of deep neural networks compared to the classical activation function since the derivative of the ReLU function is 1.0 when the input is greater than zero.

Pooling layer. Pooling is the average or maximum value of the outcomes of convolution, (Scherer, Müller, & Behnke, 2010). In order to minimize the number of parameters, it is
used to decrease the dimensional size of the representation, minimizing the approximate cost of the network. Two approaches are commonly used for pooling operations: average pooling and max pooling. There are separate operations for pooling and convolution operations. The pooling operation does not require the network parameters that the algorithm needs to recognize. The only thing you need to do is define the type of pooling operation, the size of the filter scan windows, the phase values, and the size of the filter scan windows. Both average pooling and max-pooling will complete the down-sampling operation, while a linear function is the former, while a nonlinear function is the latter, and max pooling is typically better.

The pooling layer is used to decrease the input image's spatial volume after convolution. It is used between the layers of two convolutions. If we implement FC without applying pooling or max pooling after the Conv layer, then it will be costly in terms of computation and we don't want it. The only way to reduce the spatial volume of the input image is to use max-pooling. In the above example, we have applied max-pooling in single depth slice with the stride of 2, shown in the figure 4 example max-pooling the 4 x 4-dimension input is reduced to 2 x 2 dimension

![Figure 4. Max-pooling is a convolution neural network](image)

**Fully connected layer.** In a convolutional neural network, the fully connected layers are used as a classifier explained by (Andrearczyk & Whelan, 2016). The convolutional layer, pooling layer, and activation function layer are used to transform the primitive data to the feature space of the hidden layer, while the fully connected layer is to map the learned features to the sample label space.
2.4.2 Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is one of the most important types of neural networks that is indicated by mapping all previous inputs to each output, thus creating a memory, to deal with sequential data from their repeated data sequence connections. In both forward and backward directions, bidirectional RNN allows the network to learn the meaning. As explained in detail by (S. Hochreiter Y. B., 2001) RNNs use gradient-based methods for learning, thus when the number of hidden layers increases, the error which is propagated back through the hidden layer reduces. This phenomenon is known as the vanishing gradient problem. While RNNs have several sequence labeling applications, they suffer from two important issues: first, in the present time, the RNN can access the past context. Secondly, the backpropagation error blows up or disappears by increasing time steps. The weights will oscillate in case one, and the rate of updating weights is too slow in the second case (Graves & Schmidhuber, 2009). In order to generate the production of the current time, the first problem is solved by using bidirectional neural networks that find past and future information. Two RNNs actually pass through the series in both forward and backward directions. The results of both forward and backward computations aggregate into the final network’s output. The second problem is solved by utilizing long-short term memory (LSTM). LSTM is an important memory block for accessing data in long time steps and is similar to the memory chips in computers.

**Recurrent layers:** to predict the feature sequence, these layers use bidirectional RNN (BLSTM). Learn each feature vector in the sequence and output the predicted label (true values) distribution. The German researchers, introduced by (S. Hochreiter & J. Schmidhuber, 1997), in a paper published in 1997, LSTM is a special form of Recurrent Neural Network (RNN) capable of learning long-term dependencies that are helpful for some forms of prediction that enable the network to maintain data over longer periods, a challenge that conventional RNNs are struggling with. According to the discussion given by (Graves & Schmidhuber, 2009) LSTM has three primary multiplication gates: input, output, and forget. They are similar to write, read and reset for a cell, respectively. LSTM
uses the multiplicative gates to store and access data over a long time, hence it eliminates the problem of vanishing gradient. The main part of the LSTM is the cell state, $c_t$, that conveys the main information through time. Using the gates, LSTM writes and clears the data in the cell state. Each gate consists of a sigmoid function and a multiplication operation that is element-wise. The Sigmoid function output is between zero and one. Zero implies that nothing should be passed and one implies that the data should be fully passed through, showed in the architecture of the LSTM block diagram in figure 5.

![LSTM Block Diagram](image)

Figure 5. An architecture of LSTM cell Block diagram adopted from (Safarzadeh & Jafarzadeh, 2020)

From figure 5, brief definitions of the gate’s algorithm proposed by (Safarzadeh & Jafarzadeh, 2020) an LSTM has three gates: Forget (yellow), Input (blue), and Output (red). Sigmoid functions (green boxes) play the main rules in the equations of the gates.

**Forget gate:** The forgotten gate chooses to delete data that is no longer required for the task to be completed. This move is vital to optimizing the network efficiency. The gate uses the output of the previous time step ($h_{t-1}$) and the input of the current time ($x_t$). The output is between 0 and 1, 0 means the data should be cleared and 1 means the data will be saved:

$$f_t = \sigma(W_f \cdot x_t + W_f \cdot h_{t-1} + b_f)$$  \hspace{1cm} 2.4
the values $f_t$ is between 0 and 1. An element wise multiplication of $f_t$ by $c_{t-1}$ determines which of the elements of the $h_{t-1}$ must be saved and which of the elements must be cleared. 

**Input gate:** responsible for adding information to the cells. This part has two phases, first, $i_t$ decides which values are updated and $g_t$ generates new values for the cell state:

\[ i_t = \sigma(W_i * x_t + W_i * h_{t-1} + b_i) \]  
\[ g_t = \tanh(W_g * x_t + W_g * h_{t-1} + b_g) \]

Hence, the new value of the cell state is the combination of the forget gate and the input gate. In fact, the forget gate clears the old data and saves the important data and the input gate writes on the cell state:

\[ c_t = f_t \odot c_{t-1} + i_t \odot g_t \]

**Output gate:** selects and outputs necessary information. This gate controls which parts of the current cell state $c_t$ should be read and sent to the output.

\[ o_t = \sigma(w_o * x_t + W_o * h_{t-1} + b_o) \]
\[ y_t = h_t = o_t \odot \tanh(c_t) \]

### 2.4.3 Connectionist Temporal Classification Layer

In Connectionist Temporal Classification (CTC) introduced by (Alex Graves S. F., 2006) The Connectionist Temporal Classification principle maximizes the likelihood of the label sequence given in the input sequence with a new forward-backward algorithm for input and label sequence alignment. This allows the network to eliminate the need for pre-segmented data and systems for post-processing. As the output layer with the soft-max activation function, a Connectionist Temporal Classification (CTC) layer is used and the number of units is equal to the number of labels plus one. This additional unit is for the likelihood of the blank's occurrence. This helps to establish the swapping of labels that represent the beginning and end of a series and also to distinguish the same labels that occur in the network consecutively. The use of CTC makes it possible to operate on unsegmented data as it can begin labeling from any point in the input sequence, regardless of the alignment. Only when the alignment between input and output sequences is unclear is CTC used as the output layer. The CTC output is a sequence of probabilities, thus removing the need for post-processing schemes as required by RNNs for the complete label
sequence. The CTC loss function calculates the likelihood of an alignment per each time-step using a dynamic programming algorithm, given the performance of an RNN a sequence of probabilities.

Now it is possible to identify 1D-LSTM networks or 2D-LSTM networks as 1D-LSTM networks depending on how the data is viewed in the input layer for LSTM networks, the content explains by (Breuel, UI-Hasan, AI-Azawi, & Shafait, 2013), the input is in the form of a single-dimensional sequence for the 1D variant and the 2D case, the bidirectional mode implies scanning the input in four directions, namely right-to-left, left-to-right, top-to-bottom and bottom-to-top. We find that 1D-LSTM networks perform better than their 2D siblings for printed OCR tasks. The input text-line image is then scanned by a 1-pixel-width fixed-height window to transform the 2D-image into a one-dimensional sequence using 1D-LSTM for OCR tasks. This slice of pixel width is called a frame.

The text line is translated into a one-dimensional sequence and each frame is fed to the LSTM network, where the secret and output layer moves forward. The forward-backward algorithm (CTC) then aligns the activation of the output with ground-truth labels and the error is then back-propagated (backward pass). The LSTM network learns to classify each frame into a target class (including space and rejection classes) during this process. The CTC output sequence is one-dimensional data and text because the output must be collapsed vertically before being added to CTC since it has horizontal writing (Alex Graves N. J., 2013).

**Padding:** To preserve the same dimensionality, it is possible to apply padding to border the input with zeros (Arden, 2020). Padding is a zero-border value that is positioned outside the matrix layers of the image, mitigating the effects of decreasing the output by zero-padding the input. It is possible to achieve the effect of canceling dimensional discounts and retaining the dimension at the output. Therefore, in order to prevent too much output from being reduced, padding may be needed. We also used the same padding used in our research. In this case, to the output image, we apply 'p' padding layers so that it has the same dimensioning as the input image.
2.5 Network Optimization

Given any set of hyperparameters, the loss function is a metric that quantifies the quality of our model. The aim of optimization is to figure out that the loss function is minimized. This is not sadly, a trivial task. The problem is not convex because of the nonlinearities within the neural network, and we have millions of parameters to set, facing a difficult computational issue to solve. Not all neural network hyperparameters can be automatically learned from the gradient-based algorithm such as the number of layers, the number of hidden units (neurons), the activation function, and so on.

Adaptive Methods

Machine learning practitioners have long known that one of the most difficult hyperparameters to identify is the learning rate since it greatly affects the efficiency of the model. The momentum-based algorithm attempts, but at the cost of adding another hyperparameter, to mitigate this problem. Much work has been done to develop approaches that can change the leaning adaptively, ideally per parameter, and with no new hyperparameters being introduced. This purposed, unfortunately, has yet been accomplished. To address the first problem, adaptive methods were developed and still require hyperparameters.

One of the most recent adaptive learning rates approaches proposed is Adam (Kingma & L.J.Ba, may 2015), or ADAptive Moment estimation. It looks like the RMSProp update, but instead of the likely noise gradient, the first order momentum is used. The algorithm (simplified) looks like

\[ v \leftarrow \alpha_v + (1 - \alpha_v) \frac{\partial \tau}{\partial \theta} \]  \hspace{1cm} 2.10

\[ m \leftarrow \alpha_m m + (1 - \alpha_m) \left( \frac{\partial \tau}{\partial \theta} \right)^2 \]  \hspace{1cm} 2.11

\[ \theta \leftarrow \theta - \frac{v}{\sqrt{m + \epsilon}} \]  \hspace{1cm} 2.12
where $\alpha_v \in [0, 1]$ and $\alpha_m \in [0, 1)$ are hyperparameters, and $\varepsilon = 10^{-8}$, $\alpha_v = 0.9$, and $\alpha_m = 0.99$ are the recommended values (Kingma & L.J.Ba, May 2015). The Adam typically works marginally better than RMSProp and is the recommended algorithm to use when the hyperparameters are searching, since Adam is faster than Nesterov. The most popular approach is to look for the best model configuration with Adam and train the best model configuration (tuning the hyperparameters) with SGD plus Nesterov.

Finally, in the paper we proposed a combination of CNN-BLSTM architecture with CTC for optimized multi-script Ethiopic text-line image recognition. This architecture used primarily has Convolutional layers and bidirectional LSTM layers using the input and output layers.

### 2.6 Related Works

Recognition of character image has been studied and solved for multiple scripts. Researchers have made lots of work to address the problems in most Latin and non-Latin scripts and even most of the scripts, now, have commercially off-the-shelf OCR applications. Until recent times, the OCR for Ethiopic script remained relatively unexplored. Different researchers use and evaluate various algorithms, such as statistical, template matching, structural, support vector machine, and neural network to enhance the performance of the recognition system.

Since it is important to know the current state-of-the-art of the OCR is present it is critical to conduct intensive review related works of previous studies both traditional machine learning and deep learning technique for trends high level language and OCR for low resource language would be conducted. The study would be proposed and conducted on the implementation of an optical character recognition for multi-script Ethiopic text recognition using deep learning. They have different researches which, are done Ethiopian single script recognition system, but not working multi-script Ethiopic text recognition are the following reviews. Also, there had been many types of research that compared different
machine learning techniques including ANN, HMM, CNN, SVM, RNN, and a combination of CNN and LSTM with CTC approaches. However, methods lack an explicit representation of the entire sequence. Applying these algorithms on OCR tasks have their benefits and limitations as discussed below.

The first work on the Amharic OCR used different machine learning techniques a character written with the common font and size type (Worku Alemu, 1997). Then, other attempts were made, such as machine-printed (Million, "A Generalized Approach to Optical Character Recognition of Amharic"., 2000), Amharic braille document image recognition (Yaregal & Hassen, 2017), OCR for Geez script written on vellum (Shiferaw, 2017), Amharic document image recognition and retrieval (Million, 2008), Amharic character recognition printed real-life documents using ANN (Abay, 2010), and bilingual scripts recognition for mixed Amharic and English printed document using Support Vector Machine algorithm (Sertse, 2011), all of these researchers, however, the applied technique of traditional machine learning. Also, character recognition was given at the character level. This is time consuming process because used segmentation-based approach, each image of each character depends directly on the design of the segmentation algorithm text images (Israr Uddin, 2017).

Since the success of deep learning, there have also had other attempts for the different characters using CNNs (Mesay, Lars, Abiot, & Hadi, 2019), (Belay, Habtegebrial, Belay, & Stricker, 2019), (Jieru Mei, 2016) almost all the researchers to segment text images into the level of character, which also directly affects the performance of OCR output. Whereas, along with CTC (Connectionist Temporal Classification) for Amharic text image recognition, the proposed Long Short-Term Memory (LSTM) networks (Birhanu, Tewodros, Marcus, Gebeyehu, & Didier, 2019) and (Addis, Liu, & Ta, 2018). In these works, attempts at Amharic OCR did not produce effects on large datasets or recognize all Ethiopian characters used in the various writing papers. Off-the-shelf commercial and open-source OCR implementations provide multiple languages, including the capability of generating ground truth from current printed text train models. (Christophe, Jean-Christophe, & Jean-Marc, 2017). Many segmentation-based and segmentation-free OCR
techniques have been studied on the basis of deep neural networks. Many Latin and non-Latin scripts, ranging from handwritten historical documents to up-to-date machine-printed texts, have also made tremendous development.

Bi-directional recurrent neural networks with Long–Short-Term Memory (LSTM) architecture for online handwriting recognition proposed by (Marcus, Alex, Horst, & Jurgen, 2007), Convolutional Recurrent Neural Networks (CRNN) for Japanese text offline handwriting recognition (Nam-Tuan, Cuong-Tuan, & Nakagawa, 2017), Urdu Nastaleeq scripts recognition using bidirectional LSTM (A. Ul-Hasan, 2013), a hybrid Convolutional Long-Term Memory Network(CLSTM) for text image recognition (Thomas M. B., 2018), Multidimensional LSTM (MDLSTM) for Chinese handwriting recognition (Wu, Yin, Chen, & Liu, 2017), combined connectionist temporal classification (CTC) with Bidirectional LSTM (BLSTM) For unconstrained identification of handwriting online (Alex, Santiago, & Marcus, 2007), a combination of CNN and RNN with CTC offline Persian text handwriting recognition (Safarzadeh & Jafarzadeh, 2020) and an end-to-end learning using combined with convolutional neural network (CNN), Bidirectional LSTM (BLSTM), and connectionist temporal classification (CTC) in a recognition image for Amharic text-line images (Birhanu, Tewodros, Million, Marcus, & Gebeyehu, 2020) have been studied.

In an OCR system, in summary, we can verify that all of the researchers use single Ethiopic character image recognition but multiple scripts for Ethiopic real-life documents are not built on a multi-Ethiopic recognition model. In this paper, therefore, we proposed a deep learning trainable neural network that includes a CNN, BLSTM with CTC techniques for Ethiopic multi-script document text-line image recognition. Finally, the study would be proposed a model for multiple script text-line writing image recognition that achieves state-of-the-art results.
CHAPTER THREE: RESEARCH DESIGN AND METHODOLOGY

In this chapter, methodologies, techniques, and algorithms that were used and the general study method methodology has been discussed. On each step and process, appropriate techniques were discussed with sufficient examples and justification for their use in the study. The proposed multi-script Ethiopic character recognition system involved training of deep learning using text line image segmented from the mixed writing document. The trained network was used to recognize character inputs into a defined set of output classes from Amharic, Geez, Tigrigna and Qubee mixed text-line image.

3.1 Proposed Text Image Recognition Methods

The study followed an experimental research design, which included dataset preparation, implementation, and performance evaluation. The research worked concerning the development uses of a combination of the CNN and LSTM with CTC for multi-script image recognition from Amharic, Geez, Tigrigna, and Qubee documents image. There were two major work components in the system; the first one was pre-processing of collected data and segmentation of character and the second one was extracting features and predicting sequential labels. Digitalization, noise removal, binarization, text-line segmentation, normalization, and others belonged to preprocessing step generated artificially using the OCRopus tools in the proposed system from the given scanned image of printed multiple writing text document. For the creation of the dataset, we used the original grayscale images of all pages together with the corresponding texts. To extract text-lines, we first binarized the grayscale images and then applied layout analysis and segmentation processes. The figural representation of the proposed workflow multi-script model was shown in Figure 6. In the next section, each task and subtasks were elaborated and discussed in detail.
3.2 Digitization

Digitalization refers to set of tasks to convert the hard text document or soft texts into a format of document image readable by the developed for the research purpose (Bhatia, 2014). For the printed text image recognition system, the first task after the collected the printed document was converted into electronic form. The data was collected from Ethiopian documents written using multiple scripts historical documents. Datasets from mixed
writing documents and prepared for further preprocessing step. All text images, for training and testing the model, were collected from Ethiopian orthodox Bible books for mixed writing for Amharic, Geez, Tigrigna and Qubee scripts.

### 3.3 Preprocessing Technique

The preprocessing phase was necessary for efficient recovery of the information from scanning synthetic images. The scanned image preprocessing algorithms rely on the quality of the document, the resolution of the scanning image, the amount of image bias, the image format and layout, and text, etc., (Rehman A. & Saba T., 2014). Since the text images were gathered from the Bible book of Ethiopian script writing, there were some noises that appeared on the image that degraded the quality of the image as well as reduced the model's output. This research performs image preprocessing tasks to create an improved quality of the input image dataset to minimize the impact of noise.

In this stage, we made some preparations on the generated synthetic image data using Ocropy tools. Since OCRopus was a collection of programs for document analysis (ocropy, GitHub. [Online]., 2020). To apply it to documents used to work image preprocessing to train models. When running the OCRopus pipeline convert our document into a collection of .png files. We used the convert function that came standard with many Ubuntu OS’s and used the ocropus-preproc command on every .png file. This did all of the preprocessing steps to PNG files (whitening, alignment, binarization, etc.) could be accomplished with a consistent numbering system of its .png files and the use of the character for the incrementing digits. We needed to run separate commands, binarization, and page layout analysis, using the software used to identify text pages. OCRopus’ default parameters and settings support 300 dpi black-on-white binary images.

For different resolutions, the synthetic text line image recognizer was very vigorous, but the layout analysis was fairly resolution dependent. To make a directory dataset containing training data suitable for training OCRopus with synthetic multi-script identification from mixed Amharic, Geez, Tigrigna, and Qubee document image then, the synthetically
generated text-line images taken from the dataset. The study used to be proposed in preprocessing steps were binarization, text line segmentation, and size normalization.

### 3.3.1 Binarization

Image binarization is the process by which a color or grayscale image is transformed into a binary image, that is, an image with only two colors (typically, black and white). The first preprocessing stage of the study was binarization in the Ocropus pipeline tool; the conversion of the source image from grayscale to black and white. A type of adaptive thresholding was used by Ocropus, where the cutoff between light and dark could differ throughout the picture. When dealing with scans from the Bible book, where there can be variance in light level across the page, this is significant. Skew estimation, which tried to rotate the picture by small amounts so that the text was really horizontal, was also lumped into this step then, in multi-script recognition from mixed Amharic, Geez, Tigrigna, and Qubee document image, the researcher proposed to deal with the binarized image.

### 3.3.2 Text Image Segmentation

Segmentation refers to a sequence of tasks or processes in which a document image is chosen and segmented for further processing into the required image representation unit. In the proposed research recognition from a document, the image contained mixed writing Amharic, Geez, Tigrigna, and Qubee scripts the unit of image representation was text image. Each paragraph segmentation into line text of the binary image was extracted using the horizontal projection profile method and directly complete text-line were segmented from the text image.

The dataset is composed of four Ethiopic scripts. To measure the number of all black pixels on each row and create a corresponding histogram, the horizontal projection method was used. After the paragraph segmentation, the next step was to extract individual text lines for feature extraction. The valley of a horizontal projection determined by a row wise number of black pixels was used to segment the image into text lines. All text line images
input must first be normalized to a specific height in order to use a combined CNN and BLSTM network with CTC, since the normalization of the text line was the first step for extracting features. For the input image and the CRNN networks to select and implement the process of focus on normalization that was available as open source in OCRopus (ocropy, GitHub. [Online]., 2020). Would be used a line generation tools available in OCRopus tools for this purpose. This tool could generate any number of text-line images with given text files and font-files.

### 3.3.3 Text-line Normalization

When using this method, the input image height and width of the text-line was normalized to the specific height. Therefore, skew angels were corrected and measured. The orientation of smoothed image vertical path and vertical center of the output image as the writing point and resized all image. At the text-line stage, the method was able to distinguish multiple scripts when we are one and two or more scripts were present in the same presented text-line. We used the OCRopus system's utility (ocropus-linegen) for the actual text-line generation to produce synthetic text-line images.

### 3.4 Feature Extraction and Classification

In the CNN-LSTM architecture the use of the Convolutional Neural networks (CNN) layers for feature extraction on input data combined with LSTM’s to aid sequence prediction was involved. CNN-RNN for visual time series prediction problems and the application of textual description generation from image sequences were developed (Ghosh R., 2019). CNN and LSTMs were a class of model that were both spatially and temporally deep and had the versatility to adapt to a variety of vision tasks, including sequential inputs and outputs, the suggested model, the sequence recognition problem, the most appropriate model, it would be easier to combine CNNs and RNNs with CTC algorithms due to reduced computational costs during training.
Extraction of features is the method of extracting meaningful text image information used for classifying images into various categories (Kumar & Pradeep, 2014). To extract the descriptive feature from the text-line image, this study employed using CRNN. After the model to create training and validation dataset for text line image recognition model load from preprocessing and had been created the model architecture and trained it with the preprocessed data. For the proposed feature extraction and classification network model workflow consisted of a combination of three layers. To extraction of features from the text-line image, the first layer was the convolution neural network. Long Short-Term Memory of the second layer to predict sequential performance per time-step. The Connectionist Temporal Classification (CTC) was the third layer LSTM output matrix that served as the classifier alignment between the input and output sequence in the final text expected.

Convolutional Neural Network (CNN) to extract the feature from the text line image, we had used the CNN model. It has an enhancement of ANN which was specifically designed for two-dimensional data. CNN is an efficient detection algorithm that is widely used in pattern detection and image processing. CNN aims to use spatial information between the pixels of an image (Zhang, Pezeshki, & Bengio, 2016). CNN is used directly from the 2D image to automatically learn a hierarchy of characteristics that could use for both function identification and classification purposes. This was achieved by successively transforming the input image into a hierarchy of function maps with learned filters (Browne, 2011). In general, the extraction of features was categorized into low-level features that addressed the line, points, edge, and corners and high-level features that were designed on top of high-level features to detect objects and shape in the picture.

The CNN model introduced by (Durjoy Sen Maitra, 2015), they had an input layer, output layer, and hidden layers. By grouping functionally distinct layers, the hidden layer of the CNN model was created. The hidden layers typically consist of convolution layers, ReLU layers, pooling layers, and full connected layers, all of which were technically distinct, whereas the convolution layer was used to extract the feature from the input data. The pooling layer was also used to sample the feature map obtained by using the convolution
layer by taking one superior invariant feature without affecting the feature map, and the activation function was also used to introduce nonlinearity between the feature map that was measured in the convolution layer using linear operation.

Recurrent neural networks (RNNs) are connectionist models containing a self-connected hidden layer. The recurrent connection allowed information of previous inputs to remain in the network’s internal states; then, it made use of past contextual information. RNN’s signals traveled in both directions, creating a looped network. It considered the current input with the previously received inputs for generating the output of a layer and memorized past data due to its internal memory. The typical RNNs, however, suffer from the problem of the gradient vanishing and exploding problem. A specific architecture of recurrent neural networks was the LSTM. LSTM had been proposed by (Hochreiter, 1998) to long-range dependencies in sequential data using a special memory cell with three multiplicative gates to control and access information. The bidirectional LSTM network consisted of two LSTM networks. The first handled the input sequence in the forward direction and the other in the backward direction. The BLSTM architecture proposed by (S. Hochreiter & J. Schmidhuber, 1997), it helps us to take past and future contexts into consideration. To prevent the vanishing and exploding gradient problems, LSTM was the gateway. Due to its ability to access long-range contexts, this architecture was chosen to learn sequence alignment and function without the need for segmented data. Extract from the previous stage in the training process and enhancement the efficiency of OCR. It worked, the network took the BLSTMs are based on a feature extractor and the sequence extracted from the previous part to predict a label distribution for each feature.

3.5 Feature Learning sequential for Training Phase

The following are the description of each layer in each phase to learn the features extracted and learning sequential that each label of text-line images.
**Input Layer**: This layer takes the dimensional binary text-line image as input. Each feature was different scripts of the all-input text-line images are normalized to a fixed size of 32 by 256 pixels and label.

**Convolutional Neural Network (CNN) Layer**: The first layer to extract features from the text-line images and pre-trained text. It was feedforward neural network signals traveled in one direction from input to output. There were no feedback loops; the network considers only the current input and cannot memorize previous inputs. Convolution preserves the relationship between pixels by learning image features using small squares of input data. The feature sequence was extracted from the text line image by the convolutional feature extractor, which was the input of the recurrent layer. The weights of the CNN model in the convolutional feature extractor from preparing text line image dataset. CNN which was a complex type of linear operation consists of mathematical operations such as convolution. Pixel values were stored in a two-dimensional (2D) grid in digital images, i.e., an array of numbers and the small grid of parameters called kernel an optimizable feature extractor was added to each image location.

We used seven convolution layers, which have different kernel size types, raising the number of layer-by-layer feature maps. The aim of ReLU was to implement non-linearity operation in ConvNet (Ahmed Elsawy, 2017). ReLU (Rectified Linear Unit) was the most widely used activation function \( f(x) = \max(0, x) \), when the explain \( f(x) \) was 0, if the value of \( x \) less than 0 and \( f(x) \) is greater than 0 if the value of \( x \) is greater than or equals to 0. It effectively removes negative values from an activation map by setting them to zero and it was often used for hidden layers. Finally, the activation function was used to control how the data flows from one layer to the next layers.

The model understands the input and output dimensions at the end of each convolution layer. We decided to take a text-line image to make it capable of identifying the output dimensions. The hyperparameter would control the size of output volume. We could apply a simple formula to calculate the output dimensions. The special size of output text image could be calculated as equation 3.1.
\[ D = \left( \frac{W - F + 2 \times P}{S} \right) + 1 \]

Now, \( W \) is the input volume size, \( F \) is the size of the filter/kernel, \( P \) is the number of padding applied, \( S \) is the number of strides and \( D \) is the output dimension.

Pooling layers in this section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down-sampling which reduces the dimensionality of each map but retains important information. Pooling layers are commonly inserted between successive convolutional layers. We wanted to follow convolutional layers with pooling layers to progressively reduce the spatial size (width and height) of the data representation. Pooling layers reduced the data representation progressively over the network and help control overfitting. The most common form of pooling layer usually applied was the max-pooling. The pooling layer operated independently on every depth slice of the input. A max-pooling layer was used to summarize image regions and outputs a downsized version of the previous layer. And also used batch normalization layers for at convolutional layers training process speed rate.

Then we used a reshape function to squeeze the output from the convolutional layer and made it compatible with the LSTM layer. The reshape function is used to give a new shape to an array without changing its data. Array to be reshaped and the new shape should be compatible with the original shape.

**Long Short-Term Memory (LSTM) layers:** LSTM was a special form of RNN that was able to learn long-term dependencies, it was introduced by (S. Hochreiter & J. Schmidhuber, 1997), to avoid long-term dependence problem, the main components of an LSTM network layer are a sequence input layer. A sequence input layer inputs the network with sequence or time series data. A layer of LSTM learns long-term dependencies between time stages of sequence information. It was useful to use future as well as past contextual data for LSTM tasks such as text-line image recognition. The basic LSTM can, however, only use past contextual data in one direction. Then we have used a bidirectional system, there were two LSTM sub-layers, namely forward and backward sub-layers. The two hidden layers were connected to a single output layer. At present, each layer contains 128
LSTM units. To recognize text in an image as bidirectional to present each sequence forwards and backward to two separate hidden states to capture past and future information, respectively. For forward pass the feed all input data for sequence into BLSTM and calculate all prediction output and backward pass calculate the error function for sequence used in the forward pass. The study proposed a CNN-LSTM model, the BLSTM is built on the upper of the convolutional feature extractor, as the recurrent layer to predict a level distribution for each feature of sequence extracted from the previous steps.

In the output layer of neural network models that predicted a polynomial distribution of probability, the soft-max layer function was used as the activation function. For multi-class classification problems where class membership was needed on more than two class labels, soft-max was used as the activation function. In order to get a probability distribution over the C possible characters in the script (including spaces, punctuation marks, numbers, or other special characters) plus a black label required by the CTC loss and decoder, the output of the LSTM layers at each stage was fed into a soft-max layer.

**Transcription layer:** the CTC loss sequence was used to convert a collection of label distributions acquired from the layer of the loop into the final label sequence. CTC was a particular loss function designed for the tasks of sequence labeling where it was difficult to segment the input sequence exactly matching a target input sequence to the segment. CTC performs alignment of a probability output sequence to the label sequence. According to proposed by (A. Graves, 2009), there is no need to segment the system outcome into the input sequence for training. The CTC was used in the training process of the BLSTM to estimate the error vector to back-propagate using unsegmented data at each time-step. In a neural network, a CTC acts as a transcription layer. We required to identify the text present in the image in text recognition issues. In the case of sequence generation, CTC layers are used to solve alignment problems, decode text from it and escape the blank character and predict the correct output. The CTC performance was a series of probabilities, thereby removing the need for post-processing schemes for the complete label sequence.
As explained in the work of (Nam-Tuan, Cuong-Tuan, & Nakagawa, 2017) we employed framework in CTC to be built on top of the recurrent layers, as the transcription layer. To denote the character set as $C = C \cup \{\text{blank}\}$, where $C$ is a fixed set of labels and ‘blank’ represents no label. For an input sequence $x = x_1, x_2, x_3, \ldots, x_T$ of length $T$, the conditional probability of a path $\pi$ through the frame of output labels over all the time steps is calculated by multiplying the probabilities of labels along this path:

$$P\left(\frac{\pi}{x}\right) = \prod_{t=1}^{T} P(\pi_t, t/x)$$  \hspace{1cm} (3.2)

where $\pi_t$ in the label of the path $\pi$ at time $t$.

Taking an example from (Nam-Tuan, Cuong-Tuan, & Nakagawa, 2017), a label series was obtained by a reduction method denoted as $B$ from a line, which first removes repeated labels and then removes blanks in the path, for example, $B(_\text{dd}_\text{o}_\text{gg}__) = B(_\text{do}_\text{g}__) = \text{dog}$. The total probability of all the paths is the probability of a mark sequence from an input sequence $x$, where $B$ reduces each path to that label sequence. It is shown as follows:

$$P(l/x) = \sum_{\pi: B(\pi) = l} P\left(\frac{\pi}{x}\right)$$  \hspace{1cm} (3.3)

To applying the CTC forward-backward algorithm formulated as equation (3.3), $P(l/x)$ was obtained efficiently. For decoding, we could obtain the best label by formulated equation (3.4)

$$l_{\text{max}} = B(\pi_{\text{max}}); \pi_{\text{max}}^{t} = \operatorname{arg\, max }_{k} (y_{k}^{t}), f o r \ t = 1, 2, \ldots, T$$  \hspace{1cm} (3.4)

This was obtained without explicitly segmenting the input sequence.

Soft-max was an activation function that produces the output between zero and one from the neural network of the study used. It divides each output, such that the total sum of the outputs was equal to one.

**Decoding** is an output of the soft-max layer is a sequence of time steps of $(c+1)$ class. A CTC decoding scheme was applied on the BLSTM soft-max outputs using the best path decoding algorithm.
**Dropout** is a form of technique regularization that randomly drops out secret and visible network units that is used to prevent a network from reducing overfitting training data, operating at each training phase by randomly dropping units (along with their connections) from the neural network.

**CTC loss:** it is very helpful in text recognition problems. It helps us to prevent annotating each time step and help us to get rid of the problem where a single character can span multiple time step which needed further processing if we do not use CTC. If you want to know more about CTC. A CTC loss function required four arguments to compute the loss, predicted outputs, ground truth labels, input sequence length to LSTM, and ground truth label length. To get this we need to create a custom loss function and then pass it to the model.

### 3.6 Feature Learning for Testing Phase

In this phase, we follow the same procedure at preprocessing steps as the training phase. In the testing phase, we have to preprocess the text image in the same manner as the training. The different feature extracting and learning is sequence alignment also done in at in the training phase by using the learning model constructed from the training. Input text line images that are split used for the training datasets and testing dataset.

In Summary, develop the convolutional neural network for extract features from multiple script text-line images and the LSTM to predict sequential output per time-step and CTC the transcriber calculates encoding and error loss. The feature extraction and classification were implemented in the training phase. The training phase uses a training dataset to form the feature extraction and sequential learning model. The testing phase, however, uses testing dataset passes directly text-line normalization to the trained model to finally predict text.

### 3.7 Evaluation Techniques
In this study, the performance of the designed model was evaluated based on the total number of text image which was correctly and incorrectly predicted by the model. To use the same architecture in all experiments and result evaluated the performance of the proposed model was calculated by the total number of characters inserted, substituted, and deleted in each sequence and then divided by the total number of ground truth characters. The formula proposed by Levenshtein distance (Li & Liu, 2007), the most commonly measured in Character Error Rate (CER).

\[
CER(\%) = \left( \frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{total number of character in test set}} \right) \times 100
\]

3.5

Where the number of all insertions, deletions, and substitutions were summed up over all the characters in the test set.

### 3.8 Implementation Tools

The experimental research set up to follow and step-by-step procedure in text image recognition such as image preprocessing, segmentation, feature extraction, and recognition. Octopus tool and Python3.6 programming language with the Tensor flow was also free machine learning library used for mathematical operation and Keras deep learning library with a TensorFlow backend on CPU used during creating a layer in CNN and BLSTM-CTC model. Text image recognition libraries designed for python such as NumPy, Pillow, matplotlib are used for processing the input text image for the networks. For doing all tasks we used Jupyter Notebook which is a web-based interactive computing notebook environment for edit and run python codes. To operate environment running on the Ubuntu operating system of Hp desktop, Intel(R) Core (TM) i7-8700U CPU @ 4.6GHZ processor, with 16GB DDR4-2666 SDRAM, under the experimentation.
CHAPTER FOUR: EXPERIMENTS AND RESULT DISCUSSION

In this chapter, the research talked about the experiments carried out to develop a text recognition system model for Ethiopic mixed writing documents. The implementation of preprocessing step, feature extraction, and recognition techniques have been presented in the previous chapter. The study developed a recognition model that could recognize printed synthetic Ethiopic multiple script documents using combine CNN and BLSTM networks with CTC algorithms. To train and test the model, the dataset was prepared from different sources multiple written in Geez, Amharic, Tigrigna, and Affan Oromo scripts. Training and testing datasets were preprocessed using the OCROPUS tool publicly available to convert the document to ground truth and text line image. Thus, on each OCR procedure appropriate techniques model was discussed with the adequate experiments and result in discussion.

4.1 Dataset Collection and Preparation

The research text image recognition using deep learning networks for the printed Ethiopic documents. The primary task was collection the required data and preparing it for further processing. The Ethiopic multiple writing real-documents text line recognition the benchmark dataset for Amharic, Geez, Tigrigna, and Qubee scripts. Recognition of Ethiopic scripts was very challenging due to the presence of the large number of classes and highly similar shapes of basic characters. Therefore, to prepare an Ethiopic script dataset organized from a type of printed documents collected dataset from mixed writing documents of the Bible books in Ethiopia Orthodox Churches.

Prepared an electronic copy of the multiple writing documents in a printed text file from Bible book written with Visual Geez for font type Amharic, Geez, Tigrigna, and Qubee were collected. In the case of Qubee texts images contained scripts prepared using both in upper case and lower-case format. The synthetic text-line images in the dataset were generated with the most common font size and types.
The dataset split the text line image data into training, validation, and testing datasets. After creating a training and validation dataset for our recognition model, the text line image produced the document image file extension with “bin.png” format and saved the document images in the specific folder for further processing. Therefore, the researchers conducted experiments with the two datasets. For experimentation, the research printed text collected 122 pages of multiple writing word documents from sources for training and testing dataset to develop an OCR system, sample mixed writing text file document of characters set from the Bible book at Ethiopian Orthodox Churches for the training and testing purpose, sample shown in figures 7.

Figure 7: Sample collected text file image document from Bible with various languages, size and fonts style

Our experiment used summarized dataset text-line images of all characters in the mixed writing for Amharic, Geez, Tigrigna, and Qubbe scripts. To use the dataset was split into 93% for training, validation 2% of the training set, and 5% for testing randomly selected.
We provided the raw data, as well as the .bin files of the train and test sets. In another experiment, we used the dataset to train and test both our proposed model. To validate our results, and the best classifier, according to the accuracy metric, was used to report the results. From the dataset contain 20,000 text-line images and 517,610 characters were generated synthetic images. These lines may contain a mixture of Amharic, Geez, Tigrigna, and Qubee scripts writing with Visual Geez font type and size 12. For printed synthetic image dataset prepared common fonts type, size and formatting shown in table 7.

Table 7. The detail analyses dataset.

<table>
<thead>
<tr>
<th></th>
<th>Dataset Name</th>
<th># of text-line</th>
<th># of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td># of total sample</td>
<td>20,000</td>
<td>517,610</td>
</tr>
<tr>
<td>3</td>
<td># of training samples</td>
<td>18,620</td>
<td>481,840</td>
</tr>
<tr>
<td>4</td>
<td># of validation samples</td>
<td>380</td>
<td>9,741</td>
</tr>
<tr>
<td>5</td>
<td># of test samples</td>
<td>1,000</td>
<td>26,029</td>
</tr>
<tr>
<td>6</td>
<td># of unique characters</td>
<td></td>
<td>292</td>
</tr>
</tbody>
</table>

The experiment resulted in the dataset text line images used 292 unique Ethiopic includes characters, numbers, and punctuation marks with true blank. Whereas the maximum text length of the ground-truth text was 41 characters and the minimum text length 4 characters including true blank space.
Figure 8: The ground truth uses dataset unique Ethiopic characters and punctuation marks.

| 0 \n | 21 ? | 42 y | 63 u | 84 h | 105 c | 126 l | 147 r | 168 l | 189 k | 210 h | 231 h | 252 h | 273 h |
| 1 - | 22 A | 43 a | 64 w | 85 h | 106 c | 127 l | 148 r | 169 l | 190 k | 211 h | 232 h | 253 h | 274 h |
| 2 ! | 23 B | 44 b | 65 x | 86 h | 107 h | 128 l | 149 r | 170 h | 191 k | 212 h | 233 h | 254 h | 275 h |
| 3 ' | 24 C | 45 c | 66 y | 87 h | 108 h | 129 l | 150 r | 171 h | 192 k | 213 h | 234 h | 255 h | 276 h |
| 4 ( | 25 D | 46 d | 67 U | 88 f | 109 h | 130 r | 151 l | 172 h | 193 k | 214 h | 235 f | 256 h | 277 f |
| 5 ) | 26 E | 47 e | 68 s | 89 y | 110 h | 131 r | 152 l | 173 h | 194 k | 215 h | 236 h | 257 h | 278 h |
| 6 , | 27 G | 48 f | 69 l | 90 i | 111 h | 132 r | 153 l | 174 h | 195 k | 216 h | 237 h | 258 h | 279 h |
| 7 - | 28 H | 49 g | 70 m | 91 j | 112 h | 133 r | 154 l | 175 h | 196 k | 217 h | 238 h | 259 h | 280 h |
| 8 . | 29 I | 50 h | 71 n | 92 k | 113 h | 134 r | 155 l | 176 h | 197 k | 218 h | 239 h | 260 h | 281 h |
| 9 0 | 30 J | 51 i | 72 u | 93 t | 114 h | 135 r | 156 l | 177 h | 198 k | 219 h | 240 h | 261 h | 282 h |
| 10 \n | 31 K | 32 L | 53 k | 74 h | 95 w | 116 h | 137 r | 158 l | 179 h | 200 h | 221 h | 242 h | 263 h | 284 h |
| 12 3 | 33 M | 54 l | 75 m | 97 h | 117 h | 138 r | 159 l | 180 h | 201 h | 222 h | 243 h | 264 h | 285 h |
| 13 4 | 34 N | 55 m | 76 m | 98 h | 118 h | 139 r | 160 l | 181 h | 202 h | 223 h | 244 h | 265 h | 286 h |
| 14 5 | 35 O | 56 n | 77 n | 99 h | 119 h | 140 r | 161 l | 182 h | 203 h | 224 h | 245 h | 266 h | 287 h |
| 15 6 | 36 P | 57 o | 78 m | 99 h | 120 h | 141 r | 162 l | 183 h | 204 h | 225 h | 246 h | 267 h | 288 h |
| 16 7 | 37 Q | 58 p | 79 d | 100 h | 121 h | 142 r | 163 l | 184 h | 205 h | 226 h | 247 h | 268 h | 289 h |
| 17 8 | 38 R | 59 q | 80 q | 101 h | 122 h | 143 r | 164 l | 185 h | 206 h | 227 h | 248 h | 269 h | 290 h |
| 18 9 | 39 S | 60 r | 81 h | 102 h | 123 h | 144 r | 165 l | 186 h | 207 h | 228 h | 249 h | 270 h | 291 h |
| 19 0 | 40 T | 61 s | 82 c | 103 h | 124 h | 145 r | 166 l | 187 h | 208 h | 229 h | 250 h | 271 h | 292 h |
| 20 ; | 41 U | 62 t | 83 h | 104 h | 125 h | 146 r | 167 l | 188 h | 209 h | 230 h | 251 h | 272 h | 293 h |

4.2 Text Image Preprocessing

To make the dataset, a sequence of preprocessing steps was needed to make it appropriate for our model, we need to use some preprocessing. For both the input image and output labels, we needed to preprocess the text file. The OCRopus contains various modules for document image analysis, including binarization modules, page segmentation, text-line normalization, and line recognition (“OCRopus Online, n.d.). Then, in the PYTHON image processing OCROPUS tool, these images were processed using various preprocessing algorithms. Vowels and consonants, numbers, and commonly used symbols punctuation marks, uppercase and lowercase for all using scripts included. Ocropus tended to read text image column of text left to right then top to bottom order.
4.2.1 Binarization

The first step in the Octopus pipeline was to binarize the conversion of the source image from a color or gray scale image to binary image a black and white format for printed text files and then apply the threshold method over a gray scale image. First when to run binarization in ‘OCRopus-nlbins’ to generate new images of the scans that were deskewed and split those page images into images of the individual lines. To binarized the digitized image and removed the noise the researchers used the default parameters and settings of OCRopus 300dpi (dots per inch) binary black on text image and white backgound images.

4.2.2 Text Line Segmentation

Segmentation was a series of tasks or processes in which a document image was selected and segmented into the appropriate unit of image representation for further processing. The line segmentation, each line of the text was extracted by constructing the horizontal histogram. First, the histogram was constructed for the sum of values of pixels in each row and based on a selected threshold extracts a text line (SahareP. and Dhok, 2018). The stage-by-stage line segmentation algorithm effectively segmented noise free image text lines from multiple documents. For each binarized page lines were segmented using horizontal projection profile and found the window where line segment could be created.

OCRopus train model to generate an arbitrary amount of image and labeled data using ocrus-linegen. To convert the collected text file into text-line images, we used a line generation tool available in OCRopus. With a given collected text files and font-files, this tool could generate any number of text-line images. For each transcription, we changed this utility to produce a ground-truth file that matched our needs. We set numerous parameters when text-line images generated artificially, for making similar the artificially generated text line images with ground truth file generated from scanned images. OCRopus command line a new directory that would be created OCR scans of the pages and split pages into individual lines for creating synthetic image datasets. OCRopus requires utf-8-encoded text lines
(ground truth) to create the equivalent text-line images along with the type of font files for TrueType Font (.TTF). In addition, when to perform recognition annotated text line image to used better data annotation using tools. The experiment OCRopus generate truth data to be in (.gt.txt) text file corresponding with the same name as the PNG file for the line images, for example, created a directory contained training data suitable for training OCRopus with synthetic data.

_Dataset/010000.gt.txt_  
_Dataset/010000.bin.png_

The method for creating text-line image and Ground Truth (GT) text file shown in figure 9.

![Diagram showing the process of creating text-line image and GT text file](image.png)

Figure 9: To generate Text-line image and GT text file from multi-script data

To run OCRopus successfully some steps have to be completed synthetic text line image to generated new images of the scans that are deskewed, split those page images into images of the individual lines, “read” those lines, and then created an output file of the results.

Our network model was learned with a dataset organized from type of documents synthetic images. Then all binary images dataset converts in to NumPy file (.npy) split for training, validation and testing dataset, because of text image were best stored smaller memory consumption and better runtime behavior processed in python programing and all text line
image different size pixels. Show in figure 10, sample text line image takes from synthetic text image dataset and degradation level from prepared dataset.

```
አማርኛ ይምantro: Waaqayyo Jalqabatti
26 እንዳንዳን እን ከጋ ከፍስ
አማርኛ ይምantro: ከጋ ከፍስ
አማርኛ ይምantro: ከጋ ከፍስ

godhatu mukkeetiin ija

አማርኛ ይምantro: ከጋ ከፍስ ከጋ ከፍስ
```

Figure 10:- Sample text line image taken from dataset generated before resize.

### 4.2.3 Resizing Normalization

The size of text line image worked on Octopus in the data size were all text line different size pixels therefore, we resized the text line image using Python3.6 tool. The main goal of the normalization phase was to adjust for the translations in the vertical axis. All text-line images were resized to a width, we created a height target each image specific size. Then we copied each image in to the white target image, in such a way all text-line images were resized to a width of 256 and height of 32 pixels to minimize computational costs during the train model in batch. Sample synthetic image generated resized text-line images taken from the dataset were showed in figure 11.

```

godhatu mukkeetiin ija

Figure 11: Sample Ethiopic text line images normalized to a size 32 by 256 pixels.
4.3 Recognition of Proposed Optimal Model Architecture

On this section to create model which recognized text line image. To design the suitable CNN_BLSTM_CTC proposed model for multi-Ethiopic text-line image recognition. Recurrent neural networks (RNN) were the best equipped neural networks in the proposed model of the sequence recognition problem (Ghosh R., 2019), whereas for an image-based problem, the most suited were convolutional neural networks (CNN) (Hijazi S., 2015). It would be better to use combine CNNs and RNNs with CTC to deal with OCR problems (Birhanu, Tewodros, Million, Marcus, & Gebeyehu, 2020). Therefore, the three main tasks proposed were to extract features from multi-script text-line images and to predict sequential output per time-step and the transcriber module. After conducted some experimentations, the study identified proposed the CNN-BLSTM model showed in figure 12.

Figure 12: A designed proposed model for multi-script Ethiopic text-line image recognition
In this section the results of model architecture were presented. Based on experimental analysis, the CNN, BLSTM, CTC architecture was used for the proposed multi script Ethiopic text-line image recognition system model. When to configuration of the model architectures by changing different parameter values such as epoch size, batch size, optimizer selection, dropout and the layers of the network until the best fit model would be create.

**Configuration of Convolution Neural Network (CNN) layer:** CNN layer to feature extractor from text line image, the process of training a CNN model about the convolution layer was to identify the kernels that work best for a given task based on a given training dataset. Kernels were the only parameters automatically learned during the training process in the convolution layer; on the other hand, the size of the kernels, number of kernels, padding, and stride were hyperparameters that need to be set before the training process starts. We need applied max pooling were reduces the number of learnable parameters and enables the CNN to accept inputs of variable size. Training a network was a process of finding kernels in convolution layers and weights in fully connected layers which minimize differences between output predictions and given ground truth labels on a training dataset. It might be getting to understand the input and output dimensions at the end of each convolution layer. We decided to take text-line image to make you capable of identifying the output dimensions. The hyperparameter would control the size of output volume.

Here we used seven convolutional layers of the CNN network architecture, of which the 6 Conv2D layers had kernel size (3,3) and the last one was size (2,2). One was fixed on the strides and the same padding was used in all layers of Conv2D. Each uses a verifier's Rectifier Linear Unit (ReLU) activation function to create an output after each convolution layers. And the number of filters was increased from 64 to 512 layer by layer. We used four max-pooling layers with pool sizes on the first one max-pooling layers were added with size (2,2), two max-pooling layers of size (2,1) and then the last one max-pooling layers of size (1,1) were added to extract features with a larger width to predict long texts and the two-dimensional output array from the operation is called a feature map. Also, we
needed to batch normalization these layers after the fifth and sixth convolution layers which speed up, accelerates, and avoid over-fitting on the training process. It helped to accelerate training speed and improve performance. The reshape function to make the output of convolutional layer compatible with the LSTM layers.

**Configuration of Long-Short Term Memory (LSTM) Layer:** - It was a method of extracting output from the convolutional layer and making it with the LSTM. Input form for our architecture with height 32 and width 256 input image. Then the proposed model consisted of two bidirectional LSTM layers with the soft-max function on the top each of which has 128 hidden layer sizes which has a sequential output of 127 units each time-step from the LSTM layers was fed into a soft-max layer. The experiment used a network model trained for a batch size of 32 with run network 10 epochs. The batch size and the number of epochs were selected based on the best performance measurement values of the proposed network model during the experimental setup. This LSTM output sequence was mapped to a matrix size 127 by 293. The dataset consists of 292 different characters, further on additional characters were needed for CTC operation (CTC black label), therefore there were 293 entries for each of the 127 time-steps. Where 292 was the total unique number of output classes.

In each of the LSTM layers, the dropout rate of training was set at 0.25. There is the soft-max layer between the CNN output and the LSTM input which was highly effective in reducing the number of parameters used for our training. We set the total number of the output classes relevant class all total number of 292 unique scripts for Amharic, Geez, Tigrigna, and Qubee characters. The soft-max outputs could 292 probabilities classes plus including the blank characters at each time-steps. Thus, the last neural network layer used specific character weights for each output of the LSTM frame. The soft-max layer was used in the final layer to get the output class that had the highest probability in a CTC model frame.

**Configuration of the Connectionist Temporal Classification (CTC) layer** was also transcription layer the output of soft-max layer that used to calculate error loss and
decoding. For loss calculation, we feed both texts of ground truth and operation matrix. The real test of the ground was encoded as a sparse tensor. Both CTC operation had been passing the input sequences. CTC loss function was an analytical function used to minimize the loss. Whereas other loss function optimized single objective function, the CTC loss was explicitly designed to optimize the duration of the prediction sequence as well as the classes of the predicted sequence, since the input text image differed in nature. In the back-propagation process, the convolutional filters and the LSTM weights were learned jointly. Adaptive Moment Estimation (Adam) optimizer method was used for the training with a learning rate of 0.001. The Adam optimizer is a popular algorithm because it achieves good results fast and improve algorithm functionality was used.

CTC calculates loss function was very helpful in text recognition problems. It helped us prevent each time step from being annotated and helped us get the rid of problem where a single character could span multiple time steps which required further processing. In the model a CTC loss function we used four reasoning for calculating the loss, predicted outputs, ground truth labels, LSTM input sequence length, and ground truth label length. Hence the last layer of the neural network had a total unique 292 Ethiopian characters. A total of 293 class labels (292+1 for CTC blank label) were used for classification. The class labels corresponding to all Amharic, Geez, Tigrigna characters and Qubee for vowels, consonants, and uppercase and lowercase characters, commonly used number, and punctuation marks.

4.4 Experimental Result

The implementation was based on a deep learning platform that uses Keras Application Program Interface (API) python programming along with the TensorFlow backend and the model was trained on CPU. The experiment was run the following proposed network architecture in above figure 12. The batch size and the number of epochs were selected based on the best performance values of the proposed model during the experimentation setup.
For the experimental result, there used several hyperparameters when training the model, variables that determined the network structure for optimized training results. For the network, the first group of hyperparameters was determined based on the number of training datasets by selected the batch size. Those as follows:

- Number of Epoch: How many times the model reads all the data set, the values used
- Batch Size: Parts from all dataset to be visited at a time, so we divided the dataset into some batches.
- Number of iterations: The number of batches to compete for the dataset in one epoch.
- Kernel Size: Filter size to generate feature maps.
- Padding: For keeping the size of output equal to the input of the next layer.
- Stride: The distance of moving the kernel over convolving the image.
- Learning Rate: a positive fraction determining the step of learning of the network neurons.
- Dropout: a mechanism of regularization of the training to overcome overfitting based on our dataset we set values for the above parameters. We had set the value by making a comparison according to training time and performance.

The proposed model would be used for training and testing the model created “model.h5”. After conducting a number of experimentations, the proposed model used the hyperparameters and the value assigned for each parameter was described below in table.
Table 8: Configuration summary in the proposed model layers. ‘k’, ‘s’, and ‘p’ stand for kernel size, stride, and padding size respectively.

<table>
<thead>
<tr>
<th>Type of network layers</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>32<em>256</em>1</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:64, k:3<em>3, s:1</em>1, P: same</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>K:2 * 2, s:2*2</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:128, k:3 * 3, s:1*1, p: same</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>k:2 * 1, s:2*1</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:256, k:3 * 3, s:1*1, p: same</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:256, k:3 * 3, s:1*1, p: same</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>k:2<em>1, s:2</em>1</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:512, k:3 * 3, s:1*1, p: same</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:512, k:3 * 3, s:1*1, p: 0</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>-</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>K:1* 1, s:1*1</td>
</tr>
<tr>
<td>Convolution</td>
<td>#maps:512, k:2 * 2, s:1*1, p:0</td>
</tr>
<tr>
<td>Bidirectional-LSTM</td>
<td>#hidden layer size:128</td>
</tr>
<tr>
<td>Bidirectional-LSTM</td>
<td>#hidden layer size:128</td>
</tr>
<tr>
<td>Transcription</td>
<td>-</td>
</tr>
<tr>
<td>Softmax</td>
<td>No.class 293</td>
</tr>
</tbody>
</table>

Our proposed model was learned with 20,000 text line images synthetic prepared dataset. These lines image may contain a mixed of Amharic, Geez, Qubbe, and Tigrigna scripts and include punctuation marks. When to use dataset split to build the text line image model the dataset was divided into training and testing sets. We provided the raw data, as well as the .bin files, convert to the .npy file of the train and test sets. In the second experiment, we used the dataset to train and test proposed model. In order to study multi-script text-line image recognition by using a combination of BLSTM together with CTC approach, the output summery proposed a model depicted in figure 13.
4.4.1 Compile and Train the Proposed Model

On the basis of the proposed model, we will use 0.001 Adam optimizer and Keras callback features to save the best model weights on the basis of the validation loss. The result we evaluated to training and validate results was used to report the results we various parameters were evaluated until in each procedure, better results were obtained by evaluating various parameters and from the experiment result in better results were reported. Text line images are fed to the network along with their transcription, which first performs the forward propagation step. Alignment of output with associating transcription
was done in the next step and then finally backward propagation step was performed after each epoch, training and validation error were computed and best results were saved.

Training and validation errors were recorded and the network was evaluated on the test set. Therefore, with a batch size of 32 chosen, the proposed network model was trained for 10 epochs, providing better performance. After the current training epoch was completed, validation metrics were measured over the validation collection. Such percentage splitting technique is done using cross validation taken out of training data to calculate validation independently during the training process shown in the figure 14.

| Epoch 1/10 | 582/582 [==============================] - 1276s 2s/step - loss: 108.9350 - val_loss: 132.6067 |
| Epoch 2/10 | 582/582 [==============================] - 1186s 2s/step - loss: 93.3943 - val_loss: 101.2992 |
| Epoch 3/10 | 582/582 [==============================] - 1185s 2s/step - loss: 86.7997 - val_loss: 74.2918 |
| Epoch 4/10 | 582/582 [==============================] - 1185s 2s/step - loss: 57.2190 - val_loss: 58.3744 |
| Epoch 5/10 | 582/582 [==============================] - 1185s 2s/step - loss: 42.4970 - val_loss: 37.3863 |
| Epoch 6/10 | 582/582 [==============================] - 1184s 2s/step - loss: 32.3677 - val_loss: 25.7991 |
| Epoch 7/10 | 582/582 [==============================] - 1184s 2s/step - loss: 25.5582 - val_loss: 22.8944 |
| Epoch 9/10 | 582/582 [==============================] - 1184s 2s/step - loss: 18.0720 - val_loss: 15.4244 |
| Epoch 10/10 | 582/582 [==============================] - 1183s 2s/step - loss: 15.7958 - val_loss: 15.3236 |

Figure 14: Training progress report

We used a transcription function to convert it into actual texts to study the multi-script Ethiopian text-line image recognition when the model predicts the probability for each class at each step. The performance of the experiment resulted in the proposed model is improved, archives better recognition performance some epochs used. comparing training set performance against the validation set. The loss was calculated on the training and validation interpretation was how well the model was doing for those two sets. The sum of the errors made for each in training and validation sets. During each period, the training loss and validation loss was measured after each epoch. The training model up to 10 epoch and 32 batch size, plot training losses values and validation losses against of epochs. The
The graph in figure 15 shows a loss of training and validation with 10 epochs. The training loss reduced from 108.9% to 15.79% and the validation loss reduced from 132.68% to 15.32%. The training loss vs validation loss curves were plotted in Figure 15.

![Training vs Validation loss](image)

Figure 15. The training versus validation loss of the proposed was recorded for 10 epochs.

As the experiment result number of epochs increase, where there was no significant change in training. Adding a dropout layer to the model helps to decrease the risk of a model being overfitted. In comparison, the loss of validation increases, illustrating the overfitting of the training data model. So, the result in training loss was nearly greater than the valid loss it was due to some just right and better job classifier model relation between input data image and output targets.

### 4.4.2 Test the Proposed Model

Our proposed model was trained the network, we artificially prepare 20,000 text-line images and 517,610 characters with the corresponding ground-truth in a common font style and font sizes. We used our training model because it also required labels as input and at test time, we could label. So, to test the model we would use” model.h5” that we had created earlier which took an input of test text line images. In performance metric according to Levenshtein distance (Li & Liu, 2007), the measurement metric based on distance was better in handwriting OCR system, this study in order to calculate the edit distance.
performance of the generated model that we used techniques for evaluating CER using this technique, the ratio between deletion (D), Insertion(I) and Substitution(S) of error characters in the predicted output and the total number of ground-truth (GT) test characters, were determined by the model’s character error rate (CER).

\[
CER(\%) = \frac{D + I + S}{Total\ test\ characters} \times 100
\]

4.1

The multi-script Ethiopic text-line image recognition was developed using the implementation proposed model for the trained prepared dataset. The performance of the training multiple writing Ethiopic scripts model was tested. We performed an experiment result, depending on the designed datasets. The model was evaluated by synthetic text line image type experiment with Visual Geez font type and 12 font size. Finally, the result proposed model tested dataset character error rate experiment summarized testing 9.5% was recorded which, were generated printed artificial text line image from multiple writing Ethiopian documents. So, the experimental results errors were calculated by summing all over CTC losses.

4.5 Discussion of the Result

The experiment was conducted using a combination of CNN and BLSTM together with CTC approaches. This may be due to the ability of CNN to extract important features and BLSTM able to learn backward and forward related information of the text. Since our model predicts the probability at each time stage for each class, we need to some transcription feature to translate it into real texts. We're going to use the CTC decoder here to get the output text. Character recognition errors occur when we test the Ethiopic OCR model. The most common cause of the recognition errors occurred due to similarity of characters. The testing results used font type Visual- Geez and font size- 12 for 26,029 number of characters from the dataset.
The proposed model achieved better recognition performance with a small number of epochs so, the proposed model took short training time based on our dataset. Compared with the character error rate observed on the test results, the model proposed works from synthetic test dataset Latin script for Afan Oromo writing script better recognition than Amharic, Geez, and Tigrigna scripts produced from the dataset, since the dataset contain larger number of Amharic scripts used in the testing results. The results output prediction text containing different types of character errors which may occur due to misspelled error characters, lost characters are presented. For sample multi-Ethiopic text-line images and characters misrecognized for predicted texts contained different types of character errors like deletion, substitution, and insertion during testing were shown in figure 16.

Figure 16: Sample misrecognized multiple text-line images

As shown at the above of the sample figure 16A, the character (4) marked in the input image with a green rectangle was substituted by the character (3) in the predicted text. In
figure 16B, the three character (አ, ል, and s) marked by the red rectangle were deleted characters and a character (ከ) marked in the input image with a green rectangle was substitute by the character (ም) in the predicted text. In figure 16C, the character (ት) marked in the input image with a green rectangle was substituted by the character (ሠ) in the predicted text. on the other character error show at the top of figure 16D, one character (ጾ) marked in the input image with a green rectangle was substituted by the character (ጾ) in the predicted text, and the character (ጭ) an orange rectangle was inserted character into predicted text output. Finally, from the proposed model the research CER 9.5% performance on the test dataset. The rationale is that this model trained with a training set contained in models plus numerals and punctuation marks. Finally error analysis the characters recognition errors occurs when we test model commonly, the cause of the recognition errors occurred due to the similarity shapes of Ethiopic characters.

In general, this paper proposed CNN-BLSTM architecture with the CTC model and used prepared dataset training set for training and testing. The observation of comparing experimental results demonstrated multiple writing Ethiopic text line image recognition. The model used for the performance of the CNN-BLSTM combining approach was less expensive in memory and computing time and prove more performance.
CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

This section contains the conclusion of observations from our research work. It also contains recommendations to show further researches that can be done in this area.

5.1 Conclusions

Deep learning self-extracts features with a deep neural network and classifies itself. Compare to traditional algorithms its performance increase with the amount of data and its model can achieve state-of-the-art accuracy. The purpose of the study developed a recognition model that can recognize machine printed multiple writing Ethiopic script documents using a combination of CNN and BLSTM with CTC neural networks. To train and test the model, the dataset was prepared from sources multiple written in Geez, Amharic, Tigrigna, and Qubee scripts. The study collected and prepared from multi-script Ethiopic text 20,000 text-line images were contains 517,610 characters synthetic printed images. A dataset was preprocessed using the OCROPUS tool generate corresponding to ground truth and text line image. We also propose a technique that reduces the time and effort needed for manual annotation of printed multiple Ethiopic documents. Observable feature values of text-images have been taken after a series of preprocessing and segmentation activities. Binarization, text line-segmentation, and size normalization are the common tasks that were performed by the multi-script identifier.

The use of a text line will allow the detector to reduce labeling costs, and the pre-trained model can improve the performance of the detector. The experiments showed that the low-cost text labeling line can be used to train and further validate the feasibility of reinforcing weak labels using a synthetic text dataset. A new recognition model architecture consisting of three parts of the Convolution Neural Network (CNN) was developed in the present paper to extract features from the input image, the Long Short-Term Memory (LSTM) to predict sequential output per time-step, and the transcription layer is used to predicted output per time step. It better to use combine proposed algorithms because eliminating the segmentation procedure and training reduced computational costs.
The performance test results show that the model of multiple Ethiopic text-line images an overall performance 9.5%-character error rate for the printed test sets. Results from the experiment proposed model showed the effectiveness of the multi-script recognition model, so, script recognition is processed in multi-script optical character recognition (OCR) and needs a high level of accuracy, however, more comprehensive future works make this finding more improved.

5.2 Contribution of the Study

Some of the contributions of the research work would be the first attempt to the analysis of multi-script of Ethiopic characters are listed below:

- Building new dataset from the real printed multiple writing documents synthetic text line image and for detail analyzing and identifying of the dataset is presented.
- We have conducted the preprocessing steps for extracting better datasets to reduce computational time and also achieve better performance values that have been gained.
- To develop a new model for multi-script Ethiopic text-line image using a combination of CNN and BLSTM with CTC deep learning approaches.
- To evaluate Ethiopic OCR system ease analysis and understanding for writing multiple script documents recognition.
- To create an efficient OCR system with minimal costs that is, minimize human effort during the annotation process as well as the reduce time required to train models.
5.3. Recommendation

This research is a first attempt in the recognition for Amharic, Geez, Tigrigna, and Affan Oromo scripts from mixed writing documents image. Although there is still a gap in this proposed work, the proposed work that can be further extended to improve the performance. Therefore, the following are some notable future work recommendations observed while implementing this research work:

- In this work, the amount of dataset used is not much. It is better to increase dataset since deep learning needs a huge amount of data and further system to achieve high accuracy.
- This study considers only Amharic visual Geez fonts. An effort should be made towards the development of the system that works to all other Latin and Ethiopic fonts type.
- The proposed method will be extended for other multiple writing Ethiopic scripts document image recognition. Similarly, we planned to create an OCR system that would recognize some complex multiple writing Ethiopian documents, such as Ethiopian text images of historical and natural scenes.
- Additionally, the future work, the proposed method will be extended for multi-script Ethiopic handwriting historical document image recognition.
REFERENCES


Alex Graves, N. J. (2013). Hybrid speech recognition with deep bidirectional LSTM. In ASRU, 273-278.


Bender, M., S.W., & Cowley, R. (1976). The Ethiopian writing system. In *Bender,Bowen, Cooper and Ferguson (eds.)*.


Lawrence , L. (2010). a companion from prehistory to today of world writing system. *Ancient scripts.com:.*


Appendices

Appendix I: configuration of the proposed deep learning models

```python
# input with shape of height=32 and width=256
inputs = Input(shape=(32, 256, 1))

# convolution layer with kernel size (3,3)
conv_1 = Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)

# pooling layer with kernel size (2,2)
pool_1 = MaxPooling2D(pool_size=(2, 2))(conv_1)
conv_2 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool_1)
pool_2 = MaxPooling2D(pool_size=(2, 1))(conv_2)
conv_3 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool_2)
conv_4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv_3)
pool_3 = MaxPooling2D(pool_size=(2, 1))(conv_4)
conv_5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool_3)

# Batch normalization layer
batch_norm_1 = BatchNormalization()(conv_5)
conv_6 = Conv2D(512, (3, 3), activation='relu', padding='same')(batch_norm_1)
batch_norm_2 = BatchNormalization()(conv_6)
pool_4 = MaxPooling2D(pool_size=(1, 1))(batch_norm_2)
conv_7 = Conv2D(512, (2, 2), activation='relu')(pool_4)

squeezed = Lambda(lambda x: K.squeeze(x, 1))(conv_7)

# bidirectional LSTM layers with units=128
blstm_1 = Bidirectional(LSTM(rnn_size, return_sequences=True, dropout=0.25))(squeezed)
blstm_2 = Bidirectional(LSTM(rnn_size, return_sequences=True, dropout=0.25))(blstm_1)

outputs = Dense(len(char_list)+1, activation='softmax')(blstm_2)
```