Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Character Recognition of Bilingual Amharic-Latin Printed Documents

A thesis submitted to Addis Ababa Institute of Technology, School of Graduate Studies, Addis Ababa University

in

Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Engineering

By

Abeto Alemu Kesito

Advisor: Menore Tekeba

Addis Ababa

November 2018

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Character Recognition of Bilingual Amharic-English Printed Documents

By

Abeto Alemu

Approval by Board of Examiners

_____	_____
Chairman Department of Graduate Committee	Signature
Menore Tekeba_	_____
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

DECLARATION

I declare that the thesis with title of "*Character Recognition of Bilingual Amharic-English Printed Documents*" is my original work and has not been presented for a degree in any other university. Where material has been used from other sources, it has been properly acknowledged.

ABETO ALEMU Signature_____

Date of submission: _____

Place: Addis Ababa, Ethiopia

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

Advisor: MENORE TEKEBA Signature _____

ACKNOWLEDGEMENTS

First of all, I would like to thank Jesus for supporting me throughout all challenges in my research work and my life.

Next, I would like to express my gratitude to my Advisor Menore Tekeba for your useful comments, remarks and for introducing me to the topic as well for the support on the way.

Finally, I want to Acknowledge my wife Selamawit W/mariam and by beloved son Natnael Abeto for all their support and love during my research work.

ABSTRACT

Optical character recognition (OCR), is system that automatically converts captured images of handwritten, typewritten or printed text documents into machine encoded text. In Ethiopia more than 80 language are spoken and those languages use either Amharic scripts or adopted Latin scripts. In such environment, in order to reach a larger cross section of people, it is necessary that a document should be composed of text contents in different languages written in Amharic and/or Latin characters.

To prepare dataset, several documents were collected from different sources for both script types. Character images were collected for 231 Amharic characters and 52 characters for English (merged capital and small letters). Totally for 257-character classes, 49,087-character image are prepared to train and test the system. Randomly selected 80% of dataset were used to train the system where as remaining 20% for purpose of testing the accuracy.

Data acquisition, image binarization, noise removal, skew correction, character segmentation, feature extraction and character classification are steps in developing character recognition system. A number of algorithms were implemented to develop the proposed OCR system. In this research work, it was discussed the process of developing an OCR for bilingual Amharic and Latin script using Convolutional Neural Network (CNN) which is feature extraction and character classification model. From the experiment 99.20% of classification accuracy was obtained when the number of neurons is 256 and with adaptive learning rate. In character segmentation stage, average of 98.85% accuracy was achieved for clear sample document and 95.86% for unclear sample documents. Therefore, overall recognition accuracy become 98.06 % and 95.09 % respectively.

Key words: Bilingual OCR, CNN, Neural Network, Convolutional Neural Network, Ethiopic OCR.

Table of content

LIST OF ABBRIVATION

- OCR Optical Character Recognition
- K-NN K-Nearest Neighbor
- HMM Hidden Markov Model
- OpenCV Open Source Computer Vision
- ReLU Rectified Linear Units
- IPA International Phonetic Alphabet
- BGN Board on Geographic Names
- PCGN Permanent Committee on Geographical Name
- PNN Probabilistic Neural Network
- RQ Research Question

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE : INTRODUCTION

1.1. Background

Libraries and archives house have many old books that are of importance to historians and other scholars. However, to search and read these texts the historian needs to visit a library and carefully handle delicate old books. If these books could be made available online in a searchable way that would be very useful to these scholars and other interested persons.

OCR technology enables us to convert documents such as text document images captured by a digital camera or image scanner, pdf files into editable and searchable text document. The main purpose of an OCR is to make machine encoded from existing paper documents or image files. In our day to day life, we often need to reprint text with after editing or modification. However, in many cases, the printable document of the text does not remain available for editing. For example, if a newspaper article written in Amharic was published 6 or 12 years ago, it is quite possible that the text is not available in an editable document such as a word or text file. So, it is required to type the entire text which is a very exhaustive process if the text is large. The solution of this problem is optical character recognition system. It is a process which takes captured images as inputs and generates the texts from images. So, a user can take an image of the text, feed the image into OCR and then the OCR will generate an amendable text file for the user which is editable and stored for farther uses. Amharic is one of the Languages in Africa which uses its own scripts to represent textual and numeric information [2]. Amharic is an official language for Ethiopian government. A number of documents are produced daily in this language scripts. There are also a great number of collections in libraries, museum which is already produced with the specified language scripts [3]. Amharic document conversion to electronic format is essential with respect to historical documents preservation, saving storage space, and facilitating retrieval of relevant information via the internet [3]. As we could find a

number of historical documents written in Amharic fonts, there are also a number of official, historical, legal, and academic documents which are written with bilingual fonts particularly in Amharic and English scripts in many institution, museums, and libraries[4]. For instance, bilateral treaties that performed between the Ethiopian and other countries government since a century ago, Ethiopian constitution, letters, Amharic-English or Amharic-Other dictionaries those adopted Latin script were written in a mixed language script.

The character recognition is achieved through important procedures like scanning documents or taking photograph using scanner or digital camera, pre-processing to clear unwanted pixels, segmenting each character form scanned or imaged document, extracting features of a character and finally recognizing characters by training the network using the prepared dataset. OCR has gained increasing attention in both academic research and in industry. It has been man's ancient dream to develop machines which replicate human functions [1]. One of such replications of human functions is reading of document images broad forms of text. Over the last few decades' machine reading has grown from dream to reality through the development of advanced OCR systems [5]. OCR belongs to the classes of machine recognition techniques performing automatic identification. Automatic identification is the process where the recognition system identifies target automatically, gather data about the target object and enters data directly into computer without human inclusion. The traditional way of entering data in a computer is through the keyboard. However, this is not always the best or the most effective way. The automatic identification may serve as an alternative in many cases. There exist various techniques for automatic identification which cover the needs for different application areas.

All optical character recognition software starts with an electronic image of the text, usually created with a document scanner [5]. The scanner only produces an image of the document, much like taking a picture of it. The optical character recognition software then examines the image of the scanned document, identifies each letter,

number and punctuation mark, and produces equivalent text in a machine-readable digital form that can be used by a computer system.

A character recognition system is categorized into online and offline character recognition system. Characters are recognized at a time it has being written for online recognition system. It accomplish better than offline system as they have timing information and since they forefend the search step of locating the character at the beginning [1]. Online systems get the position of the pen as a function of time directly from the interface. This is always done through pen-based interfaces where the writer writes with a pen prepared for such purpose on an electronic tablet. There are two category of offline character recognition, handwritten and printed character recognition. In offline character recognition, written paper documents are scanned to form image and brought to the recognition system. Therefore, offline character recognition can be said as it is more challenging task than its online counterpart[1]. Handwritten character recognition is even more challenging than printed character recognition as handwriting of persons are differs in various attributes like size of characters, their shape, style of writing and so on.

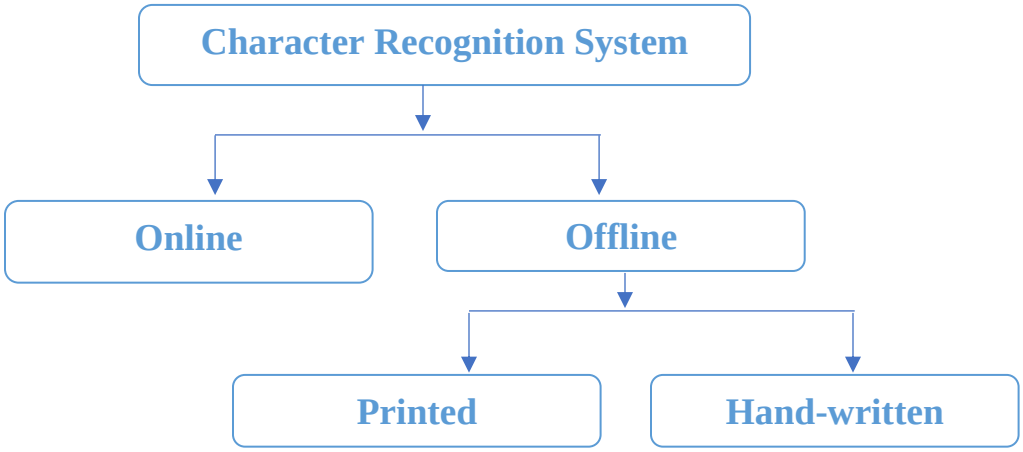


Figure 1: - Type of OCR source [1]

System developers face challenges in developing offline OCR system. Firstly, visual similarity between many numerical and alphabetical symbol shapes. For example, in Amharic character recognition there is little visible difference between an italic

ፓ(Pa) and regular ፐ(po) or many other characters. The morphological similarities between characters in Amharic script is higher especially when they are italic font style. Humans can re-read the sentence or whole paragraph to understand the correct meaning. However, this procedure is much more challenging for a machine. The other challenge is contrasts in document image in recognizing characters. It is very challenging to read text which appears against a very dark background or is printed over words or graphics or character written in light color.

A lot of algorisms have been used for implementation of this bilingual Amharic-English OCR system in binarization of scanned image, binarization, skew detection and correction, character segmentation, feature extraction and character classification. In the research for feature extraction and character classification Convolutional Neural Network (CNN) was used. CNNs are hierarchical neural networks that have vast representational capacity and have been successfully applied to many problems such as printed and handwritten character recognition and visual object recognition system.

1.2. Statement of the Problem

OCR system for Amharic-English characters is required to convert numerous printed books of Amharic characters as well as bilingual Amharic-Latin characters into editable computer text files. Europeans, Americans and others have been conducting researches and applying OCR technologies to their languages. As a result, these systems can read different documents written in English, Latin, Japanese, Chinese, Hindu, Arabic, Russian, and the like but do not read documents written in Amharic. Those days in Ethiopia, there are documents which are produced in bilingual scripts particularly in Amharic and Latin scripts. For instance, the civil code, the constitutions of the country, newspaper, letters etc. are written in Amharic and Latin script on the same document. Therefore, we need to Ethiopic bilingual OCR system to recognize characters from document images.

In Ethiopia more than 80 languages are spoken and almost all languages adopted Latin script that uses English alphabet and the other use Amharic characters. Most of documents in Ethiopia are written either in one or both of scripts. Therefore, for successful character recognition of those documents, it is required to have bilingual OCR system to recognize characters.

Research Questions (RQ)

- RQ1: Can the character classification accuracy be improved using CNN?
- RQ2: Can we maintain the overall recognition accuracy with inclusive of most Ethiopic and Latin characters?

1.3. Objective of the Study

1.3.1. General Objective

The general objective of this research is to extract and recognize English and Amharic characters from printed documents written either in both scripts or one of those scripts.

1.3.2. Specific Objectives

The specific objectives of the study are:

- ✓ Preparing dataset for training and testing the network.
- ✓ To select appropriate algorism for image processing and character segmentation.
- ✓ To test the classification accuracy of the segmented characters using CNN.
- ✓ To develop an OCR system that can be applicable for practical use.

1.4. Scope and Limitation of the Study

Since preparing dataset for all characters is very difficult and time consuming, the dataset was limited to all Amharic characters and all English alphabets excluding numbers, punctuation marks and Amharic labialized characters. 231 characters

from Amharic script and 52 English characters with merged uppercase and lowercase.

1.5. Methodology

In this sub topic, method of data collection, tools used in the research work and procedures in the implementation of the system were discussed.

1.5.1. Data Collection

Datasets were collected from different sources that written in both Amharic and English or one of script, like newspapers, fictions, books, letters, civil code, journals, and different religious documents. The dataset was written in different font size and font format.

In general document images are collected from two sources.

A. Datasets from real-life documents such as newspapers, fictions, books, letters, journals, and different religious document. Those documents are captured by scanner and digital camera. From such documents it is difficult to get required number of character images for all characters. Because like vowels in English character, in Amharic also some characters appear repeatedly like character "ን", "የ", "በ", "አ" etc. But many characters like family of "ጰ", "ጸ", "ጎ" and other characters may not get one from many document images. Therefore, to get balanced numbers of character images for all, the following type of documents are used.
B. Synthesized document image prepared for dataset. It is prepared in Microsoft word with a variety of font type and style. The document has font style of Regular, Bold, Regular-Italic and Bold-Italic and font type of Visual Geez Unicode, VG2000 Title, Power Geez Unicode1, Power Geez Unicode2, Power Geez Unicode3, Ebrima, AbnetZethion, AbnetZeSematate, AbnetZeMelakt, AbnetZeHawariat, Ethiopic and Ethiopia-Jiret for Amharic characters. For English character it contains font type of Times New Roman, Calibri, Albertus,

Arial, Candara, Century, Felix Titling, Impact and Microsoft YaHei Light were included. Prepared documents were captured in the following three ways:

- Image taken from screenshot: - This type of image was noise free.
- After printing prepared document, it was captured by scanner or digital camera
- After printing the document, it was captured after treating printed paper with noise. The noise treatments were done by putting on dust, inserting in oily water and distorting papers.

Both collected real-life documents and prepared documents were scanned with different resolution level. Most of those documents were scanned with Epson L382 scanner with resolution of 200dpi to 1200dpi. The remaining captured through digital camera and already scanned book from internet. The dataset consists of binary image of a normalized size of 32x32 pixels after preprocessing and segmentation. In this research dataset consists of 231 Amharic characters and 52 English characters (both capital letter and small letter are taken as single character class). For each 257 different characters, 191 different character images were prepared. Therefore, total collected characters for dataset were 41,120character images as shown in the Table 1. From this sample data 80% randomly selected for training dataset and the remaining 20% for test dataset.

Table 1: - Summary of prepared dataset

Amharic			**English**		
Number of character classes	Numbers of characters for each class	Sub-Total	Number of character classes	Numbers of characters for each class	Sub-Total
231	160	36,960	26	160	4,160
		Total	41,120		

In addition to the above dataset, randomly selected 10% of dataset was augmented by rotating with random angle between -5^0 to $+5^0$ and 10% of dataset treated by Gaussian noise. Therefore, total number of training and testing dataset become 49,087 characters.

Adding rotated images to dataset is used to train the system in order to recognize skewed characters due to non-uniform text line skew. Non-uniform text line skew type of skew cannot deskewed due to the existence of different skew angle in single line. Also treating some images by noise is used to train the system to identify characters with noise.

1.5.2. Preprocessing

Pre-processing step involves RGB to grayscale image conversion, noise removal, line removal and removing skew. Preprocessing techniques are needed on color, grey-level or binary document images containing text and/or graphics. In character recognition systems most of the applications use grey or binary images since processing colored images are computationally higher [6]. Such images may also contain non-uniform background and/or watermarks making it difficult to extract the document text from the image without performing some kind of preprocessing. Therefore, the desired result from preprocessing is a binary image containing text only. To achieve this, several steps are needed. First, some image enhancement techniques to remove noise or correct the contrast in the image. Second, thresholding to remove background containing any scenes, watermarks and/or noises. Finally, line and skew are removed.

1.5.3. Segmentation

Character segmentation is the process of dividing a document image into separate characters. This is typically used to identify objects or other relevant information in digital images. Generally, document is processed in hierarchical way. At first level

lines are segmented using row histogram. From each row, words are extracted using column histogram and finally characters are extracted from words.

1.5.4. Implementation

For implementation of the system, Python programing language was used. Libraries like Google TensorFlow, OpenCV for image processing, NumPy to compute large multi-dimensional arrays and matrices were selected.

1.6. Organization of the Thesis

This research paper was organized in to six chapters. The first chapter introduce about the research. Objectives, statement of problems, scope of the study and general methodology are discussed in this chapter. Chapter Two is about related work. In this chapter a lot of related works done by different researchers are reviewed. Chapter Three is about history of OCR, procedures in multilingual OCR and its challenges. In the fourth chapter implementation procedures and algorisms used for the research were discussed. In Chapter Five about result of the research found from experiment is discussed. The last chapter conclusions and recommendations are mentioned.

CHAPTER TWO: LITRATURE REVIEW

2.1. History of OCR

To perform the human functions by machines or making the machine capable to carry out tasks like reading documents, is an ancient dream [7]. According to [7] character recognition can actually be found back in 1870. This was the year that C.R.Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells. After two decades the Polish P. Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. Different attempts were made during the first decades of the 19'th century to develop devices to support the blind by try out with OCR. However, the modern version of OCR did not exist until the middle of the 1940's with the development of the digital computer.

The commercial OCR systems appearing in the period from 1960 to 1965 may be called the first generation of OCR. This generation of OCR machines was mainly characterized by the constrained letter shapes reading. The symbols were specially designed for machine reading. With time multi-font machines started to appear, which could read up to ten different fonts.

Second generation of the reading machines came out in the middle of the 1960's and early 1970's. These systems were able to recognize printed characters of regular machine and also had hand printed character recognition.

For the third generation of OCR systems, appearing in the middle of the 1970's, the challenge was documents of lower quality and large number of printed and hand-written character sets.

In fourth generation OCR, complex documents intermixing with text, graphics, tables and mathematical symbols, unconstrained hand-written characters, colored files, low-quality noisy documents reading. Among the commercial products, postal address readers, and reading aids for the blind are available in the market.

Even though OCR machines became commercially available already in the 1950's, due to price of the system only a few systems had been sold worldwide up to 1986 [7]. However, as hardware was getting cheaper, and OCR systems started to become available as software packages, the sale increased considerably and the price is decreasing.

2.2. Overview of Ethiopic Script

❖ Ge'ez Script (ፊደል)

According to [8] there are basically two hypotheses on the origin of Amharic: it may be a descendent of a common Proto-EthioSemitic language or it may have evolved as a Semitic-based pidgin, which became a creole and eventually developed into a full-fledged language. While the first hypothesis is commonly accepted in Semitic studies, the second hypothesis first proposed in the beginning of the 1980s. It became quite popular very recently and even entered Amharic textbooks. The earliest known inscriptions in the Ge'ez script date to the 5th century BC. At first the script represented only consonants. Vowel indication begin to appear in 4th century AD during the reign of king Ezana, though might have developed at earlier date.

There are different families of writing systems which serve different language users. Writing system grouped in to the following families: Alphabetic, Consonantal (abjad), Syllabic (Abugida), Alpha-syllabic and Logographic. Geez script grouped to abugida (አቡጊዳ) writing system. Its writing direction is left to right in horizontal lines. Each symbol represents a syllable consisting of a consonant plus a vowel. The basic signs are modified in a number of different ways to indicate the various vowels.

❖ Amharic (አማርኛ)

Amharic is a Semitic language and the national language of Ethiopia (ኢትዮጵያ). In addition to Ethiopia, Amharic is spoken in a number of other countries, particularly

Eritrea, Canada, the USA and Sweden [9]. The name Amharic (አማርኛ) comes from the district of Amhara (አማራ) in northern Ethiopia, which is thought to be the historic center of the language.

Amharic is written with a version of the Ge'ez script known as ፊደል (Fidel). There are a number of ways to transliterate Amharic into the Latin alphabet, including one developed by Ernst Hammerschmidt, the EAE Transliteration system, developed by Encyclopedia Aethiopica, and the BGN/PCGN system, which was designed for use in Romanizing names written in Amharic characters and adopted by the UN in 1967 [9]. Amharic script has now 33 core characters. Each of which occurs in seven different forms or orders (one basic form and six non-basic forms). Here is list of Amharic core characters, numerals and punctuation marks are show in table 2 and the remaining all labialized and core characters are shown in the appendix B.

Table 2: - All Amharic Core characters

1st	2nd	3rd	4th	5th	6th	7th
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ

ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

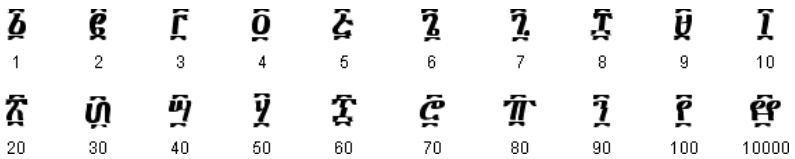
Punctuation marks

Here is a list of Amharic punctuation marks.

፡	።	፣	፤	፥	፦	፧	፨
space entre mot	point	Virgule	Point Virgule	Double Point	Preface Colon	Point Interrogation	Separateur de paragraph

Numerals

Ethiopian Numerals which consists of symbols from 1 to 10 (፩ ፪ ፫ ፬ ፭ ፮ ፯ ፰ ፱ ፲) and multiples of 10 it means ፳ ፴ ፵ ፶ ፷ ፸ ፹ ፺ ፼ ፼፼



The seven orders represent syllable combinations consisting of a consonant and following vowel. This is why the Amharic writing system is often called a syllabary rather than an alphabet. The non-basic forms are derived from the basic forms by more or less regular modifications. Other symbols representing labialization, numerals, and punctuation marks are also available. These bring the total number of characters in the script to 349 [3]. Table 3 shows summary of the number of characters in each group.

Table 3:- Total Number of Symbols in Amharic Writing System (FIDEL)

	Type of Amharic Character	Number of Character
1	Core Character	231
2	Labialized Character	89
3	Punctuation mark	9
4	Numerals	20
	Total	349

Existence of such large number of Amharic characters in the writing system is a great challenge in the development of OCR for the language. In addition to the above characters, since Amharic writing system does not have a symbol for zero, negative, decimal point and mathematical operators, the Hindu-Arabic numerals and Latin mathematical operators are used for computational purpose.

❖ **Characteristic of Amharic Script**

Amharic characters have a number of notable characteristics. This is mainly attributed to character formation in the script. Since one character is formed from the other by adding strokes or marks, most of the alphabets have some common features [3]. According to Million M. some of these features of Amharic characters are discussed below.

Shape similarities among characters: The shape of many Amharic characters shows similarities with few distinctions among them. The distinction is shown, for

instance by adding strokes at the top of the character, for example, 'ሰ' and 'ሸ', 'ደ' and 'ጀ', 'ተ' and 'ቸ', etc.

Structural relation: Many basic characters are also clearly related in graphical structure, for instance, 'ነ' and 'ከ', 'ረ' and 'ፈ', etc. There are very similar characters that are difficult even for humans to separate visually, for example 'ለ' and 'ሰ'; 'ደ' and 'ጸ', etc., especially when one of them is italic font style.

Shape differences: There are also remarkable differences in shapes among the basic characters. Consider 'ሀ' and 'በ' (both are open in one side but in opposite direction),'መ' and 'ወ' (both are formed from two loops but differ in the connection of the loops), etc.

Vowel formation: An interesting peculiarity of the Amharic writing system is the way vowels are formed. Vowels are written with small appendages to the consonant letters, with modifications of their shapes. This method of writing vowels is similar to that of Indic alphabets. Specifically speaking, vowels are derived from consonants in two ways. Some vowels (such as the fourth and seventh orders) take a modified shape of the base character by shortening/ elongating one of its main strokes. On the other hand, adding strokes (and loops) to the right, left, top or bottom of each base character forms the remaining vowels (like second, third and fifth orders). As shown in table 4, the second, third, and fifth orders are formed (with few exceptions) according to patterns of great regularity, while the fourth, sixth and seventh orders are highly irregular. For instance, the second order is mostly constructed by adding a horizontal stroke at the middle of the right side of the base character; whereas, the sixth order is formed by adding a stroke, loop or other forms in either side of the base character.

Table 4:- Amharic characters vowel formation

Base Character	2nd order	3rd order	4th order	5th Order	6th Order	7th Order
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ

2.3. Overview of Latin Script

Latin alphabet, also called Roman alphabet, most widely used alphabetic writing system in the world. The standard script of the English language and the languages of most of Europe and those areas settled by Europeans [10].

According to [10] the classical Latin alphabet consisted of 23 letters, 21 of which were derived from the Etruscan alphabet. In medieval times the letter I was differentiated in to I and J and V into *U, V,* and *W,* producing an alphabet equivalent to that of modern English with 26 letters. In ancient Roman times there were two main types of Latin script, capital letters and cursive.

The script is either called Roman script or Latin script, in reference to its origin in ancient Rome. Unicode uses the term "Latin" as does the International Organization for Standardization (ISO) [11]. The numeral system is called the Roman numeral system; and the collection of the elements, Roman numerals. The numbers 1, 2, 3 ... are Latin/Roman script numbers for the Hindu–Arabic numeral system.

2.4. Review of Literatures

There are some related research works done in the area. In this sub-topic literatures will be reviewed.

Sertse A.[2] did a research on a system that distinctly identifies Amharic and English Scripts from a document image. The system addresses the language identification problem on the word level. The goal of the research is to study the patterns and features that classify Amharic scripts from English scripts and to develop a method that helps in identifying of bilingual scripts. Support Vector Machine algorithm is applied to classify new instance word images. From the proposed script identification model recorded accuracy between 87% and 93% on real-world documents. The model achieved higher recognition rate in Visual Geez Unicode fonts than power Geez fonts.

Million et al [12] did on optical character recognition of Amharic documents. They used a two-stage feature extraction scheme; Principal component analysis followed by linear discriminant analysis for optimal discriminant feature extraction. Support vector machine was used for classification task. Finally, 88.23% to 91.45% accuracy were obtained from different real-life documents.

Yaregal A. et al [13] to implement Amharic character recognition system, they used direction field tensor as a tool for character segmentation, and extraction of structural features and their spatial relationships. They obtained about 92% of accuracy for clear documents and 86% for real life documents.

Donahue et al.[14] presents one of the first studies on the behavior of convolutional filters in the process of extraction of features. In that work, authors used a CNN. The architecture composed by 5 convolutional layers and 3 fully-connected layers trained using ImageNet 2012 as (TS, DS) for transfer learning. The authors observed how features from the first fully connected layer outperform features from lower layers at the task of separating concepts according to the Word-Net hierarchy. Authors test features extracted from the last convolutional layer and the first two

fully connected layers on several prepared datasets. Their results indicate that by using features from the first fully connected layers to train SVM one can achieve state of the art results on different related tasks.

In the research [15] used a natural scene text image consisting of various scripts is given as input and training involves applying discrete wavelet transform for feature extraction and linear discrete analysis for classification of different scripts. In the phase of training, the texture features are extracted from the training samples selected arbitrarily corresponding to each script using the feature extraction algorithm. These features are kept in the feature library. In order to train the system, they took images of each Script from different angles. Once all the images were taken by scanner and digital camera, small numbers of them were selected for training and testing purpose. The proposed KNN approach could successfully identify the three types of script words (Kannada, English and Hindi) with an average accuracy of 90%. The PNN based classifier could successfully identify the three script words (Kannada, English and Hindi) with an average accuracy of 95%. They concluded as the PNN based classifier is better than the KNN based classifier.

Pithadial.N.et.al [16] methods of feature extraction used in optical character recognition was discussed. Offline character recognition has division like magnetic character recognition and optical character recognition. Also, classification of feature extraction techniques is discussed. It is concluded with that efficiency is based applicable.

Vasudeva.N.et.al [17], represents each character as a single image. Each containing a single character and the gray image format coverts to binary format where each zero and one represents an individual pixel of the image. After feeding binary data into neural network, the output from the neural network is changed into ASCII text and saved as a file. The research work takes the help of Multi-Layer Perception Neural Network for recognition of printed image document. Difficultness occurs if preprocessing, feature extraction in recognition are done over large number of datasets. It is concluded with reduction of error function results of increment in

hidden nodes and epochs for character recognition. The network that is implemented is made up of input layer, output layer and hidden layer. The input layer constitutes of 180 neurons which receive printed data from a 30 x 20 symbol pixel matrix, the hidden layer constitutes of 256 neurons whose number is definite based on the optimal results on a trial and error basis and the output layer is made up of 16 neurons and in average they got about 1.11% error.

Paper [18] presented that projection profile is used as a suitable feature for in the detection of skew angle. Vertical Projection Profile Analysis allows few noises which produces error where Horizontal Projection Profile Analysis reduces the effect of noise. And they concluded the time complexity of vertical projection profile is greater than horizontal projection profile.

In this paper [19] they discussed different methods which are existing for character segmentation. Most methods are divided into three group. They are the analytical, the empirical goodness and the empirical discrepancy. Segmentation algorithms can be assessed analytically or empirically. So, the evaluation methods can be divided into two categories: the analytical methods and the empirical methods. In the analytical methods directly study and evaluate the segmentation algorithms themselves by analyzing their principles and properties. The empirical technique indirectly decides the segmentation algorithms by applying them to test images and measuring the quality of segmentation. The Empirical methods are classified into two types: empirical goodness and empirical discrepancy method. In empirical goodness method, properties of segmented images are measured using goodness parameters. Where in the empirical discrepancy method some references that presents the ideal or expected segmentation results are first found.

CHAPTER THREE: MULTILINGUAL OCR

3.1. Introduction

Multilingual OCR is the mechanical or electronic conversion of scanned images of hand written, typewritten or printed text from more than one script into editable text. It is used in as a form of data entry from some sort of original printed paper data source, such as sales receipts, books, or any number of printed records. It digitizes printed texts so that they can be automatically searched, sorted, stored more efficiently, shown on-line, and used in machine processes such as machine translation, text to speech translation and text mining.

3.2. Tasks in Multilingual OCR

OCR is a simple machine with several fragments that works according to the software design as shown in the Figure 2. At opening, a typical OCR contains of an optical scanner that scans the analog document. The next stage is of preprocessing of scanned image.

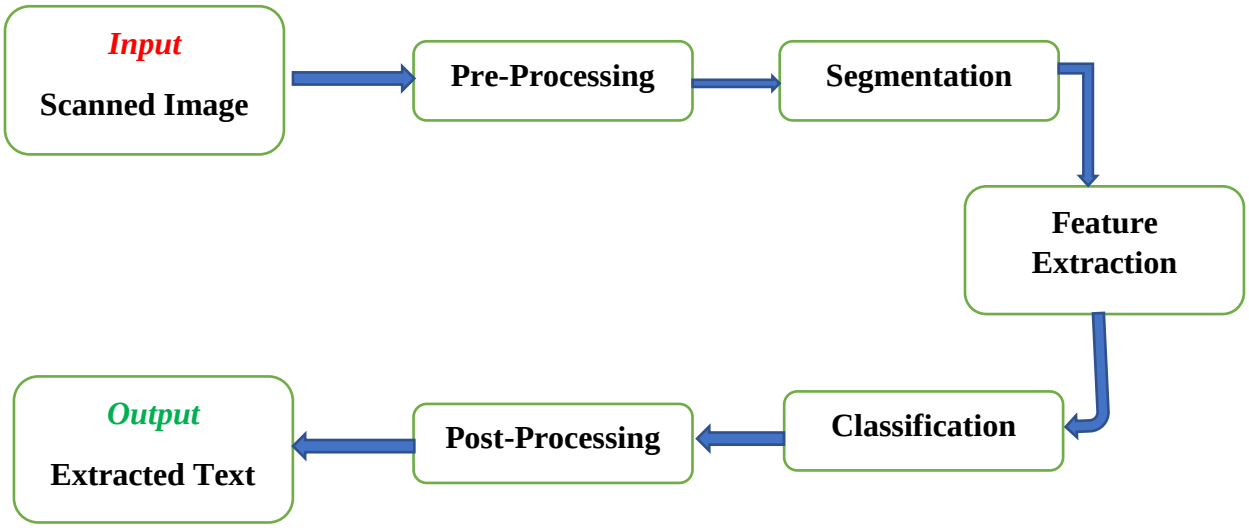


Figure 2: - Tasks in multilingual OCR system

Then document image segmented in to isolated characters. In feature extraction stage, the characters are further extracted according to the required dimensions. Then character is classified to its corresponding classes in classification stage. The last stage of recognition is post processing, in which characters are reconstructed with reduced error according to the original text.

3.2.1. Optical Scanning

In this stage hardcopy of document such as books, receipt, any paper records or target object like vehicle registration plate captured as image. In this stage of OCR, conversion of those objects to image by the help of scanner and digital camera.

3.2.2. Preprocessing

The importance of the preprocessing stage of a character recognition system lies in its capability to correct some of the problems that may occur due to some of the factors. Those factors are capturing device quality, resolution of capturing device, type of printed documents, quality of paper, font-size used in the text, and other [20]. Thus, the use of preprocessing methods may enhance a document image preparing it for the next stage in an OCR system. In order to achieve advanced recognition rates, it is important to have an effective preprocessing phase. Therefore, by means of effective preprocessing algorithms to makes the OCR system more robust mainly through correct image improvement, noise reduction, image thresholding, skew detection and correction, page segmentation, character segmentation and character normalization.

Preprocessing methods are needed on color, grey-level or binary document images holding text and/or graphics. Due to higher computation in preprocessing, in character recognition systems most of the applications use grey or binary images. Colored images may also contain non-uniform background and/or watermarks making it hard to extract the document text from the image without carrying out some kind of preprocessing. Therefore, the wanted result from preprocessing is a

binary image holding text only. Thus, to attain this, several steps are required, first, some image improvement techniques to eliminate noise or correct the contrast in the image. Second, thresholding to eliminate the background holding any scenes, watermarks and/or noises. Third, page segmentation to separate graphics from text, fourth, character segmentation to detached characters from each other and, finally, morphological processing to improve the characters in cases where thresholding and/or other preprocessing methods eroded parts of the characters or if there are additional pixels to them. The above techniques mentioned few of those which may be used in character recognition systems and in some applications. In several papers [4, 20-23] techniques of preprocessing are reviewed.

3.2.2.1. Binarization and Threshold

The Binarization method changes the grey scale image (0 up to 256 gray levels) in to black and white image (0 or 1) by the help of threshold. The high quality binarized image can yield more precision in character recognition as compared initial image because noise is present in the original image [24]. In fact, problem is that which binarization algorithm is suitable for all type images. The choice of most optimal binarization algorithm is hard, because different binarization algorithm gives different accuracy on different datasets. This is especially true in the case of old documents images with variation in contrast and brightness level.

The algorithms divide into two classes (i) Global Binarization method (ii) Local Binarization method. The global binarization methods used only one threshold value for entire image whereas the local binarization method where the threshold value designed locally pixel by pixel or region by region.

I. Global Methods

A. Fixed Thresholding Method

In Fixed Thresholding binarization method fixed threshold value is used to assign 0's and 1's for all pixel places in a given document image [24]. The basic idea for fixed binarization method is described in equation 3.1.

$$g(x,y) = \begin{cases} -1, & f(x,y) \geq T \\ x, & otherwise \end{cases} \quad 3.1$$

T is value of global threshold. For different threshold values T in fixed binarization methods the results are various. So, in order to get better result optimal threshold value is figured according to input image.

B. Otsu Binarization Method

In image processing, Otsu's thresholding method is used for automatic binarization level choice, from the shape of the histogram. Otsu's thresholding method contains iterating through all the probable threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels value that may falls in foreground or background [24]. The goal is to investigate the threshold value where the sum of foreground and background spreads is at its smallest.

STEPS for Otsu Method

1. Separate the pixels into two clusters according to the threshold by the following equation.

   $q_1(t) = \sum_{i=1}^{t} p(i)$ and 3.2

   $q_2(t) = \sum_{i=t+1}^{I} p(i)$ 3.3

   P represents the image histogram.

2. Find the mean of each cluster by equation (3.4) and (3.5).

   $\mu 1(t) = \sum_{j=1}^{t} \frac{ip(i)}{q1(t)}$ and 3.4

   $\mu 2(t) = \sum_{j=t+1}^{I} \frac{ip(i)}{q2(t)}$ 3.5

3. Calculate the individual class variance.

   $\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu 1(t)]^2 \frac{p(1)}{q1(t)}$ 3.6

   $\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu 2(t)]^2 \frac{p(1)}{q2(t)}$ 3,7

4. Square the difference between the means as follow.

   $\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = q1(t)[1 - q1(t)][\mu 1(t) - \mu 2(t)]^2$ 3.8

5. Finally, this expression can safely be maximized and the solution is t that is maximizing $\sigma_b{}^2(t)$

C. Kilter and Illingworth Method

The kilter method is used combination of Gaussian distribution to get threshold value. In kilter technique the thresholding value 't' is threshold that is used to segment the image into two portions, background and foreground [24]. Both of the portion modeled by Gaussian distribution, p_B (t) and $p_F(t)$, the $p_{mix}(t)$ mixture of these two Gaussian distributions as shown in equation (3.9).

$$\mathrm{p}_{mix}(t) = \alpha \mathrm{p}_B(t) + (1-\alpha)\mathrm{p}_F(t) \qquad 3.9$$

Where α is found by the portions of background and foreground in the image.

II. Local Methods

A. Adaptive Binarization Method

Adaptive binarization method is used in local binarization technique. In this a window of NxN blocks slide over the entire image and threshold value is calculated for each local area under the window for binarization. The adaptive method gives more precise output as related to global binarization in such conditions where the image produced from bad shading, small level resolution blurring, and non-uniform illumination [25]. In adaptive binarization any one of methods likes Niblack, Sauvola etc. used to calculate the local area threshold value.

B. Niblack Method

In Niblack method the threshold value for the local area under the window is calculated pixelwise [25]. The value of the threshold value is liable upon the local mean and standard deviation of window area. The threshold value is finding using following equation.

$$T_{Niblack} = m + k * s \qquad 3.10$$

$$T_{Niblack} = m + k\sqrt{\frac{1}{NP}\sum(pi - m)^2} \qquad 3.11$$

Where m is the mean of local area pixels of an image and s is the standard deviation of local pixel area. The value of k is fixed to -0.2 by the author.

C. Sauvola Method

The Sauvola algorithm [9] is an adapted form of Niblack algorithm shown. It gives more result than Niblack under such circumstances as light variation on document image, light texture etc. In the Sauvola modification, the binarization is given by:

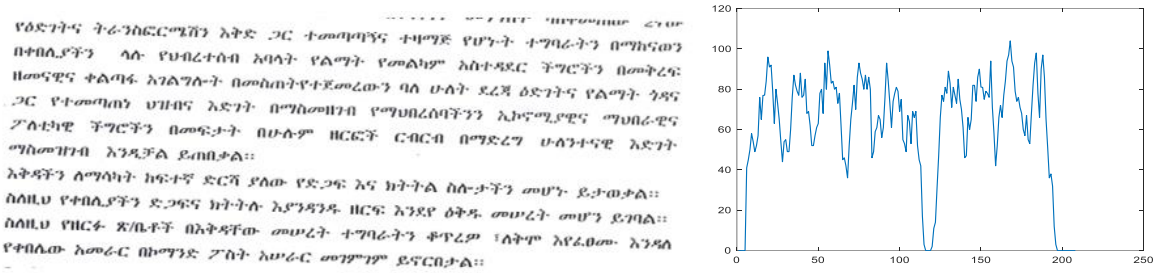
$$T_{sauvola} = m * (1 - k * \left(1 - \frac{S}{R}\right)) \quad 3.12$$

Where value of m is the mean of pixels under window area, value of S is the dynamic range of variance and the k parameter may be in the range of 0 to 1.

In general, global thresholding method is better approach for calculate the threshold values of a grey scale images. But it doesn't give good results for colored image and under intensity illumination. For such kinds of images local thresholding methods can be used [24].

3.2.2.2. Skew Detection and Correction

During document scanning, skew is inevitably added into the incoming text image. It is any deviations in orientation of the image from that of the input document, which is not parallel to the horizontal or/and vertical. Skew correction stage is one of the important parts in image preprocessing. Many techniques have been proposed by scholars for the finding of skew in scanned image documents. The skew correction is considered compulsory in pre-processing step, since it has a direct outcome on the consistency and efficiency of the segmentation and feature extraction step. Skew is normally introduced into the image at a time of scanning or during document writing. Leaving it as it is without modification will give incorrect outcomes during segmentation and recognition [26]. Figure 3 shows horizontal histogram of skewed document image.



(A) (B)

Figure 3: - Skew (A) Sample Skewed document image (B) Its horizontal histogram

As it was observed from Figure 3, line segmentation on skewed document, it considers more than one line as single line.

The text image which is captured by scanner or any other at an angle can be rotated to a correct place following a sequence of stages. As a starting point document image which is scanned in an abnormal direction is considered. The steps to be followed for correcting the skew to its normal position include [6]:

- Base line identification
- Skew angle correction

The baseline identification is normally the most significant step of the entire process. The line along which the center of gravity of the word hangs is called Baseline. A good approach is used in this algorithm where in the whole word is inscribed in a polygon with a minimum of two-dimensions. Its center of gravity is taken as a single line extending a certain angle with the horizontal. Angle is calculated for the purpose of getting the angle by which the image or document is rotated. It also shows the direction and angle by which is should be rotated for it to be a text in correct form.

Centroid: The centroid is usually said as the center-of-gravity (COG) or the "center-of-mass". The location of the centroid assuming the polygon to be made of an object of uniform density. After calculating the center-of-gravity point, a line is drawn linking the origin and the COG which is the essential called line of gravity. The

centroid for a rectangle is the point joining the diagonals which is considered as line of gravity for the polygon. Consider the document image in the Figure 4 which is rotated. Inscribe the text document in a rectangle by considering the outermost pixel called farthest pixel in the four directions and find the centroid. The line joining the centroid and the origin is the required baseline which let us to know the angle at which the image is rotated [26].

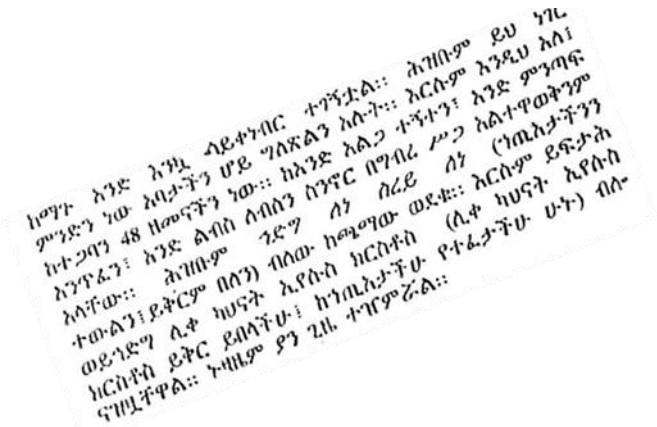


Figure 4: - Skewed input image

The steps to correct skew is shown in the following steps.

Step 1: Determine the farthest points in all the four directions. Figure 5 shows the scanned image farthest points.

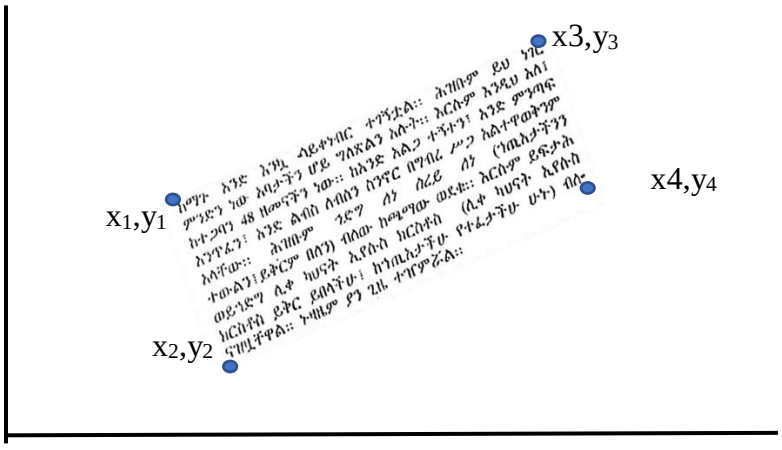


Figure 5: - The scanned images farthest points

Step 2: - Using those four points get centroid. So, the previous four points showing the polygon corners and the polygon center (COG), can be calculated by using the equations given bellow

$$C_x = \frac{1}{6A}\sum_{i=0}^{N-1}(X_i + X_{i+1})(X_iY_{i+1} - X_{i+1}Y_i) \quad 3.13$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{N-1}(Y_i + Y_{i+1})(X_iY_{i+1} - X_{i+1}Y_i) \quad 3.14$$

The Figure 6 shows the COG.

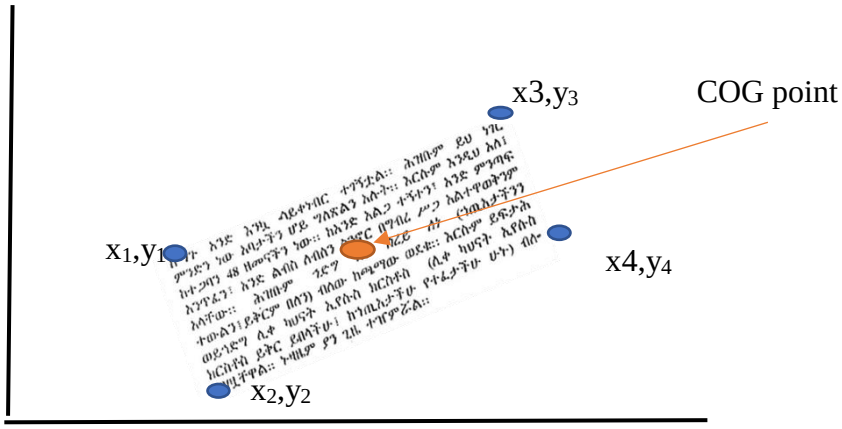


Figure 6: - The scanned images COG

Step 3: To get the baseline, join the centroid to the origin, as shows in Figure 7.

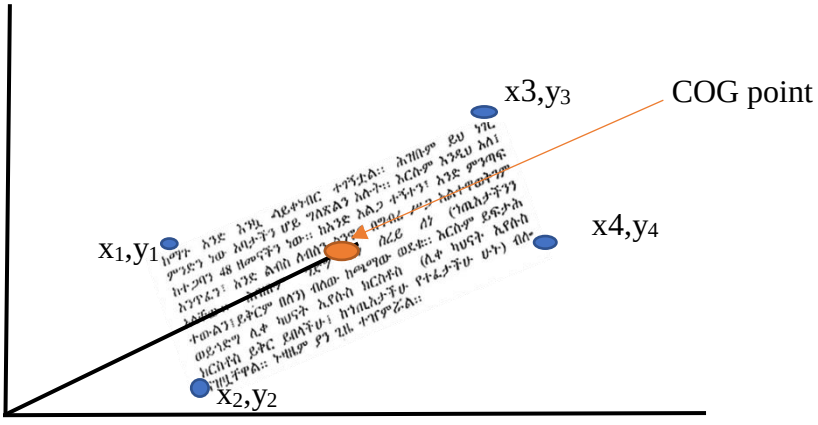


Figure 7: - The scanned images Baseline and COG

Step 4: - From formed baseline calculate the skew angle as shown in Figure 8

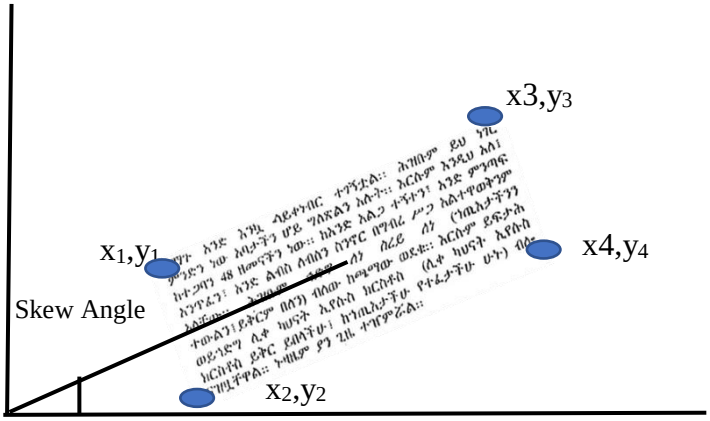


Figure 8: - The skewed angle detection

Step 5: Rotate the document in the reverse direction (clockwise direction) by the skew angle.

3.2.2.3. Noise Removal

A common problem occurs while scanning documents is noise which can arise in an image because quality of object to be scanned, the quality of typing machine used, or it can be due to scanners itself during the scanning process. It is one of the important steps in preprocessing stage. Like other stage, noise decreases the accuracy of subsequent job of OCR systems. It can exist in the foreground or background of scanned image and can be created before or after scanning. Some examples of noise appear on scanned document images are as follows [27].

The page rule line noise is a source of noise which interferes with text or characters. Those lines in document can cause the resulting challenges, (i) the ruled lines interfere with and linking to the text; (ii) inconstant widths in the ruled lines result problems in the noise removal algorithms; (iii) broken ruled lines cause problems for algorithms detecting them; iv. Some letters, for example 'ፐ', which have horizontal lines, are omitted by the algorithms as they are unable of noticing variances between those characters and the ruled lines. Numerous approaches have been proposed for ruled line elimination. The methods can be categorized into three main

categories. First, there are mathematical morphology-based techniques that depend on prior knowledge and the second category contains methods which use Hough Transform to get text features and to find lines in all direction. The last use Projection Profiles to evaluate lines and hence, reduce the difficulty's dimensions, which then improves the precision of the first step in some methods of noise removal.

The marginal noise generally appears in a big dark region around the document image. Methods to eliminate marginal noise can be classified into two group: - searching noise components and finding text components [27].

Clutter noises found in an image because of document image rotate while scanning or are from holes punched found in the document. Clutter often hits or overlays on some parts of the text which reduces segmentation and recognition correctness in OCR systems.

Background noise is noises such as irregular contrast, display through effects, interfering strokes, and background spots, etc. Historical manuscripts and captured document images often have degradations like uneven contrast, show through effects, interfering strokes, background spots, humidity absorbed by paper in different areas, and irregular backgrounds [27].

3.2.2.4. Thinning

Thinning is an image preprocessing stage performed to make the image crisper by reducing the binarized image regions to lines that approximate the skeletons of the region. Thinning cleans the image so that only compact amount of data needs to be handled in the next image processing stage.

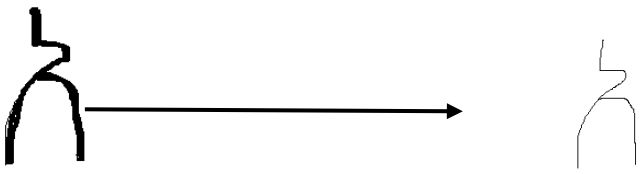


Figure 9: - Thinning Process

The skeleton of the shape is the product of the thinning "Skeletonization", which means peeling the contour of the text until it get the most medial one-pixel width [28].

A good thinning algorithm is one that can ideally tin data; reduce local noise without presenting distortions of its own. But the main goal is to hold most important features of the shape. The two categories of thinning algorithms [29], iterative and non-iterative .

The iterative approach is classified into sequential or parallel methods by eliminating the contour pixels iteratively until they reach one-pixel width. Sequential and parallel thinning methods are similar in getting the required and none required pixels, while different in removing time. In sequential, the removal of none required pixels starts in identifying the required process while in parallel the pixels are removed after identifying all none required pixels [30].

Non-iterative method gives a skeleton directly without studying all of the individual pixels. Several methods have been appointed to extract the skeleton using a non-iterative method, such as neural networks, Voronoi diagrams and wavelet transforms [30].

The proposed procedure involves of three main phases: conditional contour selection, pixel removing, and one-pixel width stage. The structure of these stages is addressed in the flowchart as shown.

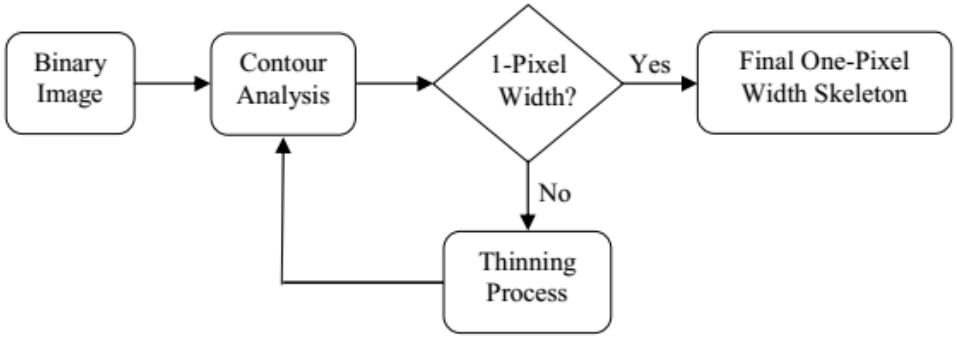


Figure 10: - Flowchart of thinning source [30]

The contour is flagged where any foreground pixel '1' is found with only one-pixel background '0' and taken as contour pixels. In case of the flagged pixels being adjacent either vertically or horizontally, it will cause desertion in removal process, or stork will vanish. To eliminate this problem, a single side of flagged pixels is frequently appearing as foreground '1'. Before going to the thinning process phase, the foreground pixels sited in corner locations are flagged as contour pixels.

3.2.3. Segmentation

Document segmentation is the process of isolating a document into its base components (lines, words, characters). Once the regions (text and non-text) have been known, the segmentation of the text elements can start. Numerous encounters exist which need to be done out in order to segment the component properly. For line segmentation, touching, fragmented, or overlapping text lines normally happen. After segmenting line, it is processed to further slice it into characters. The same difficulties like touching and broken elements occur for characters. Segmentation of document image into discrete character or digit is significant phase in document analysis and character recognition.

For segmenting text from a given input document image, it is very essential to know all the probable text areas [31]. There are numerous influences that hamper the process of text based image segmentation [32]. The quality of the image is an important issue for text segmentation. All Text image segmentation can be achieved at three levels [32, 33]. As we move at different stages of text segmentation hierarchy, we obtain definitely better details. Using all the three stages is not obligatory.

Step 1: Line Segmentation

The global horizontal projection method computes summation of all black pixels on all rows and do equivalent histogram [31, 32, 34]. From the peak/valley points on the histogram, discrete lines are separated. In line segmentation the purpose is to

draw of one upper horizontal line and one lower horizontal line for each line of text image. The steps for line segmentation are as follow:

- Construct the Horizontal Histogram for the image
- Using the Histogram, find the points from which the line starts and ends.
- For a line of text, upper line is drawn at a point where we start getting black pixels and lower line is drawn where we start getting nonappearance of black pixels. And the process continues for next line and so on.

Step 2: Word Segmentation

The next level of segmentation is word segmentation is the. It includes vertical skimming of the image, pixel-row by pixel-row from left to right and top to bottom [32, 33].

- Find the vertical histogram for each segmented line
- Using the vertical histogram, find the points from which the word starts and ends.
- Vertical lines are drawn at starting and ending points for each word.

Step 3: Character Segmentation

Character segmentation is the final level for text-based image segmentation. The segmentation of document image in to individual characters is done as follow.

- Draw the horizontal histogram for each segmented line
- From the horizontal histogram, find the row which consists of highest value.
- The row which consists of highest value of black pixel for each line is really the row which consists of header line.
- Draw the vertical histogram for each segmented word in above of header line

- Using the histogram, get the points from which the character starts and ends.

3.2.4. Feature Extraction

The aim of feature extraction is to capture the important characteristics of the symbols. Since it is the most significant and vital step of the recognition process, choice of the feature extraction technique becomes vital influence in attaining the high recognition performance [35]. It can be thought to be one of the hardest problems of pattern recognition. A number of methods are implemented in feature extraction some are of them as: moment, zoning, projection histogram, n-topples, crossing and distance [36].

Moment: - In this case the moments of different points present in a character are utilized as a feature.

Zoning: - The frame of character is divided into several overlapping and no overlapping zones and the densities of object pixels in each zone are identified and the density is identified as a no of object pixel in each zone divided by total no of pixels [36].

Projection histogram: - Projection histogram gives a no of black pixels in the vertical and horizontal directions of the specified character. It may be vertical, horizontal and left diagonal or right diagonal [36].

N-topple: - In this type the position of black and white pixels is considered as feature. This technique delivers the no of significant properties of the pixels.

Crossing and distances: - In this type, features are obtained from counts the character image is crossed by vector in certain directions or at certain angles.

3.2.5. Classification

Classification is the process of setting the identified data to their corresponding category class with respect to groups with similar characteristics, with the aim of

discriminating several objects from each other within the given image. It is carried out on the basis of kept structures in the feature space, such as structural features, global features etc. We can say that it classifies the feature space into several classes based on the decision rule[35]. Some classification methods used in earlier developed OCR systems are Neural Network, Support Vector Machine, K-Nearest Neighbors, Bayesian Classification, Decision Tree Classification etc.

3.3. Convolutional Neural Networks

Convolutional Neural Networks are much similar to usual Neural Networks. They consist of neurons that have learnable weights and biases. Each neuron accepts some inputs, computes a dot product and optionally follows it with a non-linearity. The entire network still expresses a single differentiable score function from the raw image pixels on one end to class scores at the other and they still have a loss function (e.g. Softmax) on the last fully-connected layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

The architecture of a CNN is intended to take benefit of the 2D structure of an input image. This is attained with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another advantage of CNNs is that they are less complicated to train and have many fewer parameters than fully connected networks having similar amount of hidden units [37]. A CNN consists of several layers. Implicit explanation about each of these layers is given below.

I. Convolution Layer (Conv Layer)

The Conv layer is the core building block of a Convolutional Neural Network. The primary importance of this layer is to extract features from the given input image. The Conv Layer parameters consist of a set of learnable filters (kernels or feature detector). Filters are used for recognizing patterns all over the input image. Convolution works by sliding the filter over the input image and then we get the dot product between the filter and part of the input image.

II. Pooling Layer (Sub-sampling or Down-sampling)

Pooling layer reduce the size of feature maps by the help of different functions to summarize sub-regions, such as getting the average value or the maximum value. Pooling works by sliding a window over all the input and providing the content of the window to a pooling function.

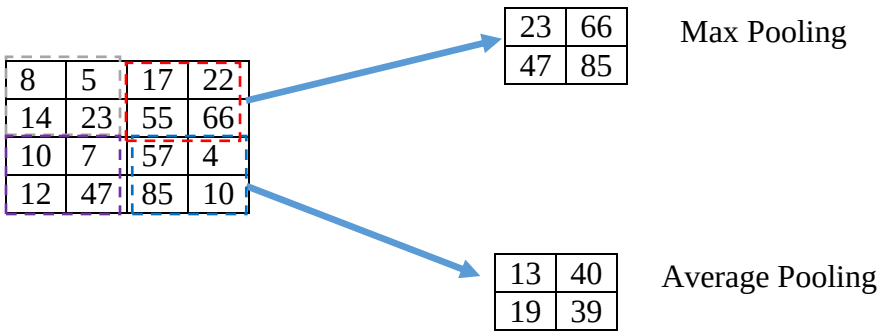


Figure 11: - Max-Pooling and Average-Pooling

The purpose of pooling is decreasing the number of parameters in our network why it is called down-sampling and making learned features more robust by making it more invariant to scale and changes of orientation.

III. ReLU Layer

It stands for Rectified Linear Unit and it is a nonlinear operation. It is an element wise operation (applied per pixel) and replacing every negative pixel value in the feature map by zero.

Output = Max (zero, Input)

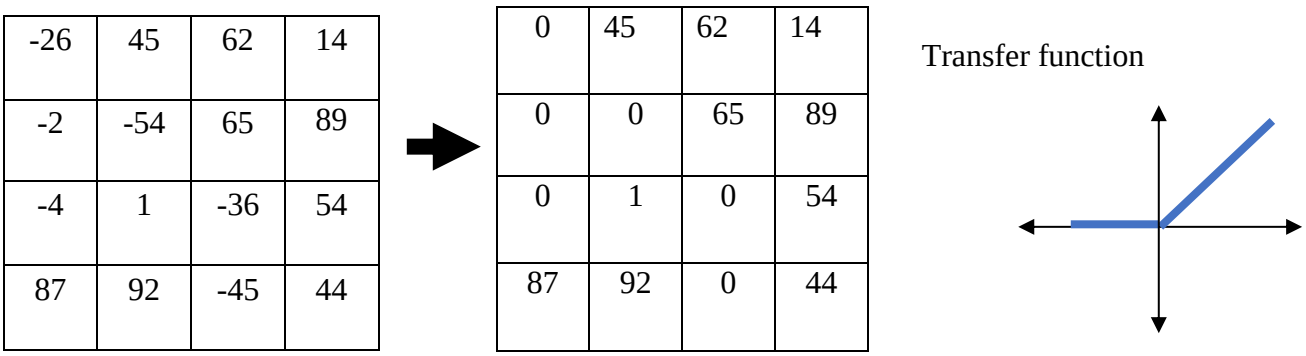


Figure 12: - Example showing ReLU operation

Since most of the real-world data we would need our ConvNet to learn would be nonlinear, it introduces nonlinearity in our ConvNet. Other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been seen that it performed better in most cases [38].

IV. Fully Connected Layer

The Fully Connected layer is arranged exactly the way its name indicates. It is fully interconnected with the output of the prior layer. A fully connected layer takes all neurons in the prior layer (be it fully connected, pooling, or convolutional) and connects it to each single neuron it has.

Adding a fully-connected layer is also an inexpensive way of learning non-linear combinations of these features. Most of the features learned from convolutional and pooling layers may be nice, but addition of those features might be even more and more.

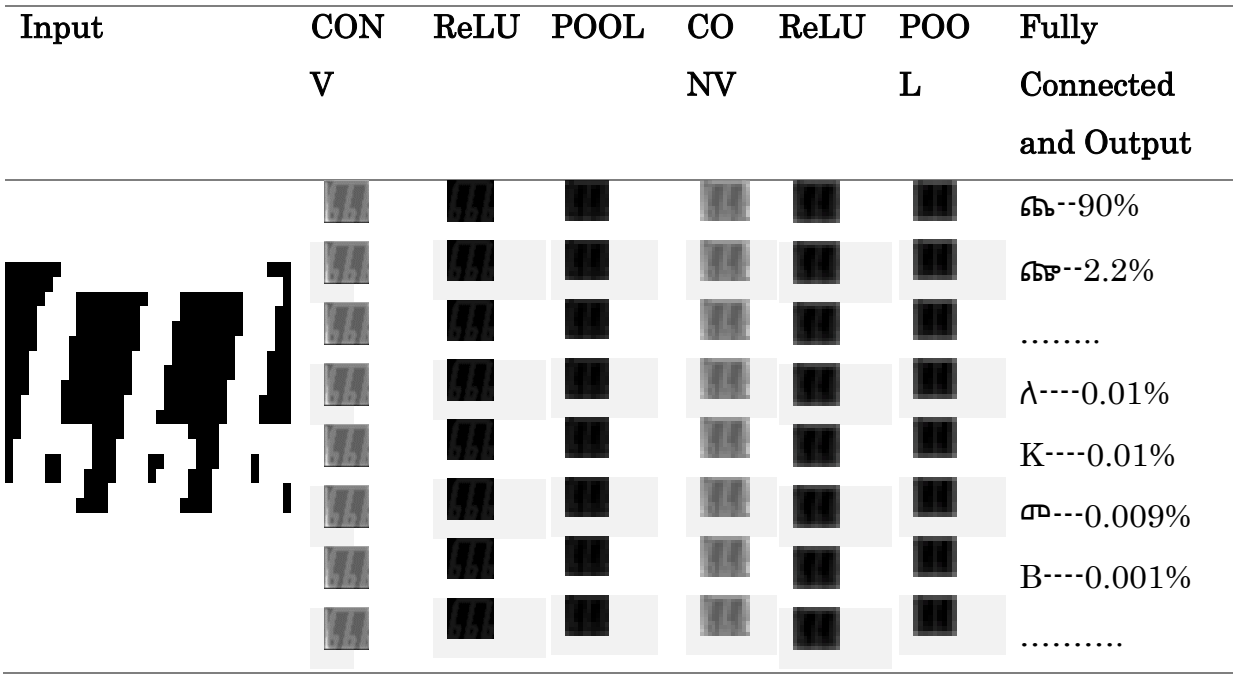


Figure 13: - Layers in CNN

3.4. Challenges in Multilingual and Monolingual OCR

For better quality and higher accuracy character recognition, OCR procedures expect high quality or high-resolution images with some basic structural properties such as high distinguishing text and background. The way images are produced is significant and decisive factor in the accuracy and success of OCR. Since this often affects the quality of images dramatically. Frequently OCR with images captured by scanners gives better accuracy and better performance. In contrast, images produced by cameras are not as good of an input as scanned images to be used for OCR due to the external or camera associated factors [39]. Many problems might emerge, which are explained as follow.

Scene Complexity: - In our environment, we can observe large numbers of manmade objects which are involved in camera taken images such as paintings, construction, and symbols. These things have comparative structures and physical appearances to text which makes text recognition very challenging in the treated image. Text itself is frequently placed out to encourage decipherability. The challenge with scene

intricacy is that the surrounding scene made it difficult to segregate text from non-text [40].

Conditions of Uneven Lighting: - Frequently capturing images in natural environments results in irregular lighting or shadows or darkness. This poses a challenge for OCR as it reduces the required characteristics of the image and it make less accurate segmentation and recognition results [40].

Skewness (Rotation): - In OCR systems, the direction for the input image that taken from camera of hand-held device or other gadgets that used for taken image is not fixed like a scanner input, which skewing of text lines from their unique orientation might be observed. Great degree poor results will be observed when such a skewed image is fed to the OCR classifier [39]. Numerous methods exit for the purpose of removing skew on the image documents, such as Projection Profile method, RAST algorithm, Hough transform, methods of Fourier transformation, etc.

Blurring and Degradation: - Since working over a variety of distances are intended to numerous digital cameras, significant factor is the digital camera's focusing. At large apertures and small distances, irregular focus can be seen when a small point of view changes. For the most part associated with photography, there are two type of obscure which is: out of focus obscure and movement obscure [41].

Aspect Ratios: - Text on image may have dissimilar aspect ratios. Text may be brief such as traffic signs, while other text may be much stretched, such as video captions. Location, scale and length of text need to be considered with search procedure to sense text, which make high computational complexity [39].

Tilting (Perspective Distortion): - Document images captured by scanners are constantly parallel to the plane of sensor, but this cannot be observed all times for recorded picture obtained by a portable camera, that may not generally be parallel to the form plane. Accordingly, lines of text that distant from the camera seem littler than those that nearer to the camera which seems greater. Such condition introduces tilted pictures. Observation of a perspective distortion is clear if the

recognizer is not perspective intolerant, which causes lower recognition rate and accuracy [42].

Fonts: - Italic style and script fonts of characters might overlap each other, making it difficult to make some of the main OCR processes such as segmentation. Characters of various fonts have large within class variations and form many pattern sub spaces, making it hard to get accurate recognition when the character class number is large like Amharic characters was number of characters are more than 300.

Multilingual Environments: - even though a large percentage of the languages have many characters, languages for example, Amharic, Japanese, Chinese and Korean have a large number of character classes. In multilingual environment the combination of those languages lead to larger number of character classes. In multilingual situations, OCR in scanned documents stays as a primary research issue [43], since OCR in complex symbolism is more difficult.

Warping: - Content or text on objects of varying geometries can be another challenge for OCR to be recognized when images of such condition captured by handheld cameras. A few circumstances may emerge with flatbed scanners, wherein the twisted text observed when the content procured on picture, for example the content towards the binding of an extremely thick book. For convention paper documents, a methods for image dewarping is proposed by Ulges et al.[44]. By expecting the way that content lines are equally separated and parallel to each other, they dewarp pictures.

CHAPTER FOUR: DESIGN AND IMPLEMENTATION

The proposed research work has a lot of task from data acquisition to training neural network and recognition. In this chapter methodology used in the proposed system, implementation algorisms and tools used to implement the system were discussed.

4.1. Review of Proposed Method

The proposed method for bilingual script character recognition from printed Amharic and Latin script containing document image consist a series of tasks such as scanning the document image, binarization input image, skew removal, line removal, line segmentation, character segmentation, feature extraction and classification of the script to their appropriate class label. Figure 14 shows the proposed OCR model.

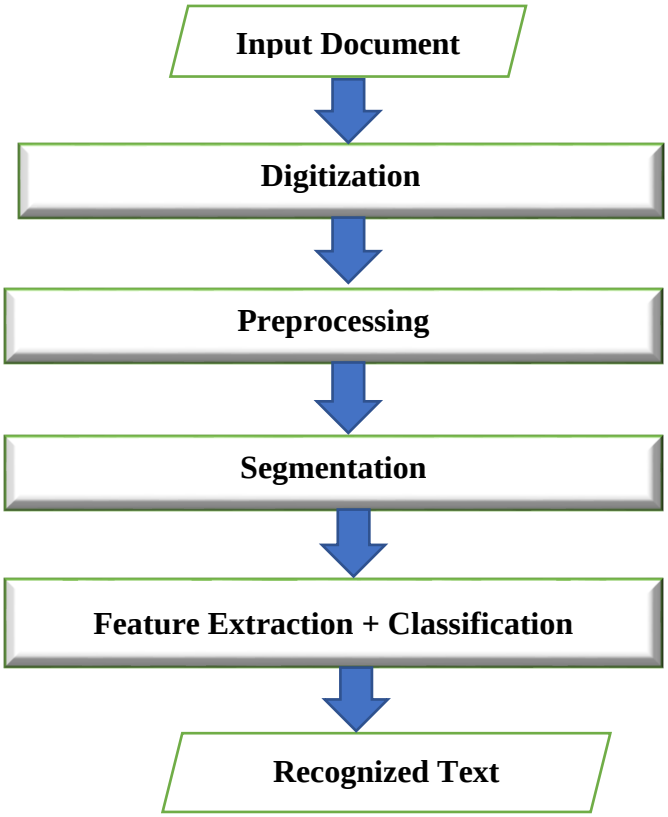


Figure 14: - Flow chart of proposed OCR system

The full OCR system experimented in two stages. First, experimentation done on preprocessing and character segmentation stage. Second experimentation done on feature extraction and character classification.

Digitization is process of changing document text to digital or machine-readable formats. Scanners and digital cameras were used to capture object and fed for preprocessing stage. Documents has collected from different sources containing Amharic and/or Latin scripts and it has been scanned with scanner with resolution of 200dpi, 400dpi, 600dpi and 1200dpi and captured with digital camera as well. Source of document is from real documents such as newspapers, fictions, books, letters, journals, different religious document and noisy and noise free document prepared from different font type and font size.

4.1.2. Preprocessing

In the pre-processing phase, there are a series of tasks done on the captured input images. It enhances the image to be appropriate for segmentation. Preprocessing is an essential stage prior to character segmentation since it controls the suitability of the results for the successive stages.

There are a number of issues that disturb the accuracy of text recognized. These factors include: scanner quality, scan resolution, type of printed documents (photocopied or laser printer), paper quality, fonts used in the text, linguistic complexities, and dictionary used. Foxing and "text show through" found in old paper documents, watermarks and non-uniform illumination are examples of problems that affect the accuracy of OCR compared to a clean text on a white background [41].

4.1.2.1. Binarization

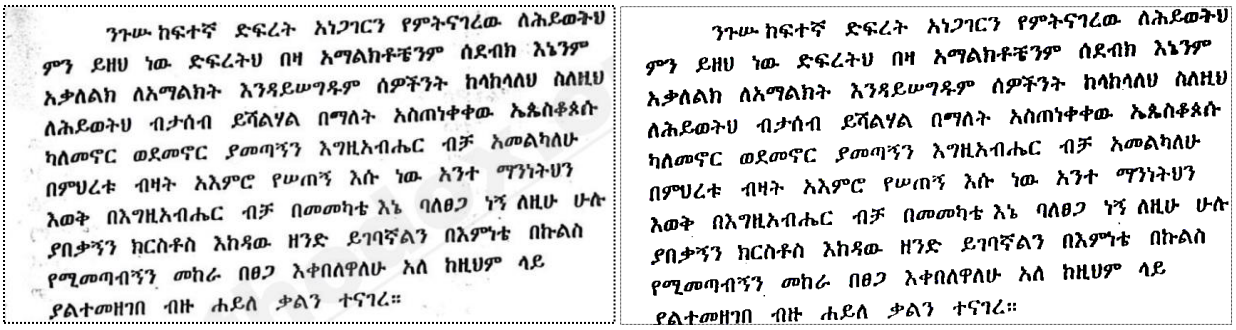
In this phase, a greyscale image is changed in to binary image containing only pixel values of ones and zeros. The binarization is used to change the pixel values of area of interest (here greatly characters or text) to higher value and to make all other

part zero by fixing a threshold value. Pixels lower than threshold value are changed to zero whereas pixels greater than threshold changed to one.

There are two main methods for binarization based on the threshold values used. Global binarization and local binarization thresholding [24, 25]. In global thresholding method, the pixels of the image are classified into text or background according to a global threshold that is determined by considering gray level pixel values of the entire image. Usually, such method is faster. This method fails to remove non-uniformly distributed noise in the image. Local thresholding techniques, classify the pixel of the image into text or background according to a local threshold determined by their neighborhood area. Such techniques are more robust to the presence of different kinds of noise in the image. On the contrary, they are significantly more complex and time-consuming.

In the year 1979 Otsu proposed the highest-class variance method known as Otsu method. For the easy computation consistency and effectiveness this method is extensively used [45]. For this research work Otsu method was used for binarizing input images.

In Figure 15 (A) shows scanned grey scale text image with noise and (B) shows image after binarization.



(A) (B)

Figure 15: - Binarization A. Gray scale image B. Binarized image using Otsu techniques

4.1.2.2. Skew Removal

A document originally may not have skew, but when a page is scanned or replicated, nonzero skew may exist. An important stage of any document recognition system is finding of skew in the image of a page [18, 26]. The projection profile can be used as a suitable feature for skew detection [18]. There are three kinds of skew in the images document [6] they are: Single Skew, Multiple Skew, Non-uniform text line Skew

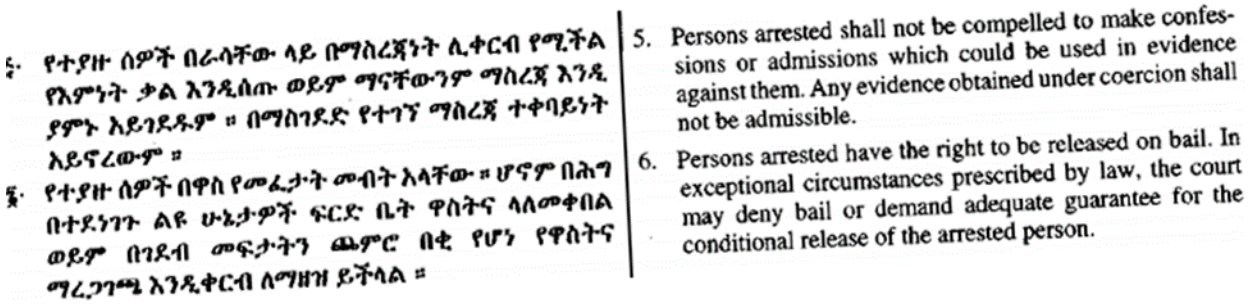
Single Skew: - In single skew whole parts of documents are skewed to single angle. Most of the document images have this kind of skew. A lot of work has been done in this field and lot of research continues to be happening.

Multiple Skew: In this, scanned document will have several parts; each may be rotated to totally dissimilar angles. Sensing such variety of skewness needs lot of efforts. Multiple Skew issues are present, but it has not got a much consideration from researchers [6].

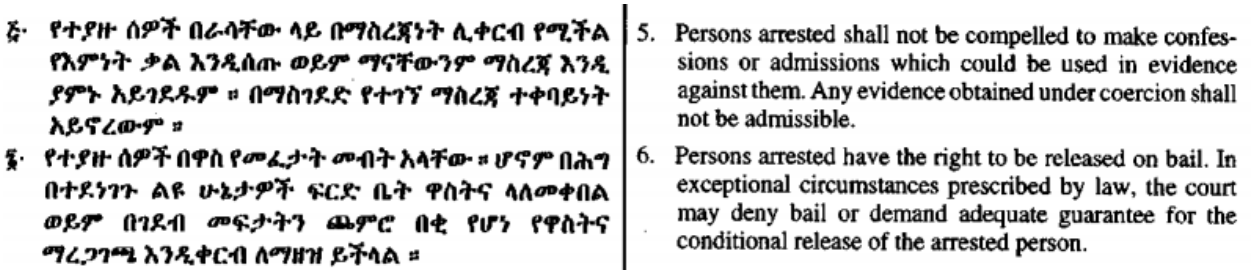
Non-uniform text line skew: Once documents having many directions within the single line is named as non-uniform text line skew. In non-uniform text line skew there are many orientations in every line of the document.

Many methods have been proposed by many researchers for the detection of skew in binary image documents [46]. The majority of them are based on Projection file, Fourier transform, Hough transform. Main advantage of Hough transform is its accuracy and simplicity. Even though Hough transform is slow in speed due to its good accuracy the proposed system used it to detect and correct skew.

In case of single skew Hough transform method used to deskew images back to normal. Figure 16 shows sample image before and after single skew correction.



(a)



(b)

Figure 16: - Skew removal (a) Skewed image (b) deskewed image

During scanning double page of books specially when the size of book is large, in a given line multiple skew angle exit as shown in Figure 17. This type of skew can be classified to non-uniform text line skew type of skew.

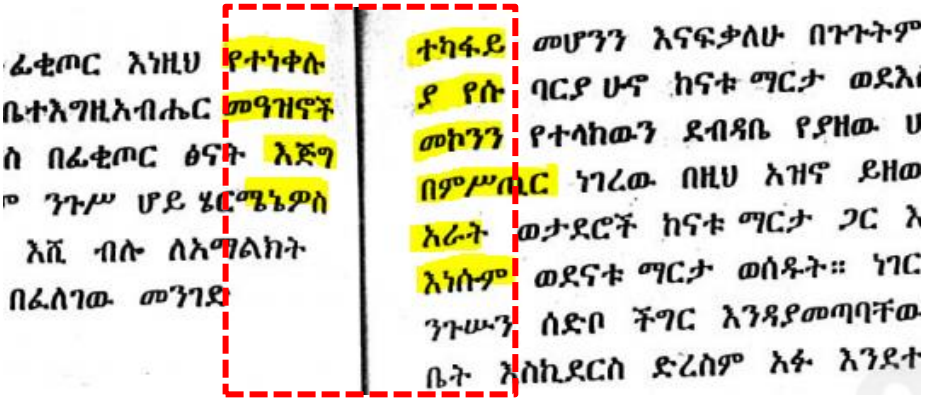


Figure 17: - Document image with non-uniform text line skew

In case of single skew like in Figure 16(a) removing skew is possible by using Hough Transform technique. But in case of non-uniform text line skew, removing with

proposed method is not possible. To overtake problems with this type of skew, character images having skew angle between -5^0 to $+5^0$ were included in training dataset. To do so, randomly selected 10% of dataset augmented by rotating with random angle between -5^0 to $+5^0$.

4.1.2.3. Line Detection and Removal

A line can be denoted as $y = mx + c$ or in parametric form, as $\rho = x\,cos\theta + y\,sin\theta$. Where ρ is the perpendicular distance from line to the origin, and θ is the angle created by this perpendicular line and horizontal axis measured in anticlockwise. That direction differs on how you represent the coordinate system as shown in the Figure 18.

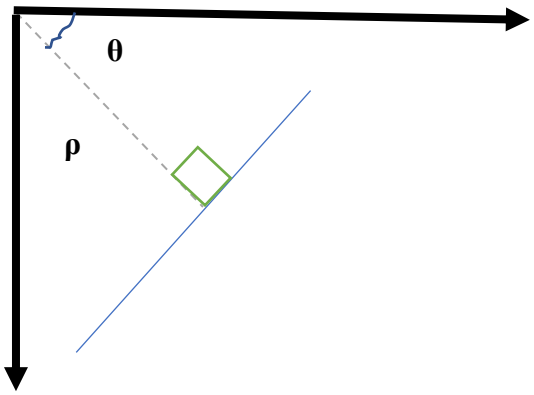


Figure 18: -Representation of coordinate system

So, if line is going below the origin, it will have a positive ρ and angle less than 180. If it is passing above the origin, in place of taking angle greater than 180, angle is taken less than 180, and ρ will be negative. At all vertical line will have zero degree and horizontal lines will have 90 degrees.

Hough Transform is a popular method to identify any shape [47], if you can denote that shape in mathematical format. It can identify the shape even if it is broken or distorted in some extent. Any line can be denoted by the following two terms, $(\rho\,, \theta)$. So, first it creates a two-dimensional array or accumulator to hold values of two parameters and it is set to zero firstly. Let rows is denoted by ρ and columns is denoted by the θ. Size of the array dependent on the accuracy you require. Assume you need the accuracy of angles to be one degree, you need 180 columns. For ρ, the

possible maximum distance is the diagonal length of the image. So, taking one-pixel accuracy, number of rows can be diagonal length of the image. Consider a 100 x 100 image with a horizontal line at the mid-point. Use the first point of the line. You identify it's (x, y) values. Now in the line equation, place the values θ = 0, 1, 2, …., 180 and check the ρ you got. For each pair, you increase value by one in the accumulator in its corresponding (ρ , θ) cells. Therefore, now in the accumulator, the cell (50,90) = 1 along with some other cells [48].

All thing explained above is summarized in the OpenCV function, **cv2.HoughLines()**. It just returns an array of (ρ , θ) values where ρ is measured in pixels and θ is measured in radians. The first parameter is input image which should be a binary image. Therefore, before finding Hough-transform finding threshold or use canny edge detection is required. The Second and third parameters are ρ and θ respectively. Fourth parameter is the *threshold*, which means smallest vote it should get for it to be taken as a line. Do not forget, number of votes depends upon amount of points on the line. Therefore, it represents the minimum length of line that should be detected as shown in the equation (4.1)

$$\mathbf{lines} = \mathbf{cv2.HoughLines(edges, 1, np.pi/180, 200)} \qquad 4.1$$

4.1.3. Segmentation

In this stage image containing textual document are segmented into individual characters. First the images are segmented in to separate line using horizontal projection histogram. Then each segmented line is farther segmented to characters by the help of vertical projection.

A. Line Segmentation

To correctly distinguish probable line segments from document image we used Y histogram projection. Since document image was preprocessed and free from skew by removing skew, line segmentation was done correctly as shown in the Figure 19(c). Procedures for line segmentation are as follow. First read pre-processed

images, then count the black pixel in each row, then construct the horizontal histogram for the image. Using the Histogram, find the rows containing white pixel (space line)[34]. Using the start and end line of white pixel, mark the bounding box for text. The procedure starts from top of image to down up to the end of the image.

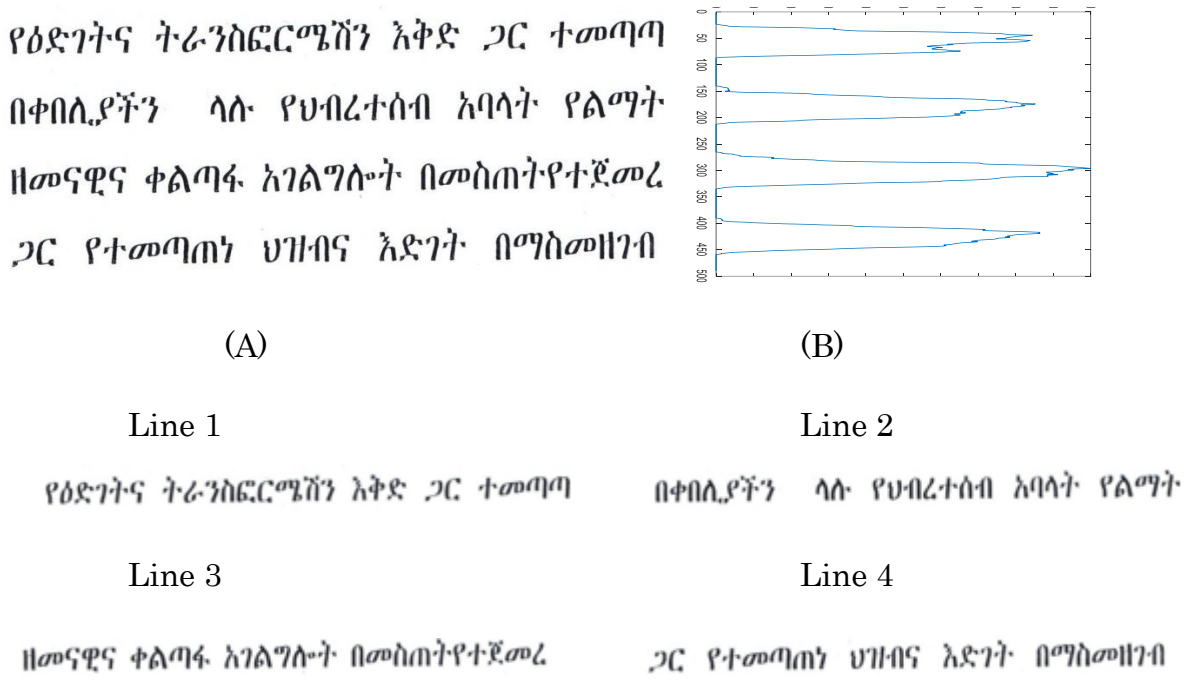


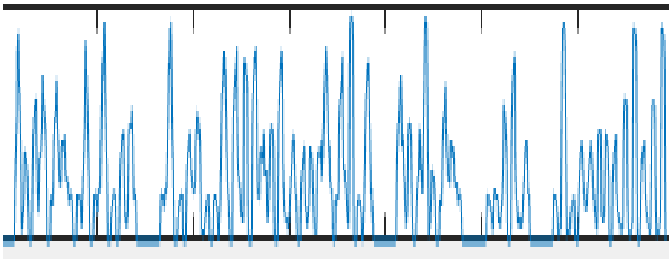
Figure 19:- Line segmentation using Y-histogram A) Input image B) horizontal projection Histogram C) Segmented lines

B. Character Segmentation

After line segmentation from top to down, each line segmented from left to right into individual character. To segment line to individual characters, first count the black pixel in each column from vertical histogram. Using the histogram, find the column containing white pixel which separates the characters. Mark the bounding box for characters using the single pixel [34]. Copy the pixels in the bounding box and save in separate file as .bmp file type. The procedure is repeated for all segmented line. Figure 20 (A), (B) and (C) shows image of input segmented line, its vertical histogram and segmented characters respectively.

የፅድገትና ትራንስፎርሜሽን አቅድ ጋር ተመጣጣ

(A)



(B)

የ ፅ ድ ገ ት ና ት ራ ን ስ ሜ ሽ .....

(C)

Figure 20: - Character segmentation A) Segmented line B) Vertical projection histogram C) Segmented Character

4.1.4. Feature Extraction and Classification

For proposed system Convolution Neural Network (ConvNet, CNN) was used for feature extraction and classification purpose. CNN is a powerful type of neural network that is used primarily for image classification. It was first designed by Geoffery Hinton, one of the innovators of Machine Learning [49]. Their location invariance makes them ideal for sensing objects in various positions in images.

A CNN contains of an input layer, many hidden layers and an output layer. The hidden layers of a CNN naturally consist of convolutional layers, pooling layers, ReLU Layer and fully connected layers.

Convolutional layer: - The layer is the main/core building block of a CNN. The layer's parameters have of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input

volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

Pooling Layer: - Another important concept of CNN is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common and it was implemented for proposed system. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in CNN architecture. The pooling operation provides another form of translation invariance.

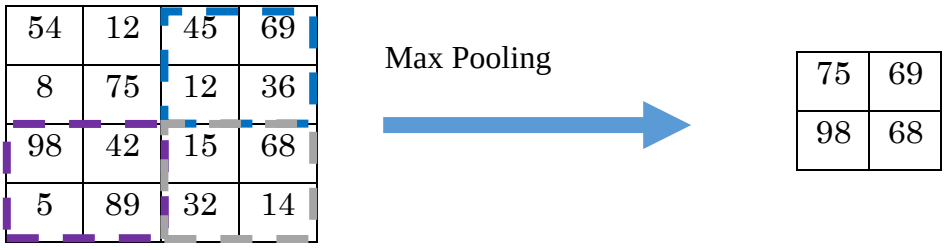


Figure 21: - Max pooling with a 2x2 filter and stride = 2

ReLU layer: - ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function $f(x) = max(0, x)$. It increases the nonlinear

properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase non-linearity, for example

The saturating hyperbolic tangent

$$f(x) = tanh(x), f(x) = tanh(x) \quad 4.2$$

The sigmoid function

$$f(x) = (1 + e - x) - 1 \quad 4.3$$

ReLU is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalization accuracy.

Fully connected layer: - Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

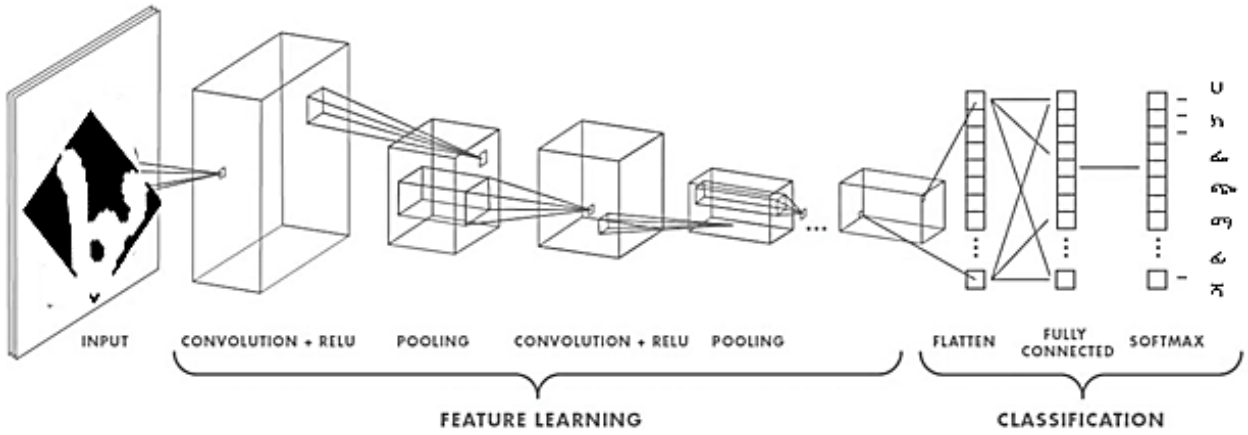


Figure 22:- CNN Layers

4.2. Implementation

To implement this bilingual OCR system, several libraries and tools have been used.

4.2.1. Development Tools

Programing Language

Python 3.5: - Python is a general-purpose programming language. Due to its simplicity and code readability, it is widely used. It allows the programmer to write ideas in less lines of code without losing its readability.

4.2.2. Libraries and Tools

TensorFlow: - Is the second machine learning framework that Google developed and used to design, build, and train deep learning models [50]. TensorFlow library can be used to perform numerical computations, which in itself doesn't seem all too special, but these computations are completed with data flow graphs. On graphs, nodes denote mathematical operations, while the edges denote the data, which usually are multidimensional data arrays or tensors that are communicated between these edges. According to [50] the name "TensorFlow" is found from the actions which neural networks do on multidimensional data arrays or tensors. TensorFlow layers module provides a high-level API that makes it simple to build a neural network. It gives ways that enable the creation fully connected layers and convolutional layers, adding activation functions, and applying dropout regularization.

NumPy: - It is the essential library for scientific computing in Python that it gives a good performance multidimensional array object, and tools for working with these arrays. It is the essential package for scientific computing with Python programing language. It consists a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities etc.

OpenCV-Python: - OpenCV is a library of programming functions primarily intended at Real-time computer vision. OpenCV-Python makes use of NumPy, which is a highly improved library for numerical operations and every OpenCV array structures are converted to and from NumPy arrays [51].

4.2.3. Experimentation Setup

System development as well as experimentation was done on personal computer with Processor of Intel® Core (TM) i5-4210M @2.6GZ, RAM of 8GB and operating system of Windows 10 pro. To capture documents Epson L382 scanner and digital camera of 10 MP are used.

4.3. Dataset Preparation and Experiment

Data was collected from different source such as books, newspaper, letters and different religious documents. Collected documents were scanned with scanner at different resolution level ranging from 200dpi to 1200 dpi, then preprocessed and segmented. The dataset was prepared for all Amharic core characters (231 characters) and all Latin script characters (52 characters with capital letter and small letter taken as a class). For each character classes 160 characters are prepared.

In addition to above dataset, about 10% of dataset are randomly selected from each class and treated with noise and 10% of dataset augmented by rotating with random angle between -5^0 to $+5^0$ to overcome problem related to non-uniform type of skew. As shown in table 5 total character images for all 257-character classes are 49,087-character images.

Table 5: - Total Original and Augmented dataset

	Original	Augmented		Sub Total
		Noise treated (10%)	Rotated (10%)	
Training	32,896	3,084	3,341	39,321
Test	8,224	771	771	9,766
Total				**49,087**

As these data from dataset has different image size, so it was a bit challenging to train them as we used fixed image size which is 32*32 pixel for each image. Therefore, image pre-processing is needed for handling this big size of data. For resizing images, we used Python Imaging Library (PIL). Anti-aliasing filter is used so that quality of the image doesn't degrade while resizing.

After setup the train environment, we convert images into NumPy and save these images in separate class file. Then images of all class are shuffled to have random validation and training dataset. Also merge them into a single dataset. Then we created a sanitized test and valid datasets.

4.4. Model Architecture and Implementation

The system architecture consists of several layers of CNN. As shown in Figure 23, the preprocessed data with a size of 32x32 image is taken as an input into the 1st convolutional neural network layer with patch or kernel size of 3x3. After Max Pooling, the output is again fed into a 2nd convolution layer with patch size of 5x5. Its output again fed into Max Pooling layer of same configuration as before. Then output is fed into a fully connected network where every single node is connected with each other. After that, another fully connected layer is added to flatten the data again.

After that a max-pooling layer down-sample the input (output matrix from previous layer) representation and also reduce the dimensionality. The 2nd layer CNN and max-pooling operations are same as 1st layer convolution output of 2nd layer CNN is input of fully connected layer.

The model architecture sometimes also known as Logits, takes some placeholder inputs such as weights and biases for each convolution and fully connected layer. The weights and biases are the main tuning parameter that is used to make a model learn from its training dataset. When the training session is first initiated, the weights are randomly generated with a standard deviation of 0.03 and biases with zero. Then the Logits use these initial placeholder weights and biases to calculate a predicted value for the initial input of character image. With the predicted output of the initial values from the Logits, a SoftMax function is used to normalize the prediction output between 0.0 and 1.0. Now, with this prediction, we can compare it to the true label of the input data and update the value of weights and biases accordingly. In order for our system to determine how much of the weights and biases value have to be changed in order to improve the accuracy of the prediction, we have calculated the loss function. To calculate the loss function, it was used the cross-entropy formula. Cross entropy is a method of comparing the scores predicted from each of training or testing step and comparing it with the ground truth label from the dataset. It is considered to be the distance between the predicted scores and labels.

The dataset has been divided into 3 parts, as Training, Validation and Testing. Training set is the sole set of data that is used to train our model. But as we are using stochastic gradient descent, the training set was divided into random min-batch so that to decrease the training time for large datasets.

SoftMax Classifiers

SoftMax is the last layer found at the end of the network that provides your real probability scores for each class label. SoftMax classifiers give you probabilities for

each class label but in case of hinge loss gives you the margin [52]. It is much simpler for us to understand probabilities rather than margin scores such as in hinge loss and squared hinge loss. The SoftMax classifier is a generalization of the binary form of Logistic Regression. Mapping function f is defined by receiving an input set of data x and maps them to the output class labels via a simple linear dot product of weight matrix W and the data x as show in equation [4.4].

$$f(x_i, W) = Wx_i \qquad 4.4$$

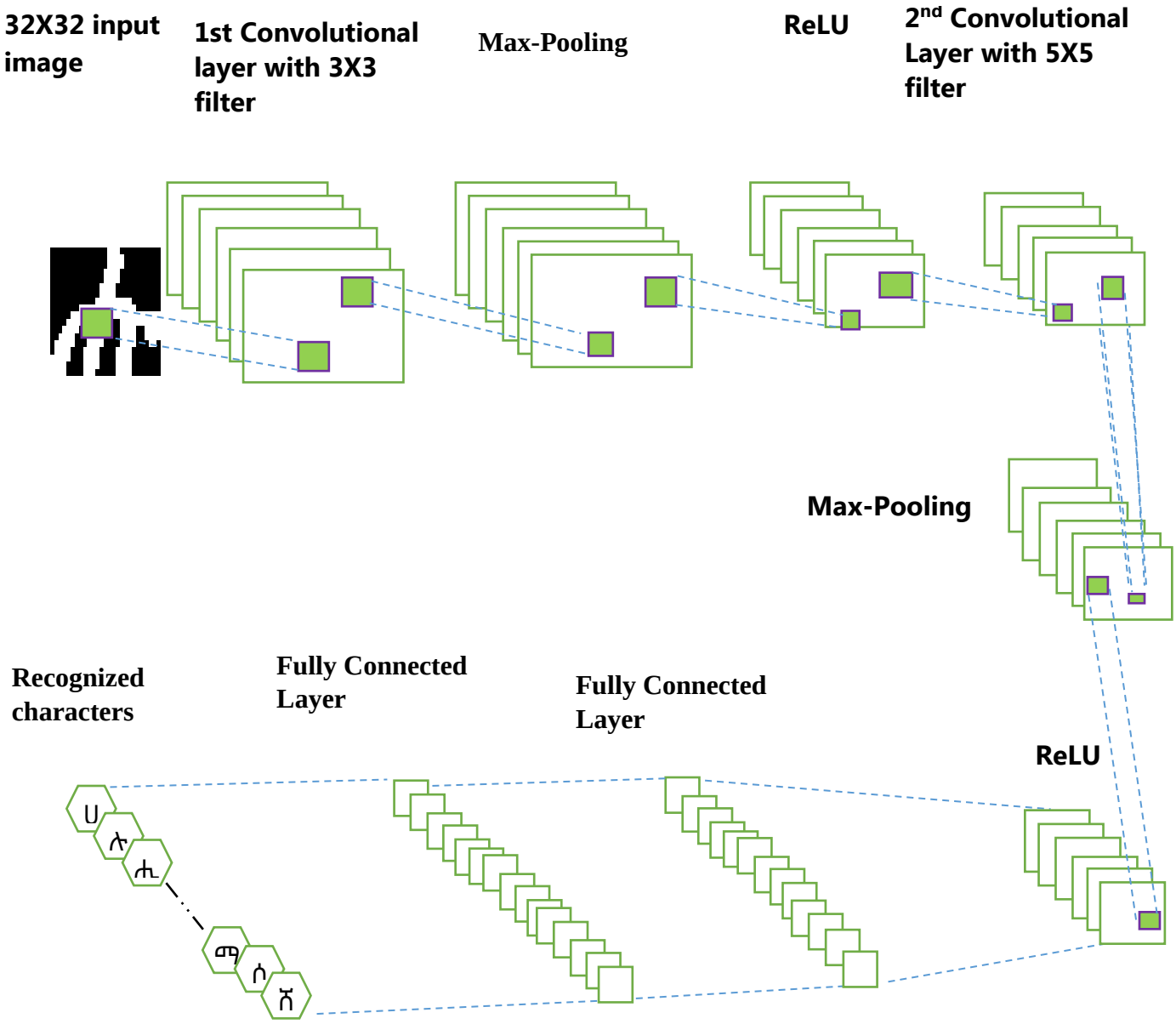


Figure 23: - Model architecture of proposed system

However, unlike hinge loss, we interpret these scores as unnormalized log probabilities for each class label. This amount to swapping out our hinge loss function with cross-entropy loss as shown in equation (4.5):

$$L_i = -log\left(\frac{e^{s_{yi}}}{\sum_j e^{s_j}}\right) \quad 4.5$$

Where s is score and L is loss

To start, our loss function should minimize the negative log likelihood of the correct class:

$$L_i = -logP(Y = y_i|X = x_i) \quad 4.6$$

This probability statement can be interpreted as shown in equation (4.7)

$$P(Y = k|X = x_i) = \left(\frac{e^{s_{yi}}}{\sum_j e^{s_j}}\right) \quad 4.7$$

Where we use our standard scoring function form:

$$s = f(x_i, W) \quad 4.8$$

As a whole, this yields our final loss function for a single data point, just like above:

$$L_i = -log\left(\frac{e^{s_{yi}}}{\sum_j e^{s_j}}\right) \quad 4.9$$

The logarithm here is actually base e (natural logarithm) because we are taking the inverse of the exponentiation over e earlier [52].

The actual exponentiation and normalization via the sum of exponents is our actual SoftMax function. The negative log gives our real cross-entropy loss by calculating the cross-entropy loss done over all dataset done by taking the average using equation [4.10]:

$$L = \frac{1}{n}\sum_{i=1}^{N} L_i \quad 4.10$$

CHAPTER FIVE: RESULTS AND DISCUSSION

In this chapter the result obtained from the experiments were presented. The image size of the training dataset is correlated to the classification accuracy that can be obtained from the model. Unfortunately, increasing the image size comes with an additional cost of computational requirements and training time. In addition to computational cost and training time, to get higher image size it is required to scan the documents with higher resolution. In the proposed system it was used character image with size of 32x32 pixels, because we can get such character image size from document image scanned with 300dpi to 400dpi.

5.1. Result of Preprocessing

The purpose of preprocessing is to convert gray scale, noisy and skewed document images into deskewed, binarized with reduced noise images.

A. Binarization

Binarization is a process of separation of pixel values of a given input image into two-pixel values white as background and black as foreground or the reverse. It is an important part in image processing and the first step in many document analysis and OCR processes. Most of the binarization methods relate a certain intensity value called threshold. Each and every pixel of the concerned gray scale input image should be compared with the threshold value and according to it, pixels are separated into two class background and foreground. Therefore, threshold plays a main part in binarization and selecting of proper threshold value is critical thing. For easy computation consistency and its effectiveness Otsu thresholding method was used in the system to convert gray scale image to binary image as show in the Figure 24. The given input image shown Figure 24 (A) was converted into binarized white background and black foreground image as shown in Figure 24 (b).

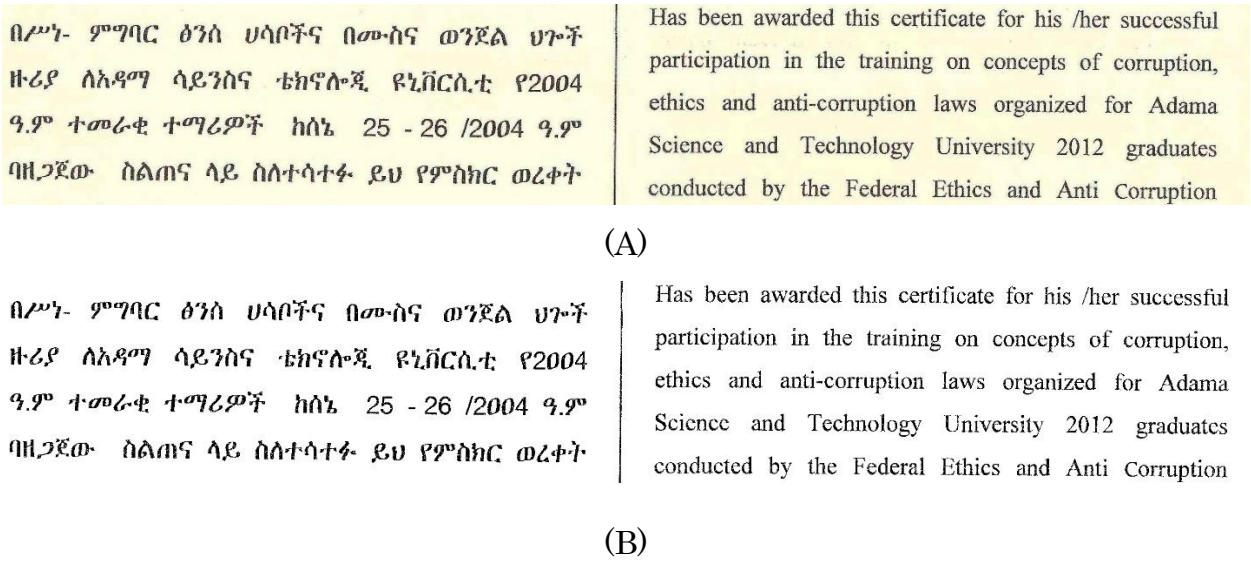
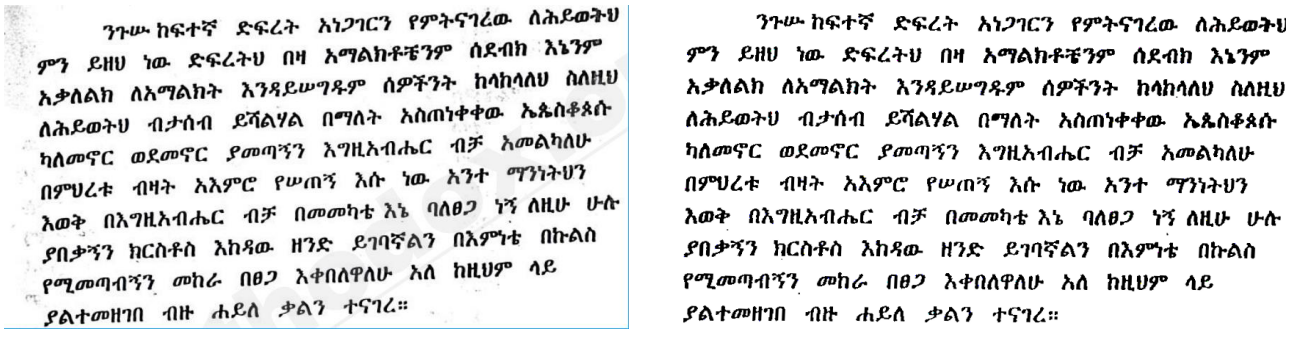


Figure 24: - Sample document image before and after binarization

B. Skew Removal

On document images, skew may exit while scanning or writing the document. This skew must be removed to get better segmented image. Segmenting image in to line (by using horizontal projection) without correcting skew lead to wrong result as shown in the Figure 3. Because, two or more lines may be considered as single line. As described in section **3.3.2.2**, by base line identification and skew angle correction, skew will be detected and removed respectively using Hough-transform method. Figure 25 (A) and (B) show sample document image before and after deskewing.



(A). (B).

Figure 25: - A). skewed image with angle of 2.3^0 B). After removing skew

5.2. Result of Character Segmentation

Documents are segmented in to individual characters after documents segmented into lines. Before image brought for segmentation, the image must be preprocessed and skew free. Since line segmentation done on such document images, 100% accuracy was obtained from the experimentation. But in character segmentation stage, the same result is not achieved. The first reason for loss of accuracy in character segmentation is characters written in italic format and/or the existence of character overlap. Because more than one characters may be considered as single character. The second reason is when the pixel in the part of characters is very small, it may divide the single characters into more than one. Third reason is some errors during preprocessing steps.

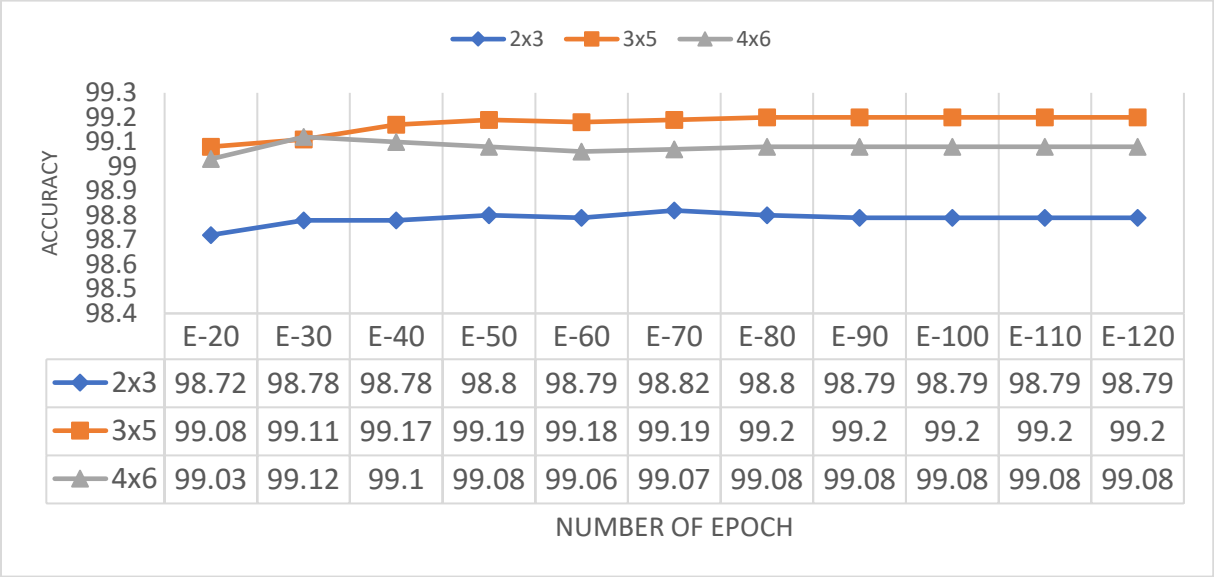
To evaluate accuracy in segmentation, six sample document images were used from both clear and unclear (unclear means image which is degraded, blurred or scanned with very low-resolution) type of documents. As shown in Table 6, out of 3,591 sample character images taken from clear documents, 41-character images were not segmented correctly. That means average of 98.85% segmentation accuracy was obtained. In case of unclear documents, out of 3,601-character images taken from unclear documents, 149-character images were not segmented correctly. Therefore, average segmentation accuracy for unclear document becomes 95.86%. Experiments on the character segmentation include accuracy in preprocessing stages.

Table 6: - Accuracy of character segmentation for sample documents

	Scanned from Clear Document			**Scanned from Unclear Document**		
	Total characters	**Error**	**Accuracy**	**Total**	**Error**	**Accuracy**
Sample -1	1233	9	99.27	1791	56	96.9
Sample -2	1293	18	98.6	1004	39	96.12
Sample -3	1065	14	98.68	807	54	93.31
Average			**98.85**			**95.86**

5.3. Result of Feature Extraction

Feature extraction is an important portion of the proposed system that describes the pattern by means of a minimum number of features or attributes that are effective. In the proposed system it consists of two convolutional layer, two ReLU layers and two pooling layers. The image of size 32x32 pixel is taken as an input into the first convolutional neural network layer with patch or kernel size of 3x3 and in second convolutional layer with kernel size of 5x5. Also, it was experimented by changing kernel size in both convolutional layers and the result is shown in Figure 26.



	E-20	E-30	E-40	E-50	E-60	E-70	E-80	E-90	E-100	E-110	E-120
2x3	98.72	98.78	98.78	98.8	98.79	98.82	98.8	98.79	98.79	98.79	98.79
3x5	99.08	99.11	99.17	99.19	99.18	99.19	99.2	99.2	99.2	99.2	99.2
4x6	99.03	99.12	99.1	99.08	99.06	99.07	99.08	99.08	99.08	99.08	99.08

Figure 26 : - Classification accuracy for with different kernel size

Where

- "2x3" means kernel size in first conv layer is 2x2 and in second layer is 3x3
- "3x5" means kernel size in first conv layer is 3x3 and in second layer is 5x5
- "4x6" means kernel size in first conv layer is 4x4 and in second layer is 6x6

As shown in the Figure 26 highest classification accuracy was obtained when kernel size is 3x3 on first convolutional layer and 5x5 on second convolutional layer. Also, during experimentation it was observed that as size of kernel increase, training time increase.

5.4. Result of Classification

Experimentation was performed through comparison by tuning different hyper parameters to measure the performance of our proposed architecture. For the proposed system, the following fixed and variable parameters were set.

- The size of input image has set to 32x32 pixels as discussed in introduction part of Chapter Five. So that the number of input neuron become 1024.
- Adaptive learning rate was used with *exponential_decay()* function having parameter of starting learning rate of 0.1. Also, learning rate set to different static value of 0.009, 0.04, 0.06 and 0.1 and their training process were discussed.
- Epoch was set to variable value for 20, 30, 40, 50, 60, 70, 80, 90, 100 and 110
- Batch size was set to 64.
- In the architecture of proposed system, two fully connected layer was implemented. Number of neurons in those hidden layers set to variable value of 16, 32, 64, 96, 128, 160, 192, 224, 256, 384, 640 and 1024.
- Number of neurons in output layer is 257, since number of characters for classification was 257.

The network trained with 39,322-character images and tested with 9,868-character images. It was experimented to test classification accuracy of system by varying the number neuron in hidden layers, number of epochs and dropout parameters.

To show the result of the system in more detail, only selected number of epochs and number of neurons in hidden layers are presented in the following paragraphs. E- stands for number of epoch and N- stands for number of neurons in hidden layers.

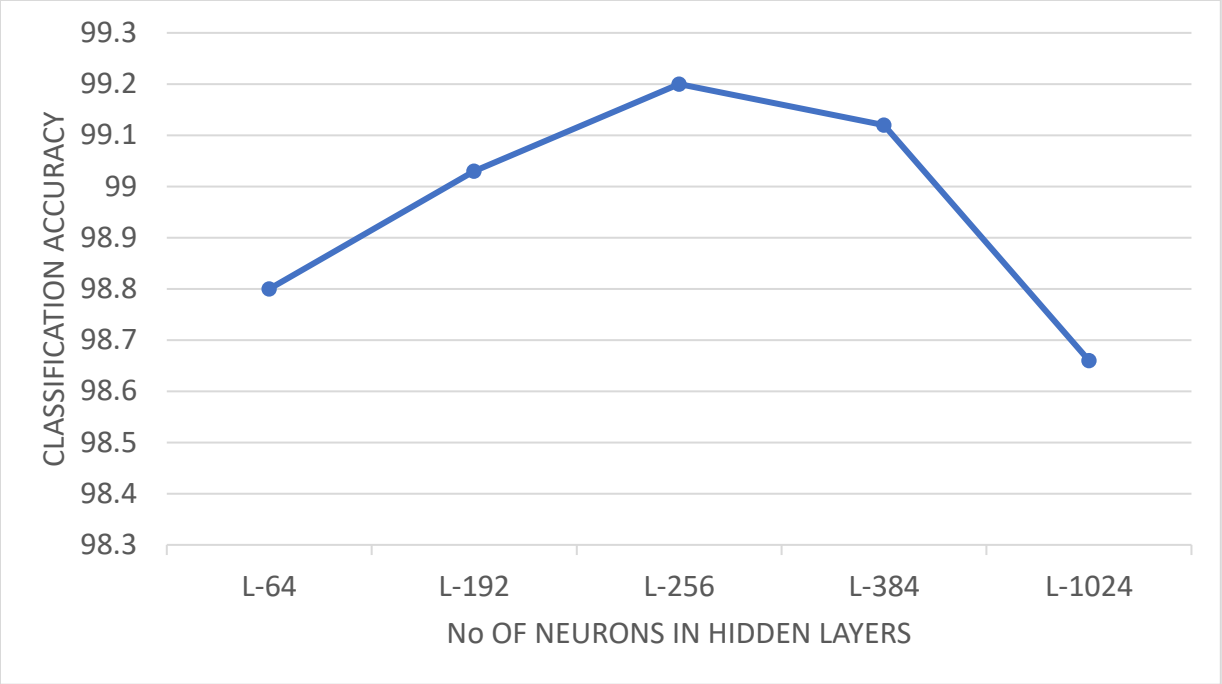


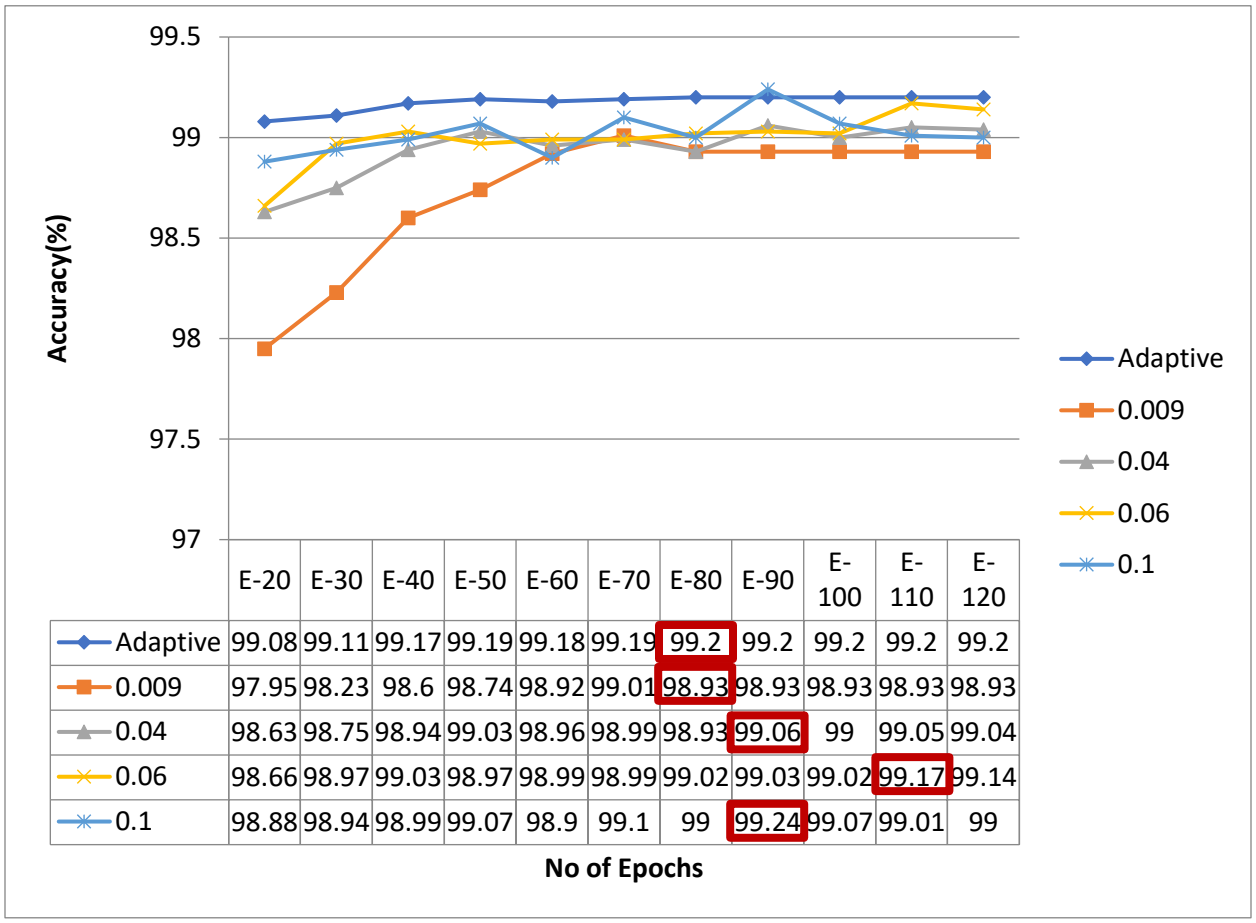
Figure 27: - Test Accuracy for selected number of neurons

The graph on Figure 27 illustrates the comparison of classification accuracy for five different number of neurons in the hidden layers i.e. when number of neurons is 64, 192, 256, 384 and 1024. As number of neurons increase, the classification accuracy increases. However, the accuracy starts to decease when the numbers of neurons exceed 256. When the neural network has few hidden neurons, it does not have capacity to learn enough underlying pattern to classify 257-character classes. When the number of neurons increases much more, it causes the network to overfit that it unable to generalize unseen data. From the experiments it was observed that best result was obtained when the number of neurons is 256 with accuracy of 99.2%.

In this research work RQ1 was answered by obtaining 99.2% of classification accuracy by using selected CNN model.

It was known that choosing a proper learning rate is one of challenges in designing neural network model. Too small learning rate cause to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to oscillate around the minimum or even to diverge.

As shown in the Figure 28 the network was trained with different learning rate of 0.1, 0.06, 0.04, 0.009 and adaptive learning rate.



	E-20	E-30	E-40	E-50	E-60	E-70	E-80	E-90	E-100	E-110	E-120
Adaptive	99.08	99.11	99.17	99.19	99.18	99.19	99.2	99.2	99.2	99.2	99.2
0.009	97.95	98.23	98.6	98.74	98.92	99.01	98.93	98.93	98.93	98.93	98.93
0.04	98.63	98.75	98.94	99.03	98.96	98.99	98.93	99.06	99	99.05	99.04
0.06	98.66	98.97	99.03	98.97	98.99	98.99	99.02	99.03	99.02	99.17	99.14
0.1	98.88	98.94	98.99	99.07	98.9	99.1	99	99.24	99.07	99.01	99

Figure 28: - Classification Accuracy with different learning rate

It was seen that for learning rate of 0.1, 0.04 and 0.06 the accuracy reaches its maximum value early, but the classification accuracy still oscillates up and down. When learning rate is 0.009, accuracy change slowly and minimum classification accuracy was obtained even when number of epochs is 120. In case of adaptive learning rate with gradient descent optimizer the training process converge and the loss become stable when the number of epochs reach some level. Therefore, best training process is found when network trained with adaptive learning rate.

Dropout is a mechanism that allows you to have a probability that random neurons become inactive during a single training element. The model that created should

not be too specific and should have the ability to generalize. To prevent over fitting or not to be too specific, regularization methods are used and dropout is one among them. The dropout function tf.nn.dropout(x, keep_prob), randomly deactivates a set of neurons based on the probability "keep_prob". In the proposed system experiments were done by changing the value "keep_prob" of 0.25, 0.5 and 0.75 and without use of dropout ("keep_prob" = 1) as shown in the Figure 29. The value of dropout is one minus keep_prob value (Dropout=1-keep_prob).

Figure 29: - Classification error for different "keep_prob" value

As shown the result of experiment on the Figure 29, when network trained without dropout it means when "keep_prob" equal to 1, due to over fitting the classification error was higher. As it has seen lowest classification error seen when "keep_prob" is 0.5.

5.5. Comparison with other Ethiopic OCR Systems

Million M. [3] did on "optical character recognition of Amharic documents". He had used principal component analysis and linear discriminant analysis, followed by a decision directed acyclic graph-based support vector machine classifier. Finally, he got 88.23% to 91.45% accuracy from different real-life documents. Another researcher Sertse A et al [2] done on "bilingual script identification for optical character recognition of Amharic and

English printed documents" using Support Vector Machine algorithm. The objective of this research was to identify whether the given character is from Amharic or English scripts. It does not classify characters to corresponding character classes. In this work the average accuracy over mixed Amharic and English real documents was 91.51%. work of Yaregal et al [13] is one of research related to this work. For the implementation of the recognition system, this work used direction field tensor as a tool for character segmentation, and extraction. Finally, the experimental result shows an average of 95.33% accuracy for clean document and 91% accuracy for real life documents.

In this thesis work, I used new feature extraction and classification algorism called Convolutional Neural Network (CNN). To the best of my knowledge this algorism was not used for other Ethiopic OCR system yet. In order to test this OCR system with dataset used by other Ethiopic OCR system, this researcher unable to get their datasets. The dataset used in this system is not tested using other proposed algorism, because the implementation of the algorism could not be found. Even though other Ethiopic OCR system used different implementation methods and different dataset, when we compare overall character recognition result with those OCR systems, promising result was achieved with accuracy of 98.06% for clear documents and 95.09% for unclear documents. From obtained overall recognition result, we can say that RQ2 was answered, since we got good recognition result and also most Ethiopic characters were incorporated in the dataset.

CHAPTER SIX: CONCLUSION AND RECOMMENDATION

6.1. Conclusion

The aim of this research is to recognize characters from printed document images those written in Latin and/or Amharic script. Since almost all printed documents in Ethiopia are written either in one of those script or both of those scripts, researching on this area have significant roles in Amharic OCR system.
Tasks in the development of character recognition system includes data collection, image capturing using scanner or digital camera, binarization captured images, noise reduction, removing skew if any, line segmentation, character segmentation, feature extraction and character classification.

Dataset was prepared from both Amharic and Latin scripts. From Amharic script all core character and from Latin script all characters were included. It means 231 characters from Amharic script and 52 from Latin (merged capital and small letter). For training and testing the system totally 49,087-character images were used. Even though the number of total character images were huge, due to large number of character classes in both scripts, 191 characters were used per each class. The factor that hindered the progress of the implementation by consuming our valuable time was preparing this huge size of dataset for the system. This is due to unavailability of prepared dataset for the research. In order to help future researcher by providing additional dataset, prepared dataset for this research work was shared in this link https://drive.google.com/file/d/1ywg25O6FAFZVixfr1YSlppqEIU4uhOGm/view.

It was shown that effective utilization of CNN achieved good performance to classify bilingual Amharic and Latin characters. Different values of learning rate were tested and best training process was observed when exponentially decaying adaptive learning rate was used. Also, the experiments were done with various number of neurons in hidden layer and finally best result of 99.20% of classification

accuracy obtained when it was 256. In character segmentation average of 98.85% accuracy was achieved for clear sample document and 95.86% for unclear sample documents. Accuracy in character segmentation also includes accuracy in preprocessing stage.

Character recognition is over all process of converting input image in to extracted and machine encoded text by preprocessing input image, segmenting it into character, extracting relevant information and finally classifying character images into their class. Therefore, over all character recognition result the system become the product of character segmentation and character classification. So, accuracy of character recognition be come 98.06% for clear type of sample documents and 95.09% unclear type of sample documents.

6.2. Recommendation and Future Work

Our system has good recognition accuracy so can be used for practical application by taking the following tasks to make it a complete OCR system.

- In this work punctuation marks, numbers and labialized characters were not included. So, adding remaining characters to dataset classes and retrain can make it full OCR system.
- The dataset contains 160 original character images, 15 augmented by noise and 16 augmented by rotating character images for each 257-character classes. Increasing the number of characters for training and test dataset may improve the accuracy further to perfection.
- By applying the recognition system of the research, commercial OCR software can be developed.

REFERENCES

[1] J. Nasriwala. *Introduction to Optical Character Recognition [Online]*. Available: http://www.srimca.edu.in/Srijan/Srijanjan2011/IT/IntroductionToOCR.html

[2] A. SERTSE, "Bilingual Script Identification For Optical Character Recognition of Amharic and English Printed Document (Masters Thesis) Addis Ababa," *Addis Ababa University,* 2011.

[3] M. Million, "Recognition and Retrieval from Document Image Collections," *International Institute of Information Technology Hyderabad 500 032,* 2008.

[4] A. Yaregal, "OPTICAL CHARACTER RECOGNITION OF AMHARIC TEXT: AN INTEGRATED APPROACH," 2002.

[5] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghash, "Optical Character Recognition Systems for Different Languages with Soft Computing," *Springer International Publishing,* vol. 352, 2017.

[6] A. Sakila and S. Vijayarani, "Skew Detection and Correction in the Document Image," *International Journal of Innovative Research in Science, Engineering and Technology,* vol. 6, 2017.

[7] L. Eikvil, "Optical Character Recognition," *International Journal of Emerging Technology and Advanced Engineering,* 1993.

[8] G. A. DEMEKE, "LINCOM Studies in Afroasiatic Linguistics," vol. 28,29, 2010.

[9] Omniglot. *Amharic Writing System [Online]*. Available: http://www.omniglot.com/writing/amharic.htm

[10] Y. Chauhan. (2013). *Latin alphabet [Online]*. Available: https://www.britannica.com/topic/Latin-alphabet

[11] (2014, 13/01/2018). *Romanization systems*. Available: https://www.gov.uk/government/publications/romanisation-systems

[12] M. Million and C. V. Jawahar, "Recognition of Printed Amharic Documents," *African Journal of Information and Communication Technology,* vol. 3.

[13] A. Yaregal and J. Bigun, "Ethiopic Character Recognition Using Direction Field Tensor," *The 18th International Conference on Pattern Recognition,* 2006.

[14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, and E. Tzeng, "A deep convolutional activation feature for generic visual recognition," *Icml,* vol. 32, pp. 647–655, 2014.

[15] A.H.Kulkarni, P.S.Upparamani, R. J. Kadkol, and P. V. Tergundi, "Script Identification from Multilingual Text Documents," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4, 2015.

[16] N. J. Pithadia and V. D. Nimavat, "A Review on Feature Extraction Techniques for Optical Character Recognition," *International Journal of Innovative Research in Computer and Communication Engineering,* 2015.

[17] N. Vasudeva, H. J. Parashar, and S. Vijendra, "Offline Character Recognition System Using Artificial Neural Network," *International Journal of Machine Learning and Computing,* vol. 2, 2012.

[18] B. Jain and M. Borah, ""A Comparison Paper on Skew Detection of Scanned Document Images Based on Horizontal and Vertical," *IJSRP,* 2014.

[19] Y. J. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition, Vol. 29, No. 8, pp. 1335-1346*.

[20] Y. Alginahi, "Preprocessing Techniques in Character Recognition," 2010.

[21] M. Shen, "Improving OCR Performance with Background Image Elimination," *12th Int. Conf. Fuzzy Syst. Knowl. Discov.,* 2015.

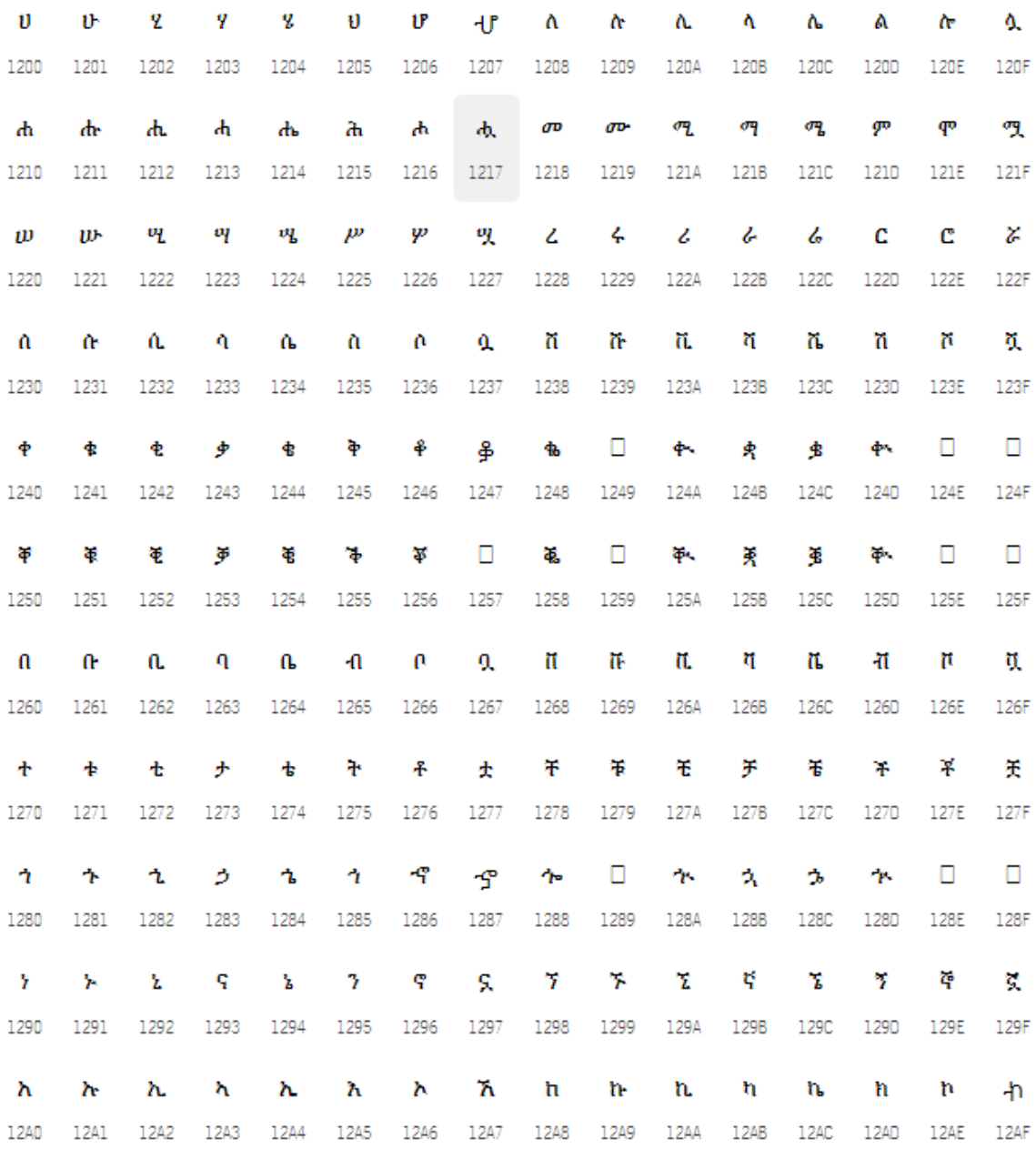
[22] N. Bhatia, "Optical Character Recognition Techniques:," *International Journal of Advanced Research in Computer Science and Software Engineering,* 2014.

[23] A. Singh, K. Bacchuwar, and A. Bhasin, "A Survey of OCR Applications," *Int. J. Mach. Learn. Comput,* vol. vol-2, pp. pp. 314–318, 2012.
[24] N. Kumar and G. BATHINDA, "Binarization Techniques used for Grey Scale Images," *International Journal of Computer Applications,* vol. 71, 2013.
[25] K. Khurshid, I. Siddiqi, C. Faure, and N. Vincent, "Comparison of Niblack inspired Binarization methods for ancient documents," *16th International conference on Document Recognition And Retrieval,* 2009.
[26] A. Mahmoud and K. Omar, "Skew Detection and Correction Technique for Arabic Document Images Based on Centre of Gravity," *Journal of Computer Science,* 2009.
[27] A. Farahmand, A. Sarrafzadeh, and J. Shanbehzadeh, "Document Image Noises and Removal Methods," *Proceedings of the International MultiConference of Engineers and Computer Scientists,* vol. 1, 2013.
[28] T. Abu-Ain, S. N. Huda, S. Abdullah, B. Bataineh, and K. Omar, "A Fast and Efficient Thinning Algorithm for Binary Images," *ITB,* vol. 7, 2013.
[29] A. Devi. (25/04/2018). *Thinning: A Preprocessing Technique for an OCR System for the Brahmi Script [Online]*. Available: https://www.ancient-asia-journal.com/articles/10.5334/aa.06114/
[30] L. Lam and S. Lee, "Thinning Methodologies-A Comprehensive Survey," pp. 869-885.
[31] N. Dave, "Segmentation Methods for Hand Written Character Recognition," *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 8, pp. 155-164, 2015.
[32] G. S. Clemente, T. I. Ren, and G. D. C. Calvalcanti, "Text-Based Image Segmentation Methodology," *2nd International Conference on Innovations in Automation and Mechatronics Engineering,* 2014.
[33] S. B. Patil, "Neural Network based bilingual OCR system: experiment with English and Kannada bilingual document," *International Journal of Computer Applications,* vol. 13, 2011.
[34] V. Kumar and P. K. Sengar, "Segmentation of Printed Text in Devanagari Script and Gurmukhi Script," *International Journal of Computer Applications,* vol. 3, 2010.
[35] H. Kaur, "A Survey of Feature Extraction and Classification Techniques Used In Character Recognition for Indian Scripts," *An International Journal of Engineering Sciences,* vol. 3, 2014.
[36] P. U. Talole and P. E. Ajmire, "Comparative Study of Feature Extraction and Classification Techniques for Handwritten Devanagari Script," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 6, 2017.
[37] A. Rosebrock. (Sep, 2016). *Softmax Classifiers Explained [Online]*. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
[38] H. SINGHAL. (2017, 24/01/2018). *Convolutional Neural Network with TensorFlow implementation [Online]*. Available: https://medium.com/data-science-group-iitr/building-a-convolutional-neural-network-in-python-with-tensorflow-d251c3ca8117
[39] A. Hamad and M. Kaya, "A Detailed Analysis of Optical Character Recognition Technology," *International Journal of Applied Mathematics, Electronics and Computers,* 2016.
[40] D. Doermann, "Text detection and recognition in imagery," *IEEE transactions on pattern analysis and machine intelligence,* 2015.
[41] A. Jain, A. Dubey, R. Gupta, and N. Jain, "Fundamental challenges to mobile based ocr," *International journal of innovative research Studies,* 2013.
[42] K. Moravec, "Grayscale Reader for Camera Images of Xerox DataGlyphs," 2012.
[43] R. Smith and D. Antonova, "Adapting the Tesseract open source OCR engine for multilingual OCR," *In Proceedings of the International Workshop on Multilingual OCR,* 2009.

[44] A. Ulges, C. Lampert, and T. Breuel, "Document image ewarping using robust estimation of curled text lines," *In Eighth International Conference on Document Analysis and Recognition,* 2005.
[45] M. Makkar and A. Pundir, "Image analysis using improved Otsu thresholding methods," *IJRITCC,* vol. 2, 2014.
[46] D. Kuma and D. Singh, "Modified approach of hough transform for skew detection and correction in documented images," *International Journal of Research in Computer Science,* vol. 2, 2012.
[47] A. Mordvintsev and K. Revision. (2013, 12/10/2017). *Hough Line Transform [Online]*.
[48] (2015). *Hough Line Transform [Online]*. Available: https://docs.opencv.org/3.1.0/d6/d10/tutorial_py_houghlines.html
[49] N. Hirschkind, J. Pari, and J. Khim. (2018). *Convolutional Neural Network [Online]*. Available: https://brilliant.org/wiki/convolutional-neural-network/
[50] K. Willems. (July, 2017). *TensorFlow Tutorial For Beginners [Online]*. Available: https://www.datacamp.com/community/tutorials/tensorflow-tutorial
[51] A. Mordvintsev and A. Rahman. (2013). *Introduction to OpenCV-Python Tutorials [Online]*. Available: https://docs.opencv.org/3.4/d0/de3/tutorial_py_intro.html
[52] A. Rosebrock. (2016, 01/03/2018). *Softmax Classifiers [Online]*. Available: https://www.pyimagesearch.com/2016/09/12/softmax-classifiers-explained/

APPENDIX A: - Unicode of Amharic Characters

(Source http://jrgraphix.net/r/Unicode/1200-137F)



ኰ	□	ኲ	ኳ	ኴ	ኵ	□	□	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	□
12B0	12B1	12B2	12B3	12B4	12B5	12B6	12B7	12B8	12B9	12BA	12BB	12BC	12BD	12BE	12BF
ዀ	□	ዂ	ዃ	ዄ	ዅ	□	□	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	ዏ
12C0	12C1	12C2	12C3	12C4	12C5	12C6	12C7	12C8	12C9	12CA	12CB	12CC	12CD	12CE	12CF
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	□	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
12D0	12D1	12D2	12D3	12D4	12D5	12D6	12D7	12D8	12D9	12DA	12DB	12DC	12DD	12DE	12DF
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	ዯ
12E0	12E1	12E2	12E3	12E4	12E5	12E6	12E7	12E8	12E9	12EA	12EB	12EC	12ED	12EE	12EF
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
12F0	12F1	12F2	12F3	12F4	12F5	12F6	12F7	12F8	12F9	12FA	12FB	12FC	12FD	12FE	12FF
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጏ	ጐ	጑	ጒ	ጓ	ጔ	ጕ	጖	጗
1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	130A	130B	130C	130D	130E	130F
ጘ	□	ጙ	ጚ	ጛ	ጜ	□	□	ጟ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	131A	131B	131C	131D	131E	131F
ጧ	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	132A	132B	132C	132D	132E	132F
ጷ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	133A	133B	133C	133D	133E	133F
ፇ	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	134A	134B	134C	134D	134E	134F
ፗ	ፘ	ፙ	ፚ	፛	፜	፝	፞	፟	፠	፡	።	፣	፤	፥	፦
1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	135A	135B	135C	135D	135E	135F
፧	፨	፩	፪	፫	፬	፭	፮	፯	፰	፱	፲	፳	፴	፵	፶
1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	136A	136B	136C	136D	136E	136F

APPENDIX B: All Amharic Characters Including Core and Labialized Characters (source www.lexilogos.com)

	e/ä	u	i	a	ē	ə	o	wa
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ
hh	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ
sz	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ
sh	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ
qh	ቐ	ቑ	ቒ	ቓ	ቔ	ቕ	ቖ	
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ
v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	ቯ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ
c	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	ቿ
x	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ
ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ
'	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ	ኧ

	e/ä	i	a	ē	ə
qw	ቈ	ቊ	ቋ	ቌ	ቍ
qhw	ቘ	ቚ	ቛ	ቜ	ቝ
xw	ኰ	ኲ	ኳ	ኴ	ኵ
kw	ኰ	ኲ	ኳ	ኴ	ኵ
kxw	ዀ	ዂ	ዃ	ዄ	ዅ

	e/ä	u	i	a	ē	ə	o	wa
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኯ
kx	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
'	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
zh	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	ዯ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
dd	ዸ	ዹ	ዺ	ዻ	ዼ	ዽ	ዾ	ዿ
j	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጏ
gg	ጘ	ጙ	ጚ	ጛ	ጜ	ጝ	ጞ	ጟ
th	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ch	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ph	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ts	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
tz	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	ፇ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	ፗ

APPENDIX C: - Sample Code

```
# This is the image processing module
import numpy as np
import os
import tensorflow as tf
from scipy import ndimage
from six.moves import cPickle as pickle
from PIL import Image
folder_for_train  = "Train"
folder_for_test   = "Test"
# dimensions to which all images will be rescaled
image_size = 32  # size of image (width and height) in pixel
dimensions = (image_size, image_size)
def invert_colors(im):
    im = im.convert('RGBA')    # Convert to RGBA
    data = np.array(im)    # "data" is a height x width x 4 numpy array
    ……
    return im2
def process_images(folder):
    classes = [os.path.join(folder, d) for d in sorted(os.listdir(folder))]
# get list of all sub-folders in folder
    img_cnt = 0
    for class_x in classes:
        if os.path.isdir(class_x):
            # get paths to all the images in this folder
            images = [os.path.join(class_x, i) for i in
sorted(os.listdir(class_x))]
            for image in images:
                img_cnt = img_cnt + 1
                if(img_cnt % 1000 == 0):                # show progress
                    print("Processed %s images" % str(img_cnt))
                im = Image.open(image)
                im = im.resize(dimensions)    # resize image according to
dimensions set
           ----------------------------
                im.save(image) # overwrite previous image file with new image
process_images(folder_for_train )
```

```
process_images(test_folder)
def load_letter(folder, min_num_images):
  """Load the data for a single letter label."""
  image_files = os.listdir(folder)
  dataset = np.ndarray(shape=(len(image_files), image_size, image_size),
                       dtype=np.float32)
  num_images = 0
  for image_index, image in enumerate(image_files):
    image_file = os.path.join(folder, image)
    try:
      image_data = (ndimage.imread(image_file).astype(float) -      #
normalize data
                    pixel_depth / 2) / pixel_depth
      if image_data.shape != (image_size, image_size):
        raise Exception('Unexpected image shape: %s' % str(image_data.shape))
      dataset[num_images, :, :] = image_data
      num_images = num_images + 1
    except IOError as e:
      print('Could not read:', image_file, ':', e, '- it\'s ok, skipping.') #
skip unreadable files
  dataset = dataset[0:num_images, :, :]
  if num_images < min_num_images:       # check if a given min. no. of images
    raise Exception('Many fewer images than expected: %d < %d' % # has been
loaded
                    (num_images, min_num_images))
  return dataset
# function to store the normalized tensors obtained from the load_letter
function in
# .pickle files for later use
def maybe_pickle(data_folders, min_num_images_per_class, force=False):
  dataset_names = []
  folders_list = os.listdir(data_folders)
  for folder in folders_list:
    curr_folder_path = os.path.join(data_folders, folder)
    ----------------------------
  return dataset_names
train_datasets = maybe_pickle(folder_for_train , 153, False)     # load,
normalize and pickle the train and test datasets
```

```
test_datasets = maybe_pickle(test_folder, 32, False)
# function to make two empty arrays, one for the input data and one for the
labels
def make_arrays(nb_rows, img_size):
  if nb_rows:
    dataset = np.ndarray((nb_rows, img_size, img_size), dtype=np.float32)
    labels = np.ndarray(nb_rows, dtype=np.int32)
  else:
    dataset, labels = None, None
  return dataset, labels
# function to merge all the images in the given pickle file. Part of the
training dataset is used to
# create a validation dataset for hyperparameter tuning.
def merge_datasets(pickle_files, train_size, valid_size=0):
  num_classes = len(pickle_files)
  valid_dataset, valid_labels = make_arrays(valid_size, image_size)
  train_dataset, train_labels = make_arrays(train_size, image_size)
  vsize_per_class = valid_size // num_classes
  tsize_per_class = train_size // num_classes
  start_v, start_t = 0, 0
  end_v, end_t = vsize_per_class, tsize_per_class
  end_l = vsize_per_class+tsize_per_class
  for label, pickle_file in enumerate(pickle_files):
    try:
      with open(pickle_file, 'rb') as f:
        letter_set = pickle.load(f)
        f.close()
       ---------------------------------
        start_t += tsize_per_class
        end_t += tsize_per_class
    except Exception as e:
      print('Unable to process data from', pickle_file, ':', e)
      raise
  return valid_dataset, valid_labels, train_dataset, train_labels
---------------------
# shuffle the images in each dataset randomly, and their corresponding labels
def randomize(dataset, labels):
  permutation = np.random.permutation(labels.shape[0])
```

```
  shuffled_dataset = dataset[permutation,:,:]
  shuffled_labels = labels[permutation]
  return shuffled_dataset, shuffled_labels
----------------------
#Training and classification
pickle_file = 'am_eng.pickle'
with open(pickle_file, 'rb') as f:
  save = pickle.load(f)
  train_dataset = save['train_dataset']
  train_labels = save['train_labels']
  valid_dataset = save['valid_dataset']
  valid_labels = save['valid_labels']
  test_dataset = save['test_dataset']
  test_labels = save['test_labels']
  del save  # hint to help gc free up memory
image_size = 32
num_labels = 257
num_channels = 1 # grayscale
# CNN in tensorflow require 4d tensor inputs
#input data
train_dataset, train_labels = reformat(train_dataset, train_labels)
    ---------------------------
    tf.summary.histogram('histogram', var)
def Conv_NN():
    batch_size = 64
    patch_size1 = 3 #Variable
    patch_size2 = 5# Variable
    depth = 16
    num_neuron1 = 128
    num_neuron2 = 128
    keep_prob = 0.5#Variable
    decay_step = 1000
    base = 0.86
    starter_learning_rate=0.1
    graph = tf.Graph()
    with graph.as_default():
      # Input data.
      tf_train_dataset = tf.placeholder(
```

```
        tf.float32, shape=(batch_size, image_size, image_size, num_channels))
      tf_train_labels = tf.placeholder(tf.float32, shape=(batch_size,
num_labels))
      tf_valid_dataset = tf.constant(valid_dataset)
      tf_test_dataset = tf.constant(test_dataset)
      # Variables.
      layer1_weights = tf.Variable(tf.truncated_normal(
          [patch_size1, patch_size1, num_channels, depth], stddev=0.5))
      variable_summaries(layer1_weights)
      layer1_biases = tf.Variable(tf.zeros([depth]))
      variable_summaries(layer1_biases)
      -----------------
      layer4_biases = tf.Variable(tf.constant(1.0, shape=[num_neuron2]))
      variable_summaries(layer4_biases)
      layer5_weights = tf.Variable(tf.truncated_normal(
          [num_neuron2, num_labels], stddev=0.1))
      variable_summaries(layer5_weights)
      layer5_biases = tf.Variable(tf.constant(1.0, shape=[num_labels]))
      variable_summaries(layer5_biases)
      global_step = tf.Variable(0)  # count the number of steps taken.
      learning_rate = tf.train.exponential_decay(starter_learning_rate,
global_step, decay_step, base)
      # Model
      def model(data, useDropout):
        conv = tf.nn.conv2d(data, layer1_weights, [1, 1, 1, 1],
padding='SAME')
        max_pooled = tf.nn.max_pool(conv, [1, 2, 2, 1], [1, 2, 2, 1],
padding='SAME')
        hidden = tf.nn.relu(max_pooled + layer1_biases)
        conv = tf.nn.conv2d(hidden, layer2_weights, [1, 1, 1, 1],
padding='SAME')
        max_pooled = tf.nn.max_pool(conv, [1, 2, 2, 1], [1, 2, 2, 1],
padding='SAME')
        hidden = tf.nn.relu(max_pooled + layer2_biases)
        shape = hidden.get_shape().as_list()
        reshape = tf.reshape(hidden, [shape[0], shape[1] * shape[2] *
shape[3]])
        if useDropout == 1:
```

```
            dropout_layer2 = tf.nn.dropout(reshape, keep_prob)
        else:
            dropout_layer2 = reshape
        hidden = tf.nn.relu(tf.matmul(dropout_layer2, layer3_weights) +
layer3_biases)
        tf.summary.histogram('pre_activations1', hidden)0
        hidden = tf.nn.relu(tf.matmul(hidden, layer4_weights) +
layer4_biases)
        tf.summary.histogram('pre_activations2', hidden)
        return tf.matmul(hidden, layer5_weights) + layer5_biases
      # Training computation.
      logits = model(tf_train_dataset, 1)#1
      loss = tf.reduce_mean(
        tf.nn.softmax_cross_entropy_with_logits(logits=logits,labels=
tf_train_labels))
      # Optimizer.
      optimizer =
tf.train.GradientDescentOptimizer(learning_rate).minimize(loss,
global_step=global_step)
      # Predictions for the training, validation, and test data.
      train_prediction = tf.nn.softmax(model(tf_train_dataset, 0))
      valid_prediction = tf.nn.softmax(model(tf_valid_dataset, 0))
      test_prediction = tf.nn.softmax(model(tf_test_dataset, 0))
    num_steps = 90000
    with tf.Session(graph=graph) as session:
      tf.global_variables_initializer().run()
      for step in range(num_steps):
           ..................
            if (step % 2500 == 0):
              print('Minibatch loss at step %d: %f' % (step, l))
              print('Test accuracy: %f' % accuracy(test_prediction.eval(),
test_labels))
      print('Over ALL Test accuracy: %f' % accuracy(test_prediction.eval(),
test_labels))
Conv_NN()
```