ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

AMHARIC CHARACTER RECOGNITION SYSTEM FOR PRINTED REAL-LIFE DOCUMENTS

By

ABAY TESHAGER BIRHANU

A thesis submitted to the School of Graduate Studies of Addis Ababa University

In partial fulfillment of the requirements for the Degree of

Master of Science in Information Science

July 2010

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF INFORMATION SCIENCE

AMHARIC CHARACTER RECOGNITION SYSTEM FOR PRINTED REAL-LIFE DOCUMENTS

By

ABAY TESHAGER BIRHANU

Name and Signature of the Examining Board

Ato Tibebe Beseha _____ _____
Chairman, Examining Board

Ato Wondwossen Mulugeta _____ _____
Advisor

Dr. Million Meshesha _____ _____
Examiner

DEDICATION

To God...

To my family…

ACKNOWLEDGEMENT

With the help of God this has become a reality. Therefore, my innumerable praise first goes to God Almighty for his guiding me all the way.

I am greatly indebted to my Advisor Ato Wondwossen M., for all his guidance, insightful comments, patience, and encouragement throughout the process of the thesis work. Without him, this thesis can't have the present form.

Also, grateful to my best friends, Ephrem M., Tewodros K., Yakob T. Helen G., for their help throughout the conduct of the study.

Abay Teshager

July, 2010

Addis Ababa

Ethiopia

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ANN	-Artificial Neural Network.
ASCII	-American Standard Code For Information Interchange.
BMP	-Bit Map.
CEC	-Committee for Ethiopic Computing.
EAS	-Ethiopian Authority for Standardization.
ECoSA	- Ethiopian Computer Standards Association.
EEDN	- Ethiopian E-mail Distribution Network.
ESTC	-Ethiopian Science and Technology Commission.
EthCITA	- Ethiopian Computing and Information Technology Associations.
IES	-Institute of Ethiopian Studies.
IT	-Information Technology.
MATLAB	-MATrix LABoratory.
MLP	-Multi-Layer Perceptron.
NLE	-National Library of Ethiopia.
OCR	-Optical Character Recognition.
PPI	-Pixels Per Inch.
QSAE	- Quality and Standards Authority of Ethiopia.
RLC	-Run Length Code.
SISA	-School of Information Studies for Africa.

ABSTRACT

Optical Character Recognition (OCR) is an area of research and development where a system is made to recognize characters from printed documents. Cultural considerations and enormous flood of printed documents motivated the development of OCR across the world. Unlike other scripts, OCR development for Amharic Characters has been started in 1997 at SISA (School of Information Studies for Africa). Some developments have been made in recognizing various types of machine-printed, typewritten and handwritten Amharic documents.

However, Amharic character recognition is still an area that requires the contribution of many research works. There is a need to enhance its performance on real-life documents such as the 'Addis Zemen' Amharic newspaper, the Bible, the 'Federal Negarit Gazeta' and the fiction 'Fiker Eskemekabir', which have a number of artifacts (mode of writing, condition of the input page, printing process, quality of paper, presence of extraneous markings, resolution and quality of scanning etc.) that affect the performance of the recognizer. One such area, OCR technology *has been investigated more for real-life Amharic degraded documents. For the recognition to be successful, robust techniques in detecting and removing various noise types are investigated and validated.*

During experimentation of the applicability of algorithms and approaches for the problem at hand, MATLAB Image processing Toolbox and neural network classifier on MATLAB Neural Network Toolbox is used. The wiener adaptive filtering method for noise removal, Otsu global thresholding

method for binarizing the digitized image, linear interpolation techniques for normalization and hit-and-miss morphological analysis for thinning are found to work very well for the problem of interest. In due course, the performance of the line segmenter is found to be 100%. The rate of segmentation for basic and labialized characters turns out to be 98.28% and 100% respectively for training character sets, 98.55% and 100% respectively for testing character sets.

For classifying the features generated, an artificial neural network approach is implemented. The neural network is trained with eight samples taken from real-life documents. The performance of the developed system is tested with documents taken from real-life documents. Accordingly, an average recognition rate of 96.87% for the test sets from the training sets and 11.40% recognition rate is observed for the new test sets. The segmentation algorithm used in the current study worked reasonably for basic and labialized characters. But it fails to segment special character |ሸ|, punctuations and numbers. In general, observation of the test results show that the performance of the system is greatly affected by the similarity of the shape of Amharic characters and effect of the application of noise removal for cleaning highly degraded document images. Such challenges require to further explore an invariant to shape feature extraction techniques and advanced noise detection and removal algorithms. Based on the results, further research areas are also recommended.

CHAPTER ONE

INTRODUCTION

1.1. BACKGROUND

There is an enormous amount of information being produced every single bit of our life time, and most of these are in printed as well as handwritten format. Reaching the level of making these information items accessible and computable is one wing of information science research. This wing is called an Optical Character Recognition (OCR). Optical Character Recognition (OCR) is a process that allows printed (typewritten, printout as well as handwritten) text to be recognized optically and converted into machine–readable code that can be accepted by a computer for further processing (Genovese, 1970).

1.1. 1. Historical Development

Investigations into the techniques of OCR started relatively early in the field of pattern recognition. It dates back almost to that of the history of computer (Simon, 1992). The concept was introduced and got recognition after Taushbeck and Handel obtained a patent on OCR in 1929 in Germany and in 1933 in the US respectively (Mori and Yamamoto, 1992). During the early days of OCR, standards that aimed at standardizing the character set, the type of paper used, the ink and character positions were developed to help guide automatic document processing. Despite this effort, the idea of a machine that reads characters and numerals remained a dream until the 1950s (Mori and Yamamoto, 1992).

1.1.2. Current Status

Modern OCR technology is said to have been born in 1951 with the invention of GISMO-A Robot Reader Writer (Srihari and Lam, 1996). The technology since its conception has shown development despite it focus on limited language characters and much efforts was vested on the recognition of typed and machine printed characters. Despite the challenge, there has been strong market demand for OCR products (Mori and Yamamoto, 1992). In response to the demand, a lot of successes have been reported in the area since the 1950s and hundreds of OCR systems are commercially available today. Furthermore, unlike the early times, when OCR systems were considered exotic and futuristic system used only by government agencies and large corporations, today due to less expensive electronic components and extensive research, OCR systems are less expensive, faster, and more reliable (Srihari and Lam, 1996).

OCR systems provide a tremendous opportunity in handling repetitive, boring, labor-intensive, error prone, and time consuming processes for human beings. Postal mail sorting according to destination addresses, bank check processing, bill processing, keying in data to the computer, etc belong to this category. The tasks can be performed with computers in a stable manner (Green, 1993).

In order to develop an OCR system it requires the development and integration of many sub systems (Green, 1993). The first step is preprocessing such as skew detection and correction, noise detection and removal, binarization, thinning, and normalization. Then segmentation of document images into line, word and characters. This is followed by feature extraction for representing character images and a classification module that label characters to their proper

class. Finally, post processing i.e. applying algorithms such as spellchecker and other semantic approaches (Green, 1993).

There are many factors affecting OCR system. Some of them include: the condition of the input page, the printing process, the quality of paper, the presence of extraneous markings, and the resolution and quality of scanning (Dereje, 1999). To overcome such problems, constraints were imposed on the size of the character list, the quality of paper and the resolution of the scanning process. In addition, robust approaches and techniques either in isolation or in combination have been tried to address the problem.

1.2. STATEMENT OF THE PROBLEM AND ITS JUSTIFICATION

These days, Europeans, Americans and others have been conducting researches and applying OCR technologies to their languages. As a result, these OCR technologies help to read different documents written in English, Chinese, Hindu, Arabic, Russian, and the like but do not read documents written in Amharic (Million, 2000).

Since Amharic is serving as a working language of Ethiopia for many years, large amount of information is mounted up in churches, in caves, governmental, non-governmental and private institutions including information centers, libraries, museums, etc. in this language. Thus, the country can take share of these advantages by developing an Amharic OCR system, as the country is endowed with countless historical, cultural and other documents written using Amharic characters.

Due to the reasons mentioned earlier, converting Amharic documents into electronic format is needed. In order to convert the text on these documents the conventional way is typing through the keyboard, which is not only time consuming, error-prone, and tedious. The problem of typing into computers is even worse for Amharic characters where typing each character needs two keystrokes on average. This emphasizes the importance and tremendous need for an Amharic OCR that is capable of recognizing characters. Thus, if automation of documents is needed the OCR software is the preferred means for converting existing documents into machine-readable form.

After Worku (1997) conducted a research on the applicability of OCR for Amharic characters, researches on the Amharic language are ongoing. Ermias (1998) attempted to incorporate preprocessing techniques (thinning and underline removal) to the adopted algorithm on formatted text. Dereje (1999) conducted a research in the area of improving Amharic OCR system by enabling it to recognize typewritten Amharic text in addition to printed ones. Berhanu (1999) studied the use of ANN (Artificial Neural Network) for the recognition of Amharic text using segmentation algorithms. Million (2000) tried to work on the aim of generalizing the previously adopted algorithm. Nigussie (2000) had investigated the recognition of handwritten legal amounts of checks, the purpose of which is investigation of the application of neural networks as a tool to recognize handwritten Amharic characters. Yaregal (2002) showed the basic concept in OCR technology and put light on various character recognition techniques and preprocessing techniques. Messay (2003) made an attempt to solve the problem of identifying postal addresses written in Amharic by applying OCR technology. As a further work, Wondwossen (2004) attempted to develop OCR engine for special type of handwritten Amharic text, which is traditionally called “Yekum Tsehuf”. Teshome (2008) tried to develop an

algorithm to recognize optically scanned Amharic Braille and convert to the equivalent printed Amharic text character. Other works worth mention include are Cowell and Hussain (2003), Worku and Fuchs (2003), Million and Jawahar (2005) and Yaregal and Bigun (2006) on Amharic OCR development.

However, Amharic recognition is still an area that requires the contribution of many research works. There is a need to enhance its performance on real-life documents such as books, magazines and newspapers etc., which have a number of artifacts (mode of writing, condition of the input page, printing process, quality of paper, presence of extraneous markings, and resolution and quality of scanning etc.) that affect the performance of the recognizer. One such area, the core research work of this thesis, was applying robust preprocessing techniques and simplifying the extraction of features which would represent and describe the Amharic characters taken from real-life documents such as books, magazines and newspapers as precisely and uniquely as possible.

Thus in this thesis, OCR technology have been investigated more for real-life Amharic degraded documents. For the recognition to be successful, robust techniques in detecting and removing various noise types are investigated and validated.

1.3. OBJECTIVES OF THE STUDY

1.3.2. General Objective

The general objective of this study is to explore and design noise detection and removal techniques that can enable the development of Amharic character recognizer for real-life documents.

1.3.3. Specific Objectives

In order to achieve the general objective of this study, the following specific objectives are drawn.

- To review literature on different algorithms and techniques employed for recognition activities in OCR.
- To study and identify the different characteristics of Amharic characters through document analysis and literature review.
- To select an appropriate recognition algorithms for the development of Amharic OCR.
- To prepare training and test data sets required for the appropriate algorithm.
- To develop a program of the selected noise detection and removal algorithms using MATLAB programming environment.
- To test the performance of the recognizer with Amharic document images prepared from various real-life documents such as books, magazines and newspapers.
- To draw conclusion and forward recommendations for further study.

1.4. METHODS

To undertake this research work, the following techniques have been used.

1.4.2. Review of Literature

Related literatures from different sources such as books, journals, and internet are reviewed to have conceptual understanding about OCR technology, its development tools, techniques, procedures and methodologies and the characteristics of Amharic characters. Related literatures in the areas of character recognition in general and Amharic OCR in particular are reviewed.

1.4.3. Development Techniques

For the purpose of implementing algorithms required to Amharic text recognition, MATLAB 7's Image Processing and Neural Network Toolboxes are used for the reason that it minimizes the need to write code which is time-consuming, and delivers increased performance and productivity by enabling developers to leverage existing skills. Moreover, to incorporate the program developed in this thesis as part of previous works, the use of the above programming environment is inevitable. Besides, MATLAB's powerful collection of matrix manipulation routines which enables the thesis work to use matrixes, again closely used to model the real world objects.

1.4.4. Testing

Testing was done on the collected real-life documents. By taking the limitations of time and cost into consideration, the neural network was trained with real-life documents. These are data sets from the popular government Amharic newspaper 'Addis Zemen', texts from the New Testament

of the Bible, texts from the ‘Federal Negarit Gazeta’, and the fiction ‘Fiker Eskemekabir’. Texts extracted from the above samples were used during testing.

1.5. SCOPE AND LIMITATION OF THE STUDY

As the main aim of this thesis is to develop an Amharic character recognizer for real-life document images, for the implementation of the recognizer algorithm , the research was planned to consider a complete set of the Amharic characters (Fidel) (core characters 231, special characters 7, labialized characters 44, punctuation marks 20 and numerals 20). But, due to time constraint, only 275 (231 core and 44 labialized) characters are considered. Besides, because of the above stated limitation, slant correction for Amharic characters was not considered in this study. As real-life character recognition is claimed to be very difficult due to many artifacts observed in characters taken from real-life documents, the research limit the experimentation to few number of documents. Accordingly, two page documents from ‘Addis Zemen’, Bible, ‘Federal Negarit Gazeta’, and ‘Fiker Eskemekabir’ sources are considered for training. For testing the recognition engine developed, a one page document is prepared from each source.

1.6. APPLICATION OF THE STUDY

A bulk of printed Amharic texts has been circulating every spot of our lives. Converting existing huge amount of documents (written in Amharic) into machine-readable form is not only time consuming, error-prone, and tedious but also it seems impossible in view of the magnitude of documents. OCR systems provide a tremendous opportunity especially in handling repetitive, boring, labor-intensive, error prone, and time consuming processes for human beings.

There are many potential applications of the proposed OCR system in this thesis work. To mention some:

- to facilitate minimized storage, fast indexing, full text retrieval and text modification of Amharic real-life documents in every automated government and private organizations.
- to save space in replacing huge collections of documents like in the National Library of Ethiopia etc.
- to make some modifications on the text around research institutions, universities etc.
- to electronically preserve irreplaceable materials in museums, archives etc.
- to facilitate office automation in government and private organizations.
- to transfer knowledge in the available printed Amharic documents first electronically and then at character level to the current generation.

There is no doubt that the development of Amharic OCR system has a tremendous contribution for the various organizations in Ethiopia when it comes to automate systems with a relatively small financial, efforts and time outlay.

1.7. ORGANIZATION OF THE THESIS

The thesis is organized into five chapters. The first chapter of the thesis includes the background, the statement of the problem and its justification. It also includes the objectives of the study, and discusses the methodology used to accomplish the research.

The second chapter discusses the basic methods of preprocessing techniques of character images along with the target printed real-life documents of this research. Different character recognition techniques reviewed from literature are also discussed.

The third chapter discusses the nature and characteristics of the Amharic writing system and reviews the different approaches used so far to develop an OCR system for the printed Amharic characters.

Chapter four deals with the experimentation activity undertaken to implement the methods and techniques described in chapter two. In this section of the paper, the success and difficulties faced while trying to implement the technique to the problem domain of interest is presented. The section also presents the recognition rate achieved for different test cases after the design and training of the neural network classifier.

Based on the results of the experiment, chapter five presents the conclusion and recommendations of the research.

CHAPTER TWO

REVIEW OF OCR SYSTEM

2.1. INTRODUCTION

Optical Character Recognition (OCR) is a process that allows printed (typewritten, printout as well as handwritten) text to be recognized optically and converted into machine-readable format. Technically, OCR comprises procedures such as scanning documents by scanning devices (handheld or flatbed scanners), segmenting each character in the scanned documents, extracting features of a character (which may help to differentiate the character from others), and match these features to the ones already stored to identify the character (Genovese, 1970).

In the following sections of this chapter the basic concept and preprocessing techniques of OCR systems are discussed. The most commonly used character recognition techniques are also presented with special emphasis on artificial neural networks.

2.2. BASICS OF OCR

OCR is performed by optical character readers which are automated system. OCR may be defined as the process of converting images of machine printed or handwritten numerals, letters, and symbols into a computer processable format. The long history of research in this area, commercial success, and the continuing need and ability to handle less restricted forms of text make OCR the most important application area in machine perception to date (Genovese, 1970).

In the 1960s a major initiative in postal number recognition was started. Gradually, there came success in the reading of postal numbers. At first, high performance of correct reading was not required, although further research and development was expected to improve the reading performance. This fortunate situation can be contrasted with speech recognition, in which essentially continuous speech recognition was required from the beginning. This level of difficulty corresponds to cursive script character recognition, which is a current topic of investigation. Thus, OCR has grown up as an industry, aided by the rapid development of computer technology. OCR technology is still at a stage of development (Ralston *et. al.*, 2000).

Ralston *et al.*, (2000) points out that an OCR system usually consists of modules for preprocessing, segmentation, and recognition. Digitization, skew detection, noise removal, binarization, slant correction, thinning, normalization, and others belong to the preprocessing stage of the overall recognition system. In the segmentation step, the text image is separated into individual characters. Two essential components in a character recognition step are the feature extraction and the classification.

The following discussion explores the three logical components of OCR systems: the preprocessing, segmentation and recognition.

2.3. PREPROCESSING TECHNIQUES

In this section of the thesis, the most widely used preprocessing techniques of recognition have been dealt. Their relevance to recognition lays in their ability to make the raw input data palatable to the recognition process. Digitization, noise removal, binarization, and thinning belong to the preprocessing stage of the overall recognition engine.

2.3.1. Digitization

Paper documents, which are and inherently analog medium, can be converted into digital form by a process of scanning and digitization. This process yields a digital image. Thus, the first phase in character recognition is digitization (Srihari and Lam, 1996).

Digitization process for the case of printed or handwritten documents is done using a device called optical scanner. This computer peripheral is capable of converting the hard copy into different format that range from true color intake to binary form scanning giving the facility of adjusting varying resolution for the image.

Recent advances in scanner technology have made available resolution in the range of 600 pixels per inch (ppi) to 1200 ppi. Recognition methods that use features (as opposed to template matching) use resolutions in the range of 200 ppi to 400 ppi, and careful consideration of gray scale. Lower resolutions and simple threshold tend to break thin lines or fill gaps, thus invalidating features (Ralston *et. al.*, 2000).

Scanners are capable of producing image representation in a variety of formats. One of the most common of these is the bit map (.BMP) format, which is dominantly used for the case of OCR. A scanned image is represented as an 8 bit per pixel or 256 level gray scales. This has to be converted to a 1-bit per pixel or monochrome to be used with the other preprocessing and recognition techniques that follow (Pandya and Macy, 1996).

2.3.2. Noise Detection and Removal

Noise is considered to be any measurement that is not part of the phenomena of interest (MathWorks, 2010). Scanned documents often contain noise that mainly arises due to printer, scanner, paper quality and age of the document. Noise removal refers to the removal of such noise on the image. Noise removal is a filtering method that eliminates the noise, enhance the quality of text characters and make the background texture uniform (Gonzalez and Woods, 2002).

Because of the level of degradations in digitized documents of magazines, books and newspaper in real-life Amharic documents, there is a need to apply noise filters so as to reduce the effect of degradation during the recognition process (Million and Jawahar, 2005). The most common types of noises in printed images are Gaussian and salt and pepper noise. Gaussian noise is characterized by each image pixel has value from a zero mean Gaussian distribution. Salt and pepper noises (also called impulse and speckle noise, or just dirt) consist of random pixels' being set to black or white (the extremes of the data range) (Karunanayaka *et. al.*, 2005).

Noise removal methods are divided into Linear filtering, Median filtering and Adaptive filtering (Ntogas and Ventzas, 2008). Linear, median and adaptive filtering methods are depicted in Table 2.1 below.

Linear filter	Median filter	Adaptive filter
The simplest linear filter is the mean filter. The intensity of every pixel in the image is replaced with the average value of intensity of its neighbor pixels. The new value of intensity of a pixel (i,j) of an image I is given by: $I(i,j)=\frac{1}{M}\sum_{(x,y)\in N} I(x,y)$ Where M represents the number of pixels in the neighborhood N.	Median filter is a non linear filter. For $A\{a1,a2,a3,....an\}$, and $a1 \le a2 \le a3 \le ... \le an \in R$ the new value of intensity of a pixel (i, j) of an image I is given by: $median(A)=\begin{cases} a_{\frac{n+1}{2}}, \text{if n is odd} \\ \frac{1}{2}\left(a_{\frac{n}{2}}+a_{\frac{n}{\frac{n}{2}+1}}\right), \text{if n is even} \end{cases}$	Adaptive filter particularly Wiener filter is applied to an image locally, by taking into account the local image variance. When the variance in an image is large the Wiener filter results in light local smoothing, while when the variance is small, it gives an improved local smoothing.

Table 2.1: Linear, Median and Adaptive filtering methods.

Mean filter and median filters are used to reduce Gaussian and salt-pepper noise respectively. But when these two noises exist in the image at the same time, using only one filter method can not achieve the wanted result. In this case Wiener filtering will perform better (Chi Chang-Yanab *et. al.*, 2005). Noise removal is performed before binarization. This is because Cheoi (2007) proved that if we use binarization algorithm without passing noise removal module, the quality of the binarized image would be very bad.

2.3.3. Binarization/Thresholding

After the process of noise removal, there are some sets of preprocessing steps that the digitized image should go through before the target blocks of the scanned document are fed to the OCR engine. One of these preprocessing steps is image binarization/thresholding. Practically, image binarization is the process of converting an image of up to 256 gray levels to a black and white image (Karunanayaka *et. al*., 2005). After the binarization process the digitized document will have only black and white pixels, which helps the OCR engine to deal with only on bi-level basis (black or white pixels). This preprocessing step enhances the performance of the recognizer since the engine will only be expected to handle binary data rather than a set of true colors.

Through the binarization of gray scale image, useful information for character segmentation and recognition as well for other steps in the recognition step may be lost (Seong-Whan Lee, *et. al.*, 1996). Thus, during binarization, one should be very careful not to lose such important ingredients of the character. For the purpose of binarizing the scanned document image, different thresholding techniques could be implemented that are used not to lose important information about the image (Wondwossen, 2004).

Binarization is used to segment an image from its background by setting all pixels whose intensity values are above a threshold to a foreground value and all the remaining pixels to a background value. The thresholding techniques that are used to discriminate an individual pixel as foreground or background pixel can generally be categorized into two major groups. One being global thresholding and the other being adaptive/local thresholding (Karunanayaka *et. al*. 2005).

In many cases of document processing, global thresholding techniques, particularly Otsu's method, were effectively applied to separate foreground and background of images because these methods can work well with variable illumination, shadows, smears and blurred documents (Chen and Leedham, 2005; Motwani et al, 2004). A brief description about Otsu thresholding algorithm is here under.

Otsu algorithm stores the intensities of the pixels in an array. The threshold is calculated by using total mean and variance. Based on this threshold value each pixel is set to either *0* or *1*. i.e background or foreground. Thus, the change of image takes place only once. Otsu uses the following formulas to calculate the total mean and variance (Otsu, 1979).

The pixels are divided into 2 classes,

C_1 with gray levels [1... t] and

C_2 with gray levels [t+1... L].

The probability distribution for the two classes is:

C_1 : (P_1/ W_1 (t)), ... ,(P_1/ W_1(t)) and

C_2 : ($Pt+_1$/ W_2 (t)), ... ,(P_L/ W_2(t))

Where $W_1(t)=\sum_{i=1}^{t} P_i$ and $W_2(t)=\sum_{i=t+1}^{L} Pi$

Also, the means of the two classes are

$$\mu_1 = \sum_{i-t}^{t} iP_i / W_1(t) \quad \text{and} \quad \mu_2 = \sum_{i-t+1}^{L} iP_i / W_2(t)$$

Using discriminant analysis, Otsu defined the between-class variance of the thresholded image as:

$$\sigma_B^2 = W_1(\mu_1 - \mu_t)^2 + W_2(\mu_2 - \mu t)^2$$

For bi-level thresholding, Otsu verified that the optimal threshold t* is chosen so that the between-class variance $_B$ is maximized; i.e.

$$t^* = \underset{t<L}{ArgMax} \{\sigma_B^2(t)\}$$

Ntogas and Ventzas, (2008) summarize the global thresholding with particular emphasis on Otsu's methods in Table 2.2.

Global thresholding	**Otsu's method**
The simplest implementation of thresholding is to choose an intensity value as a threshold level and the values below this threshold become 0 (black) and the values above this threshold become 1 (white). If T is the global threshold of image f(x,y) and the g(x,y) is the thresholding image, then $g(x,y) = \begin{cases} 1, & if \quad f(x,y) \geq T \\ 0, & otherwise \end{cases}$	Otsu's method applies clustering analysis to the grayscale data of input image and models two clusters of Gaussian distribution of pixels of the image. The optimal threshold minimizes the class variance of the two classes of pixels.

Table 2.2. Global thresholding with particular emphasis on Otsu algorithm.

Adaptive thresholding changes the threshold dynamically over the image. This change of threshold value is done for each specified area of the image. For each pixel in the image, applying adaptive thresholding, a threshold has to be calculated on the move from its neighborhood pixels. If the pixel value is below the threshold value computed uniquely for the

target pixel, it is set to the background value; otherwise it assumes the foreground value. Thus, for adaptive thresholding, a pixel will be categorized as one of the two possible pixel types, foreground or background, by consulting pixels found in its vicinity (Ntogas and Ventzas, 2008).

2.3.4. Thinning

Thinning is a process of reducing a binary image to a skeleton (one pixel thick) region without altering the shape and connectivity of the original figure (Mori and Yamamoto, 1992). “Thinning” and “skeletonization” have become almost synonymous in the literature (Lam *et. al.*, 1992). It is an important approach to represent the shape of a plane region. The objective of thinning is to reduce the representation of a region to a region of single pixel width while preserving all other relevant features (Pandya and Macy, 1996).

Hilditch (1969) is one of the originators of thinning, determined four conditions necessary for thinning to occur:

- Thinness: the thinned line’s width must be one pixel.
- Position: The thinned line must line along the center of the original line.
- Connectivity: the thinned line must keep the connectivity of the original line.
- Stability: at each step of the thinning process the thinned line can not be eroded away from its end points.

Lam *et. al.* (1992) stated wide range of thinning algorithms as iterative deletion of pixels and nonpixel-based methods.

i. Iterative Thinning Algorithms

Iterative thinning algorithms delete successive layers of pixels on the boundary of the pattern until only a skeleton remains. The deletion or retention of a (black) pixel *p* would depend on the configuration of pixels in a local neighborhood containing *p*. According to the way they examine pixels, these algorithms can be classified as sequential or parallel (Lam *et. al.*, 1992).

In a sequential algorithm, the pixels are examined for deletion in a fixed sequence in each iteration, and the deletion of *p* in the n^{th} iteration depends on all the operations that have been performed so far, i.e., on the result of the $(n-1)^{th}$ iteration as well as on the pixels already processed in the n^{th} iteration. On the other hand, in a parallel algorithm, the deletion of pixels in the nth iteration would depend only on the result that remains after the $(n-1)^{th}$; therefore, all pixels can be examined independently in a parallel manner in each iteration (Mori and Yamamoto, 1992).

ii. Nonpixel Based Thinning Algorithms

Nonpixel based thinning algorithms produce a certain median or center line of the pattern directly in one pass without examining all the individual pixels. They use image morphology i.e., the structure of objects with in an image. Morphological process use set theory operations, such as intersections (AND), UNION (OR), and complement (NOT), to combine the pixels logically into a resulting pixel value. Thinning algorithms based on hit-and-miss transforms method is the popular type of nonpixel based thinning algorithms (Lam *et. al.*, 1992).

Aishy (2002) discusses about hit-and-miss algorithm. The thinning operation makes use of a structuring element. These elements are of the extended type i.e., they can contain both ones and zeros. The thinning operation is the hit-and-miss transform. The thinning of an image I by a structuring element J is:

thin (I , J) = I- *hit-and-miss* (I , J)

Where the subtraction is a logical subtraction defined by X-Y = X NOT Y

The origin of the structuring element is translated to every possible pixel position within the image. At each position the element is compared to the underlying image. If the structuring element and the image exactly match, the image pixel underneath the origin of the structuring element is set to zero (background). If they don't match it is left unchanged. The origin of the element must never be zero but it can be either blank or one.

Direct implementation of hit-and-miss of a region is typically an expensive computational process, as it involves comparing every possible pixel position within the image to the underlying image. To increase the efficiency of these calculations, hit-and miss thinning algorithms have been designed that iteratively delete edge points on the region to the constraints that deletion of these points does not remove end points, does not break connectivity and does not cause excessive erosion of the region (Aishy, 2002).

The thinning algorithm used in the MATLAB is based on the technique of hit-and-miss algorithms that iteratively delete edge points on the region. In this type of thinning method, for every iteration, the edge pixels having at least one adjacent background point are deleted. It is an

iterative algorithm, which erodes the outer layers of pixel until no more layers can be removed (MathWorks, 2010).

According to Lam *et. al.* (1992), the nonpixel-based thinning methods that iteratively delete edge points on the region are efficient in terms of the number of operations required. This procedure is useful in applications where the detection of such points would suffice, one possible example being feature extraction in OCR.

2.4. SEGMENTATION

Once pages are preprocessed, then they are segmented into individual components. Segmentation is an operation that seeks to decompose an image into sub-images of individual symbols (Casey and Nagy, 1982). Reliable character segmentation and recognition depend up on both document quality and registered image quality (Ralston *et. al.*, 2000). Correct recognition of a character, in fact, depends on correct segmentation of a character by a system. Hence, segmentation is one of the important part of the recognition system and due consideration should be given in the selection of segmentation algorithm.

There are basically two commonly used segmentation algorithms: stage by stage and recursive segmentation (Million, 2000). In the stage by stage segmentation algorithm, a text is segmented in three successive steps: line segmentation, word segmentation, and character segmentation. Line segmentation relates to the process of separating text lines from a given scanned image of a document. Word segmentation, on the other hand, involves separating words from a given line of text. Finally, character recognition is the identification of characters from a given word. Such an algorithm is originally suggested by Pal and Chaudhuri (1995) for use in the character

recognition of printed Bangla script and not affected by noise. It shows better result if the scanned image is not skewed.

Though stage by stage algorithm is successful in line segmentation, it did not perform well for segmenting connected characters (characters having no white space that separated them) (Dereje, 1999). The algorithm considered all connected characters as one character image and hence led to misclassification of the characters by the recognition algorithm. In this case, recursive segmentation algorithm is the ultimate choice. Recursive segmentation is an approach that merges segmentation and recognition together (Casey and Nagy, 1982). It is said to be convenient for characters of connected nature (Dereje, 1999). Recursive word recognition systems work by looking for ligatures (marks that join the individual characters) and separating the individual characters that makes up the word at these ligatures. The characters are then processed, and the closest match in the lexicon is found. In this approach, once the document has been converted into a large binary data by optical scanning, that algorithm first discriminate text line. It then partitions each text line into a sequence of pattern arrays at places where there is sufficient white space between successive black regions. At the end of this process touching characters will be segmented as one pattern (but if a character was not touched by its neighbors, it will be segmented correctly) (Ralston et. al, 2000).

The segmented components comprise of the complete set of Amharic character sets in this work. All segmented characters are sent to the normalization module for scaling the characters to a manageable size for the recognizer.

2.5. NORMALIZATION

Normalization is the process of enlarging and shrinking the character size. In this step, the characters are reduced (or in the unlikely event that it is smaller than the target size, it is enlarged) to fit a target size (Ralston *et.al.,* 2000). The normalizing of characters is needed in order to make the recognition operations process independent of the printing size (Ralston *et.al.,* 2000) and to scale the characters to a manageable size for the recognizer (Pandya and Macy, 1996). Hence, the algorithms for recognition operations perform faster on small images, and the topology of the neural networks is simplified (Gonzalez and Woods, 2002). The amount of normalization is application specific. In the context of character recognition problem, Million and Jawahar (2005) claimed 20×20 is the standard size. The final result of size normalization process is a set of character components that are scaled to a standard size. This data set is used as an input for the recognizer.

According to Min *et al*., (2005), among the normalization methods interpolation is widely used as a weighted average of some set of pixels in the vicinity of the point to produced better output. Interpolation works by using known data to estimate values at unknown points. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's color and intensity based on the values at surrounding pixels. To determine the value for an interpolated pixel, we need to find the point in the input image that the output pixel corresponds to. We then assign a value to the output pixel by computing a weighted average of some set of pixels in the vicinity of the point. The weightings are based on the distance each pixel is from the point.

Common interpolation algorithms can be grouped into two categories: adaptive and non-adaptive Min *et al*., (2005).

Min *et al*., (2005) stated about the two categories of interpolation algorithms that adaptive interpolation algorithms change depending on what they are interpolating (sharp edges vs. smooth texture). Many of these algorithms apply their algorithm (on a pixel-by-pixel basis) when they detect the presence of an edge- aiming to minimize unsightly interpolation artifacts in regions where they are most apparent. Non-adaptive interpolation algorithms treat all pixels equally. They include nearest neighbor, bilinear, and bicubic interpolation algorithms. Depending on their complexity, these use anywhere from *0* to *1* adjacent pixels when interpolating. The more adjacent pixels they include, the more accurate they can become, but this comes at the expense of much longer processing time.

Nearest neighbor is the most basic and requires the least processing time of all the interpolation algorithms because it only considers one pixel- the closest one to the interpolated point. This has the effect of simply making each pixel bigger. Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. For character recognition problem domain, bilinear interpolation algorithm produces better images in producing much smooth looking images than nearest neighbor (Min et.al, 2005; MatWorks, 2010). Bicubic goes one step beyond bilinear by considering the closest 4x4 neighborhood of known pixels for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. This condition takes much longer processing time.

2.6. CHARACTER RECOGNITION

Two essential components in a character recognition algorithm are the feature extractor and the classifier. Feature analysis determines the descriptors, or feature set, used to describe all characters. Given a character image, the feature extractor derives the features that the character possesses. The derived features are then used as input to the character classifier (Ralston *et. al.*, 2000).

Character recognition problem is a subset of the more general pattern recognition problem. Therefore, it is worth to briefly mention about the basic components in character recognition algorithms: feature extractor and the classifier.

2.6.1. Feature Extraction

Extraction follows the segmentation phase of OCR where the individual characters are considered and extracted for features. Image features are unique characteristics that can represent a specific image. Image features are meaningful, detectable parts of the image. Meaningful means the features are associated to interesting elements via the image formation process. Detectable means the extraction algorithm corresponding to the feature must exist, otherwise the feature is useless. Two types of characteristics are usually referred by image features (Mori and Yamamoto, 1992):

- a global property of an image, for instance the average gray level of all pixels included in a gray level image.
- a part of the image with special properties, for example the boundary length of an image.

Different features are associated with different extraction algorithms that output collections of the feature descriptors. Good image features should satisfy the following conditions (Qing 2003):

- Robust to transformations – the image features should be as invariant as possible to image transformations including translation, rotation, and scaling, etc.
- Robust to noise – the image features should be robust to noises and various degraded situations.

The selection of image features and corresponding extraction methods is an important step in achieving high performance for an OCR system. At the same time, the image feature and the extraction method also decide the nature and the output of the image-preprocessing step. Some image features and the extraction algorithms work on color images, while others work on gray level or binary images. Moreover, the format of the extracted features must match the requirements of the classifier. Some features like the graph descriptions and grammar-based descriptions are well suited for structural and syntactic classifiers. Numerical features are ideal for statistical classifiers and discrete features are ideal for decision trees (Qing, 2003).

2.6.2. Classification

Classification is done using the features extracted which corresponds to each character. These features are analyzed using different classification approaches and labeled as belonging to different classes. This classification is generalized such that it works for all the fonts' types, styles and sizes (Qing, 2003).

There are four basic classification approaches in the pattern recognition literature. Since OCR is one of pattern recognition problem, these methods can be implemented in OCR. These methods are (Qing, 2003):

a) Syntactic/Structural Pattern Classification

b) Statistical Pattern Classification

c) Mathematical Classification

    c.i) Template Matching

    c.ii) Artificial Neural Networks

A brief discussion will follow in the following section about the four basic classification approaches.

a) Syntactic/Structural Classification

Structural classification methods use structural features and decision rules to classify characters. Structural features may be defined in terms of character strokes, character holes, or other character attributes such as concavities. For instance, the letter "९" may be described as a vertical stroke with a loop attached to the upper right side. For a character image input, the structural features are extracted and a rule-based system is applied to classify the character. Structural methods are also trainable, but construction of a good feature set and a good rule-based classifier can be time-consuming (Ralston *et. al.*, 2000).

Syntactic pattern classification utilizes the underlying structure of the patterns themselves. In some cases the significant information lies not in the feature vectors themselves but in their interrelationships, thus it is the interconnection of features which assist the classification process. For classification it is necessary to quantify and extract structural information and to assess structural similarities of patterns; at the lowest level this structural information will be primitive elements or building blocks out of which more complex elements of the image are constructed. One approach to this structural pattern relationship is to adopt the syntax structure approach of a formally defined language (Pandya and Macy, 1996).

b) Statistical Pattern Classification

Statistical pattern recognition relies on defining a set of decision rules based on standard statistical theory. A set of characteristics features measurements are extracted from the input image data which are then combined to define a feature vector identifying an object belonging to one defined pattern class. The decision rules for this assignment are derived either from a priori knowledge of the expected distribution of the pattern classes or from knowledge acquired through a training process involving many initial measurements (Pandya and Macy, 1996).

c) Mathematical Classification

Many character recognizers are based on mathematical formalisms that minimize a measure of misclassification. These recognizers may use pixel-based features or structural features. Some examples are discriminant function classifiers, Bayesian classifiers, artificial neural networks (ANNs), and template matching. Discriminant function classifiers use hypersurfaces to separate the featural description of characters from different semantic classes and in the process reduce

the mean-squared errors. Bayesian methods seek to minimize the loss function associated with misclassification through the use of probability theory. ANNs, which are closer to theories of human perception, employ mathematical minimization techniques. Both discriminant functions and ANNs are used in commercial OCR systems (Ralston *et. al.*, 2000).

c.i) Template Matching

Template matching or matrix matching, is one of the most common classification methods under mathematical classification. In template matching, individual image pixels are used as features. Classification is performed by comparing an input character image with a set of templates (or prototypes) from each character class. Each comparison results in a similarity measure between the input character and the template. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ, the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character's identity is assigned as the identity of the most similar template. Template matching is a trainable process because template characters may be changed (Ralston *et. al.*, 2000).

c.ii) Artificial Neural Networks

This approach uses architecture, which can be trained and correctly associate input patterns with desired responses. This alternative approach, using artificial neural networks, attempts to exploit knowledge of how biological neural systems store and manipulate information. Use of neural networks or artificial neural systems, offer a new computing paradigm in which the network,

through a process of learning from task examples can store experimental knowledge and make it available for use at a later time. The detail is discussed in the next section.

2.7. ARTIFICIAL NEURAL NETWORK

The human brain performs perceptual tasks such as visual pattern recognition distinguishing acoustical harmonics, and speech understanding remarkably well. Such cognitive tasks remain difficult for digital computers to accomplish. The promise of neural computing relies on the rapid solution of such problems through massive parallelism, where information is not transferred between computing units, but is encoded through patterns of interconnectivity in a distributed fashion. The study of neural networks includes the notions of connectionism, parallel distributed processing, self-adaptive systems, and self-organizing systems as discussed below (Ralston et.al, 2000).

2.7.1. Definition

A neural net work is a massively parallel distributed processor that has a natural propensity for storing knowledge and making it available for use. It resembles the brain in two respects: i) Knowledge is acquired by the network through a learning process ii) Interneuron connection strength known as synaptic weights are used (Haykin, 1994).

2.7.2. Why Neural Networks?

According to Haykin (1994), it is apparent that the neural network derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize. Generalization refers to the neural network producing reasonable outputs for inputs not encountered during training (learning). These two information-processing capabilities make it possible for neural networks to solve complex (large scale) problems that are currently intractable. In addition, Taylor (1995) wrote the prons of ANNs as:

- They can be trained to classify poorly structured inputs;
- They are robust against noise in training data;
- They can be used in hybrid neural net/artificial intelligence systems;
- A hardware system is possible.

In practice, however, neural networks can not provide the solution working by themselves alone. Rather, they need to be integrated into a consistent system engineering approach. Specifically, a complex problem of interest is decomposed into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks (e.g. pattern recognition, associative memory, control) that match their inherent capabilities (Haykin, 1994).

2.7.3. Analogy with Biological Neural Network

A *neuron* (or nerve cell) is a special biological cell that processes information (see Figure 2.1). It is composed of a cell body, or *soma,* and two types of out-reaching tree-like branches: the *axon* and the *dendrites.* The cell body has a nucleus that contains information about hereditary traits

and plasma that holds the molecular equipment for producing material needed by the neuron. A neuron receives signals (impulses) from other neurons through its dendrites (receivers) and transmits signals generated by its cell body along the axon (transmitter), which eventually branches into strands and sub strands. At the terminals of these strands are the *synapses.* A synapse is an elementary structure and functional unit between two neurons (an axon strand of one neuron and a dendrite of another), When the impulse reaches the synapse's terminal, certain chemicals called neurotransmitters are released. The neurotransmitters diffuse across the synaptic gap, to enhance or inhibit, depending on the type of the synapse, the receptor neuron's own tendency to emit electrical impulses. The synapse's effectiveness can be adjusted by the signals passing through it so that the synapses can *learn* from the activities in which they participate. This dependence on history acts as a memory, which is possibly responsible for human memory (Jain *et.al.*, 1996).

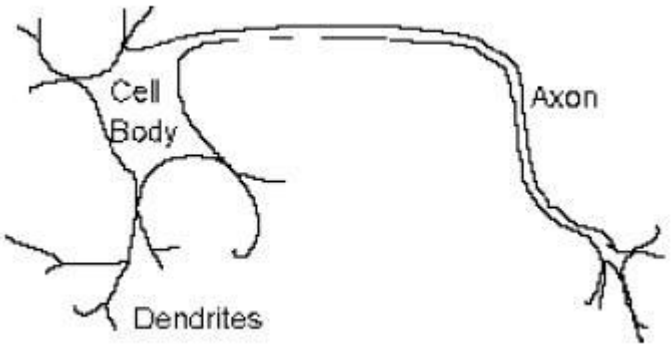


Figure 2.1. A sketch of a biological neuron.

The cerebral cortex in humans is a large flat sheet of neurons about 2 to 3 millimeters thick with a surface area of about 2,200 cm^2, about twice the area of a standard computer keyboard. The cerebral cortex contains about 10^{11} neurons, which is approximately the number of stars in the

Milky Way. Neurons are massively connected, much more complex and dense than telephone networks. Each neuron is connected to 10^3 to $l0^4$ other neurons. In total, the human brain contains approximately 10^{14} to $l0^{15}$ interconnections (Jain *et.al.*, 1996).

With the neurobiological analogy as the source of inspiration, and the wealth of theoretical and technological tools that we are bringing together, it is for certain that in our understanding of artificial neural networks will be much more sophisticated than it is today (Haykin, 1994). Artificial neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the network is usually implemented using electronic components or stimulated in software on a digital computer (Haykin, 1994).

McCulloch and Pitts (1993) proposed a binary threshold unit as a computational model for an artificial neuron (see Figure 2.2). This mathematical neuron computes a weighted sum of its *n* input signals, x_j, j = 1,2, . . . , n, and generates an output of 1 if this sum is above a certain threshold *u*. Otherwise, an output of 0 results. Mathematically,

$$Y = \theta\left[\sum_{j=1}^{n} w_j x_j - u\right]$$

where θ (.) is a unit step function at 0 and w_j is the synapse weight associated with the j^{th} input.

For simplicity of notation, McCulloch and Pitts (1993) consider the threshold *u* as another weight $w0 = u$ attached to the neuron with a constant input $x_0 = 1$. Positive weights correspond to excitatory synapses, while negative weights model inhibitory ones. McCulloch and Pitts (1993) proved that, in principle, suitably chosen weights let a synchronous arrangement of such neurons perform universal computations.

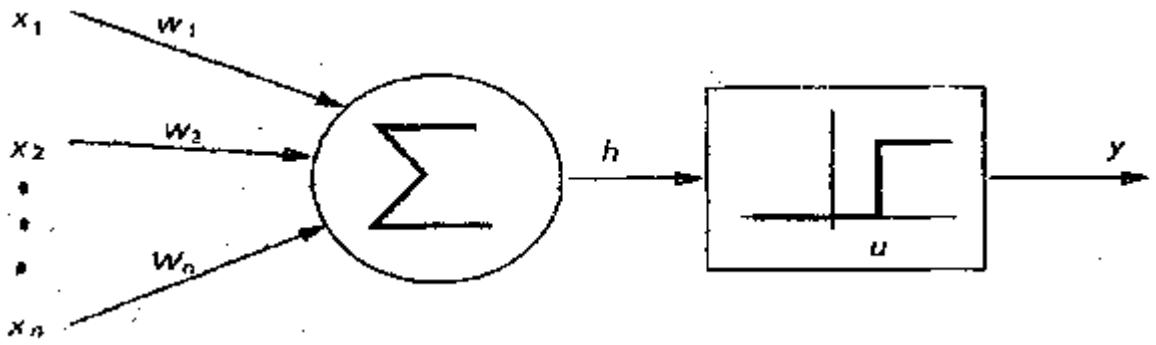
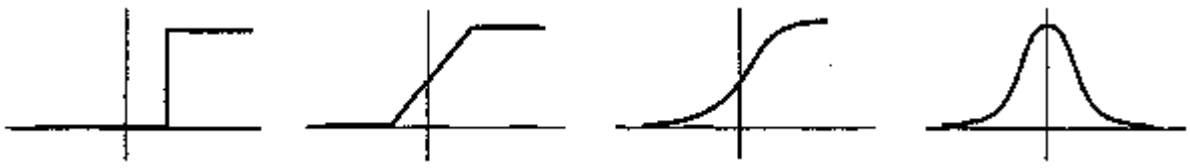


Figure 2.2: The McCulloch-Pitts neural model.

According to (Jain *et.al.*, 1996), there is a crude analogy between artificial neuron and biological neuron: wires and interconnections model axons and dendrites, connection weights represent synapses, and the threshold function approximates the activity in a cell body.

2.7.4. Activation Function

The activation function defines the output of a neuron in terms of the activity level at its input (Haykin, 1994). The McCulloch-Pitts (1993) neuron has been generalized in many ways. An obvious one is to use activation functions other than the threshold function, such as linear, sigmoid, or Gaussian, as shown in Figure 2.3 (Jain *et.al.*, 1996).



(a) (b) (c) (d)

Figure 2.3. Types of Activation Functions: (a) threshold, (b) linear, (c) sigmoid, and (d) Gaussian.

In the threshold activation function threshold, the output is set at one or two levels, depending on whether the total input is greater than or less than some threshold value. For the linear activation function, the output activity is proportional to the total weighted output (Jain *et.al*., 1996). The sigmoid activation function, the output varies continuously but not linearly as the input changes. The sigmoid function is by far the most common form of activation function used in the construction of artificial neural networks (Haykin, 1994). It is defined as a strictly increasing function that exhibits smoothness and asymptotic properties. The standard sigmoid function is the logistic function, defined by $g(x) = 1/(1+\exp\{-\beta x\})$, where β is the slope parameter. In the Gaussian function the maximal function value of a gauss function is found for zero activation. The function is even: $f(-x) = f(x)$. The function value is decreasing with increasing absolute value of activation.

2.7.5. Network Architecture

The commonest type of ANN consists of three groups, or layers, of units: a layer of input units is connected to a layer of hidden units, which is connected to a layer of output units (Stergiou and Siganos, 2002). The activity of the input units represents the raw information that is fed into the network. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units. The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units (Haykin, 1994).

Architecturally, ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron

inputs. Based on the connection pattern (architecture), ANNs can be grouped into two categories: feed forward and feed back/recurrent networks (Jain *et.al.*, 1996) (see Figure 2.4).

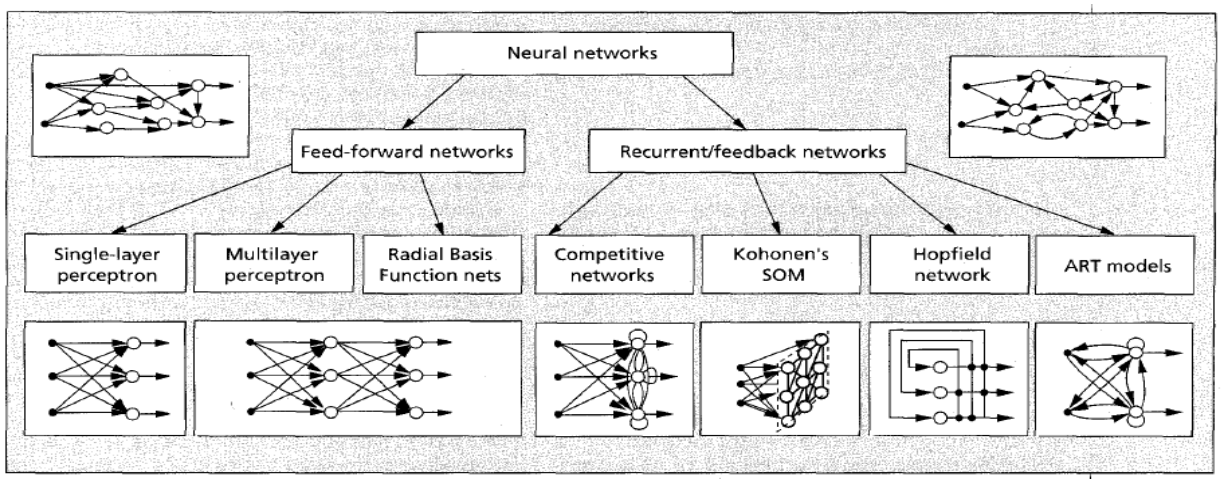


Figure 2.4. A taxonomy of feed-forward and recurrent/feedback network architectures.

Different connectivity yield different network behavior. Generally speaking, a recurrent neural network distinguishes itself from a feed forward neural network in that it has at least one feedback loop. Figure 2.4 shows typical networks for each category.

a) Feed Forward Networks

Feed forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops), i.e., the output of any layer does not affect that same layer. Feed forward ANNs tend to be straight forward networks that associate inputs with output. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down (Stergiou and Siganos, 2002). Haykin (1994) identifies two different classes of feed-forward networks: single-layer and multilayer feed forward networks.

a.i) Single-Layer Feed Forward Networks

A layered neural network is a network if neurons organized in the form of layers. In the simplest form of a layered network, input layer of source nodes project on to an output layer of neurons (computational nodes), but not vice versa (Haykin, 1994).In other words, this network is strictly of a feed forward type. It is illustrated in Figure 2.5 for the case of four nodes in both the input and output layers. Such a network is called a single-layer network, with the designation "single layer" referring to the output layer of computation nodes (neurons). In other words, we do not count the input layer of source nodes, because no computation is performed there (Haykin, 1994). In such cases, the network associates an output patterns (vector) with an input pattern (vector) and information is stored in the network by virtue of modifications made to the synaptic weights of the network (Jain *et.al.*, 1996).

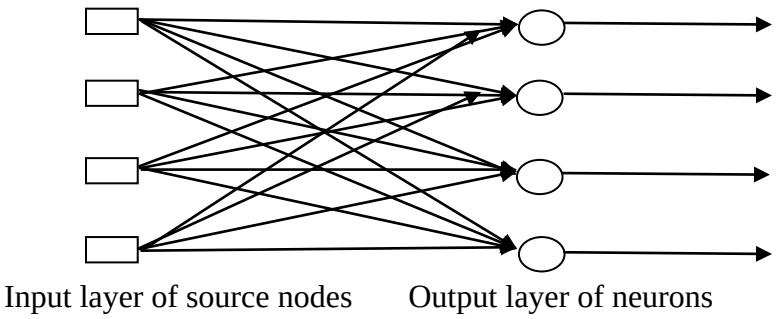


Figure 2.5. Feed forward network with a single layer of neurons.

a.ii) Multilayer Feed Forward Networks

The second class of a feed forward neural network distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons or

hidden units. The function of the hidden neurons is to intervene between the external input and the network output. By adding one or more hidden layers, the network is enabled to extract higher-order statistics, for the network acquires a global perspective despite its local connectivity by virtue of the extra set of synaptic connections and the extra dimensions of neural interactions. The ability of hidden neurons to extract higher-order statistics is particularly valuable when the size of the input layer is large (Haykin, 1994).

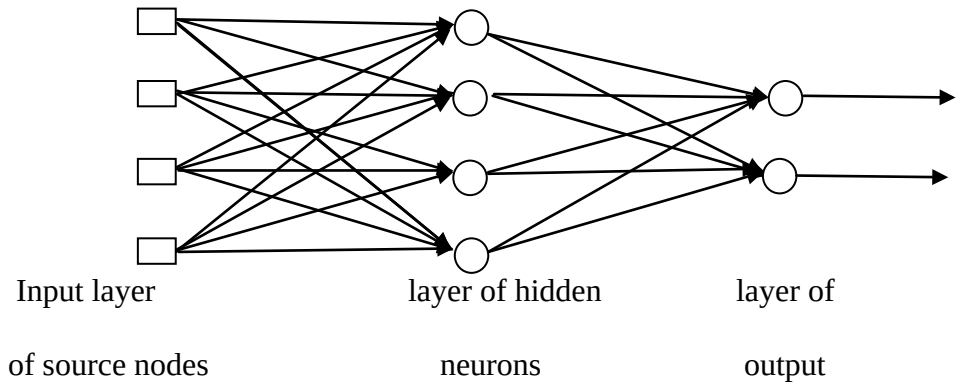


Figure 2.6. Feed forward network with one hidden layer and output layer.

The architectural graph of Figure 2.6 illustrates the layout of a multilayer feed forward neural network for the case of a single hidden layer. The network of Figure 2.6 is referred to as a 4-4-2 network in that it has 4 source nodes, 4 hidden neurons, and 2 output neurons. It is said to be fully connected in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer (Haykin, 1994).

b) Feedback (Recurrent) Networks

A recurrent neural network distinguishes itself from a feed forward neural network in that it has at least one feedback loop. For example, a recurrent network may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons, as illustrated in the architectural graph of Figure 2.7.

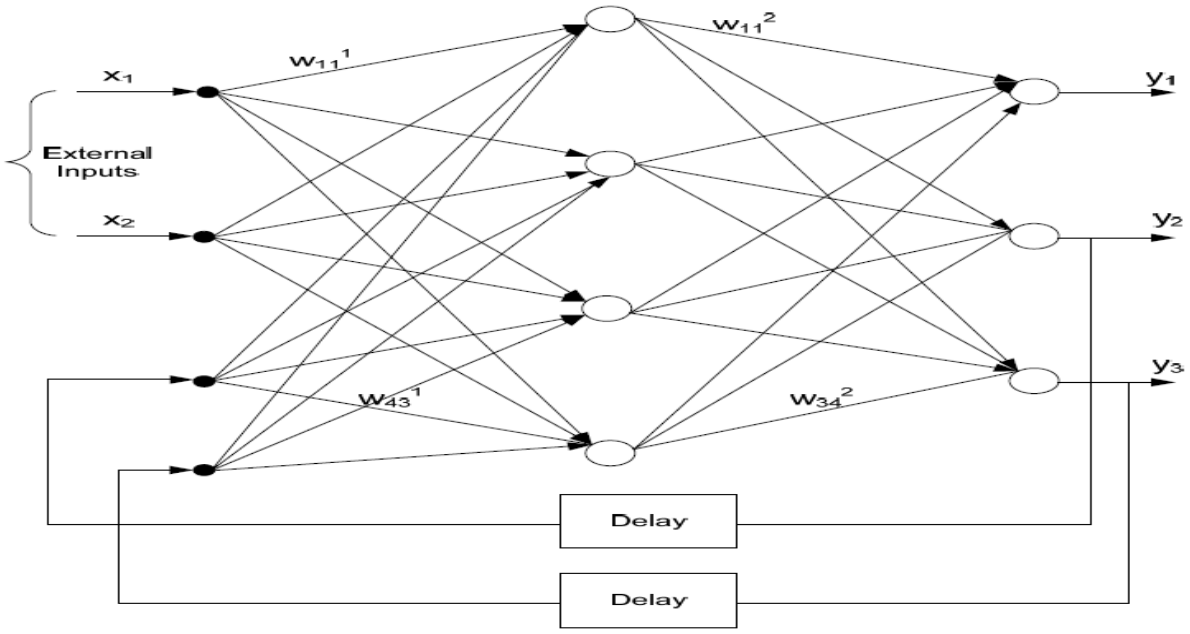


Figure 2.7. Recurrent network.

Feed back networks are very powerful and can get extremely complicated. Feed back networks are dynamic; their state is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feed back connections in single-layer organizations (Stergiou and Siganos, 2002).

Different network architectures require appropriate learning algorithms. The next section provides an overview of learning processes.

2.7.6. The Learning Process

Although a precise definition of learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. The process of determining or adjusting parameters on the basis of the environment (data set) is referred to as learning or training. ANNs' ability to automatically learn from examples makes them attractive and exciting. Instead of following a set of rules specified by human experts, ANNs appear to learn underlying rules (like input-output relationships) from the given collection of representative examples. This is one of the major advantages of neural networks over traditional expert systems (Jain *et.al.*, 1996).

To understand or design a learning process, we first have a model of the environment in which a neural network operates, that is, we must know what information is available to the network. This is referred to this model as a learning paradigm. Second, we must understand how network weights are updated, that is, which learning rules govern the updating process. A learning algorithm refers to a procedure in which learning rules are used for adjusting the weights (Stergiou and Siganos, 2002; Jain *et.al.*, 1996; Haykin, 1994).

There are three main learning paradigms: supervised, unsupervised, and hybrid (Jain *et.al.*, 1996).

In supervised learning, or learning with a “teacher”, the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. A disadvantage of supervised learning is the fact that without a teacher, a neural network cannot learn new strategies for particular situation that are not covered by the set of examples used to train the network. This limitation can be overcome by the use of reinforcement learning (Haykin, 1994). Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. Reinforcement learning is performed through a process of trial and error.

In contrast, unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Once the network has become tuned to the regularities of the input data, it develops the ability to form internal representations for encoding features of the input and there by create new classes automatically (Jain *et.al*., 1996).

Hybrid learning combines supervised and unsupervised learning. Part of the weights is usually determined through supervised learning, while the others are obtained through unsupervised learning.

Learning process in ANNs must address three fundamental and practical issues associated with learning from samples: capacity, sample complexity, and computational complexity. Capacity concerns how many patterns can be stored, and what functions and decision boundaries a

network can form. Sample complexity determines the number of training patterns needed to train the network to guarantee a valid generalization. Computational complexity refers to the time required for a learning algorithm to estimate a solution from training patterns (Jain *et.al*., 1996).

The error correction rule is the popular type of learning rule (Jain *et.al*., 1996). The error correction rule in the ANNs learning paradigm, the network is given a desired output for each input pattern. During the learning process, the actual output *y* generated by the network may not equal the desired output *d*. The basic principle of error-correction learning rules is to use the error signal *(d-y)* to modify the connection weights to gradually reduce this error (Haykin, 1994).

The multi layer feed forward network needs to be equipped with a good learning algorithm to model the approximation of the real neural system. Backpropagation is then the algorithm widely used for the multi layer feed forward network training (Wondwossen, 2004).

2.7.7. The Back Propagation Algorithms

Multilayer perceptions have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the back-propagation algorithm (see Table 2.3). This algorithm is based on the error-correction learning rule. The development of the back-propagation algorithm represents a “landmark” in neural networks in that it provides a computationally efficient method for the training of multilayer perceptions. Back propagation algorithm has been applied successfully to solve some difficult problems such as speech recognition from text, handwritten-digit recognition and adaptive control (Haykin, 1994). The back propagation algorithm is the most widely used method for determining the error derivative of the weights (Stergiou and Siganos, 2002) as seen

in Figure 2.3. According to Jain *et.al.* (1996), there are many issues in designing feed forward networks including:

- How many layers are needed for a given task,
- How many units are needed per layer,
- How will the network perform on data not included in the training set (generalization ability), and
- How large the training set should be for "good" generalization.

Backpropagation Algorithm

The back-propagation algorithm consists of the following steps:

1. Initialize all the weights to small random values.
2. Randomly choose an input pattern $X^{(Li)}$.
3. Propagate the signal forward through the network.
4. Compute δ_i^L in the output layer ($O_i = Y_i^L$).

   $\delta_i^L = g'(h^L_i)\ [d_i^L - y_i^L]$, where h^L_i represents the net input to the i^{th} in the L^{th} layer, and g' is the derivative of the activation function g.
5. Compute the deltas for the preceding layers by propagating the errors backwards;

   $\delta_i^j = g'(h^j_i) \sum_j W_{ij}^{l+1} \delta_j^{l+1}$, for l =(L-1),…,1.
6. Update weights using

   $\triangle W^l_{ji} = \eta\ \delta_i^j\ y^{l-1}_j$
7. Go to step to and repeat for the next pattern until the error in the output layer is below a prespecified threshold or a maximum number of iterations are reached.

Figure 2.8. Back propagation Algorithm

2.7.8. Neural Network Parameters

2.7.8.1. Initializing weights

All neural network training algorithms begin by initializing the weights in the network to some randomly chosen values. A suitable choice of initial weights is therefore potentially important in allowing the training algorithm to produce a good set of weights, and in addition may lead to improvements in the speed of training. For multilayer perceptions having sigmoidal hidden unit activation function, the majority of initialization procedures in current use involve setting the weights to randomly chosen small values. Random values are used in order to avoid problems due to symmetries in the network. The initial weight values are chosen to be small so that sigmoidal activation functions are not driven into the saturation regions. If the weights are too small, however, all of the sigmoidal lead to slow training. This suggests that the summed inputs to the sigmoidal activation functions will be approximately linear, which can again lead to slow training. This suggests that the summed inputs to the sigmoidal functions should be of order unity. A random initialization of the weights requires that some choice be made for the distribution function from which the weights are generated. The weights are usually generated from a Gaussian distribution with zero mean and variance σ in the order of *(1/d1/2);* where *d* is the number of input nodes /inputs.

2.7.9. ANN learning and Generalization

The goal of neural network training is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generates the data. This is important if the network is to exhibit good generalization, that is, to make good predictions for new inputs.

CHAPTER THREE

THE AMHARIC WRITING SYSTEM

3.1. INTRODUCTION

Understanding the attribute of the character of a language and the allowed combinations between them to make up words, phrases and sentences is helpful to extract features as well as fix a set of rules (e.g. dictionary) specific to that language. OCR heavily depends on the writing system of the language. Character positioning, text orientation, word segmentation, and character forms are very essential elements in OCR. Hence, understanding the language in question and its writing system are indispensable steps in OCR.

Thus, knowledge of the Amharic language and its writing system are very essential either to develop or adopt an OCR algorithm to the Amharic script. This chapter will explain the Amharic writing system in general and the distinguishing features of real-life printed Amharic documents in particular for the context of this work. For further reading on the Amharic writing system and the attempts made on printed characters of Amharic OCR one can refer the following works: (Worku, 1997; Ermias, 1999; Dereje, 1999; Birhanu, 1999; Million, 2000; Yaregal, 2002).

3.2. THE AMHARIC WRITING SYSTEM

3.2.1. Evolution

The major tool for transferring knowledge to future generation is writing. Writing started before the birth of Christ. The first writing system was in the form of picture drawing that represents events and objects. In this pictorial writing system, Egyptians are those to mention who used to draw or paint on monuments, walls or rocks in and after 3000BC (Dereje, 1999). Egyptians used pictures writing called Hieroglyphic on monuments, walls and rocks (Bender *et. al.*, 1976).

The ancient Egyptians language, which is the first written language, is one family of Afroasiatic language family. The ancient Semitic language is also a family of Afroasiatic. Amharic language is a descendant of the ancient Semitic family (Bender *et.al.*, 1976).

The Amharic writing system was adopted from that of Geez. Geez, in turn, was taken from one of the ancient south Arabian languages (Worku, 1997). By the 16th century, Geez gradually gave way to Amharic. During that time, Amharic was spoken at the royal court, and began to be used for literary purposes at the beginning of the 19th century as the administrative state changed it way of communication from oral to written one (Yaregal, 2002).

Different writing systems or scripts represent linguistic units, words, syllables and phonemes, at different structural levels. Each script has its own set of icons, which are known as characters or letters, which have certain basic shapes (Srihari and Lam, 1996).

3.2.2. The Amharic Characters (□□□)

The Amharic writing system, by the time when it took over the place of Geez, took all the 26 Geez symbols and added several new ones to represent sounds not found in Geez. The additional symbols are □, □, □, □, □, □ and □. Since the 1st century, modifications and additions on Amharic characters has been undergone and current Amharic script has 33 basic characters, one special symbol in its 7 different forms to represent the | V| sound found in Latin based languages, 44 labial symbols, e.g. □,□,□, 8 punctuation marks and 20 symbols for numerals which make up a number of characters in Amharic writing to be greater than 330 (Bender *et.al.,* 1976). Annex I. lists the complete set of symbols used in the Amharic writing system.

3.2.2.1. The Basic Characters

The current Amharic writing system consist of a core of 33 characters (□□□, Fidel) each of which occurs in a basic form and in six other forms know as orders (Nigussie, 2000). The non-basic forms are derived from the basic forms by modifications. Thus, there are 231 different characters. The seven orders represent syllable combinations consisting of consonant and following vowel. This characteristic, according to Bender *et. al*. (1976), makes the Amharic writing system a syllabic writing system. A character or a symbol is used to represent a phoneme, which is a combination of a vowel and a consonant.

In a syllabic system, like Amharic, the number of characters (symbols) needed by the language is determined by the number of basic sounds used (Bender *et. al.,* 1976). In addition to the 231 characters there are nearly 40 others which contain a special feature usually representing

labialization, for example, □ from □ and □ from □. Only about twenty of these are common and are usually listed as an appendix to the main list (Worku, 1997; Million, 2000).

3.2.2.2. The Punctuations

The Amharic writing system consists of as many as 8 punctuation marks in addition to the characters. However, only few of them are practically used, especially in computer-written system. The word-separator (Hulet Neteb), two square dots arranged like colon, **:** , and sentence-separator, four square dots arranged in a square pattern, **::** , are the basic punctuation marks in Amharic writings system. Hulet Neteb is oddly used more in hand written practices today than in modern typesetting. Its place is almost completely taken over by space (Worku, 1997).

Lists in Amharic text are separated by an equivalent to comma, 'Netela Sereze' (፣) followed by ASCII space and 'Derib Sereze' (፤) which is the equivalent of semi-colon, may also be found in use as a list separator. In addition to these, the writing system has borrowed some punctuation marks from foreign languages. For instance, the exclamation mark '!' and the question mark '?' are used in the language (Bender *et. al.,* 1976).

3.2.2.3. The Numbers

The Amharic number system consists of twenty single characters. They represent numbers one to ten, multiples of ten (twenty to ninety), hundred, and thousand. These characters are derived from Greek letters (Bender *et. al.,* 1976), and in order to make them look like the Amharic characters the symbols are modified by adding a horizontal stroke above and below.

Both Amharic and Western numerals are in use today. Though the Amharic has long since been retried to a reserved use primary for calendar dates and demarcation of sections in literature, while Western numerals are used everywhere else following western practices.

3.2.3. Features of Amharic Characters

In a nutshell, the Amharic writing system has the following basic characteristics (Bender *et. al.,* 1976; Nigussie, 2000):

- Each symbol is written according to the sound that goes with it. The vowels are integrated in the characters by modifying the base characters in some form, which together represent syllable combinations consisting of a consonant and vowel. Thus the Amharic writing system is often called syllabic rather that alphabetic.
- The symbols are written in a disconnected manner, e.g. □, □, □, □, □.
- Concerning the direction of writing the characters, while Sabaen characters are written right to left, Geez and Amharic and any language that uses the script for writing is written from left to right.
- The lines of text are read left-to-right and the lines are read in a top-to-bottom sequence.
- There is a proportional spacing between characters and the same is true for words.
- Words are delimited by wide white space either with in a line or at the end of lines.
- There is no capital, lower case distinction as it is for Latin characters.

- A line of Amharic printed script lies at the same level, having no ascent and descent features. This feature of Amharic writings system grants that there will always be a white line between two consecutive lines of characters, unless and otherwise the line is inclined.

3.2.4. Ethiopic Character Standardization

Even though Ethiopic script has served Ethiopians for more than a millennium, it is being marginalized from the digital environment due to some standardization issues that have not been adequately handled. As a result, all the users of the script are also being marginalized.

In the absence of a standard Ethiopic character encoding system, Ethiopian software companies have been developing Ethiopic based software products that are incompatible. Due to the lack of a keyboard standard, every Ethiopic based software developer comes with its own keyboard layout.

A number of efforts are being made to design Amharic computer fonts by various business establishments and most of the work in this area has concentrated on the design of characters whose shapes are similar to the one used in the printing presses (as the normal type style), and on minimizing the number of key strokes to punch a character.

Even though software products that use Ethiopic script have been developed as early as in the beginning of the 1980, Ethiopic standardization issues have started to be raised by the Ethiopian IT professional only starting from the early 1990s (Dawit, 2003). The first initiative that deals

with Ethiopic standardization for the use of computing was probably Joseph Becker's early 1990s proposal to include Ethiopic in the newly born Unicode. Several people including Lloyd Anderson, Daniel Yacob, Yitna Firdyiwek, Yonas Fisseha and others contributed to the Unicode proposal from 1991 to 1996. Thanks to the efforts of these people, Ethiopic has been included in Unicode starting from version 3.0. The inclusion of Ethiopic in Unicode is a major milestone. In addition to the above initiatives, in 1997, alerted by the fact that some characters have been left out in the Ethiopic that is to be adopted by Unicode, the Ethiopian Science and Technology Commission (ESTC) and Ethiopian Computer Standards Association (ECoSA) prepared a proposal for extension of the Ethiopic characters included in Unicode. However, these efforts did not bear fruit due to delays that did not enable submission of the proposal on time (Dawit, 2003).

Other Ethiopic standardization initiatives existed parallel to Unicode's efforts. A committee established by Ethiopian E-mail Distribution Network (EEDN) conducted a survey of Ethiopic software products in March 1993 and concluded that there is a need of an Ethiopic encoding standard. Abass (1994) stressed the need of not only an encoding standard but also of standards for keyboard layout as well as transliteration standards. On the same year, impatient of waiting for Unicode, the CEC (Committee for Ethiopic Computing) composed mainly of Ethiopians in North America developed two Ethiopic encoding standards. EthCITA (Ethiopian Computing and Information Technology Associations, Inc), who formed the CEC, adopted these standards as its official web publishing system, in 1995 (Dawit, 2003).

Almost all the above initiatives were made by Ethiopians and other scholars living in North America and in Europe. It is only starting from 1997 that Ethiopic standardization has obtained

some interest in Ethiopia. Despite the increasing importance given by the Government (Ethiopian Science and Technology Commission) and some private Companies (Dashen for example) to computing in general and Ethiopic computing in particular starting from the mid 1980s, no standardization effort existed in Ethiopia until the First International Conference on Information Technology and Computational Ethiopic was held in 1997. This conference recommended the establishment of an association with the objective of developing and disseminating computer related Standards in Ethiopia. Following this recommendation the Ethiopian Computer Standards Association was established in 1997 and became a legal entity in 1999 (Dawit, 2003).

In 2000, ECoSA submitted a proposal to the Quality and Standards Authority of Ethiopia (QSAE), for four Ethiopic computing related standards (Dawit, 2003). The proposal concerned:

- Ethiopic character set
- Ethiopic encoding system
- Keyboard layout
- Ethiopic to Latin transliteration

The standardization work followed and in October 2002, the first standard (Ethiopic Character Set) has been officially adopted by the QSAE as an Ethiopian national standard.

Currently, the standardized representation of Ethiopic characters is used to facilitate the researches undergoing in the area of character recognition.

3.3. PRINTED AMHARIC DOCUMENTS

3.3.1. Beginning of Literatures on Amharic Characters

For many centuries the Ethiopian literary tradition was based on Geez, the sacred language of the Ethiopian Orthodox Tewahedo Church, which was the exclusive presence of the ecclesiastics. Being a "secular" language, Amharic was normally used as a medium of spoken communication. Due to this dichotomy the role of Amharic as a written language was marginalized, but it acquired authority as the language spoken inside the royal court, as the medium of preaching and teaching in the ecclesiastical milieu and as the lingua franca of the large part of the Ethiopian highlands. Thus, it is natural that Amharic was taken for those needs and purposes for which a written language had to be used, a widely spoken and understood one, rather than the extinct language (Geez) (Verlag, 2003).

Amharic written composition for the period before the 17th century is rare and present valuable material for the historical study of the older stage of the Amharic language. According to Verlag, (2003), the development of Amharic written tradition was encouraged especially by several historical events and intercultural encounters. In the 16th century the Portuguese are reported to have written some texts in Amharic in order to promote the Catholic faith. A number of religious works were translated from Geez to Amharic to make them accessible to the common ecclesiastics and use in the public anti-catholic polemics and to promote the orthodox faith. Correspondence in Amharic appeared as the local Ethiopian rulers started regularly to contact foreigners.

3.3.2. Size of Printed Amharic Documents

Even though modern Amharic literary activity is not much more than a century and a half old, its beginning was preceded by the existence of an Amharic written tradition, a large heritage of which was long neglected and has been just recently begun to be studied.

The introduction of the printing press around 1900 encouraged the proliferation of books. Books and newspapers were printed using type-setting technology. Type-setting provided a fairly faithful reproduction of the Ethiopic characters, which prior to the introduction of the printing press, were written by hand. Around 1920 Ayana Birru introduced the Amharic typewriter, which consisted of a modification of the print head of the familiar English typewriter. While the Amharic typewriter allowed for a widespread production of printed documents, it was not without drawbacks. In particular, using the Amharic typewriter, glyphs were produced by a complicated series of the partial glyphs successively typed over the top of each other. The glyphs thus produced were merely an approximation of the true hand-written Ethiopic characters, which were more faithfully reproduced by a printing press. Thus, despite its utility, the Amharic typewriter, led to the proliferation of sub-standard characters and incomplete Amharic sets (Dawit, 2003).

With the advent of computer technology, it became possible to efficiently produce printed documents having Ethiopic characters without the use of a printing press. Early attempts at adopting computer technology for this purpose, did not use the true Ethiopic characters, but instead used glyphs similar to those produced by the Amharic typewriter. Accordingly, the full range and variety of Ethiopic characters were not expressed with improvements to computer

technology, it became possible to more faithfully render the true Ethiopic characters (Verlag, 2003).

To exemplify the collection of books in Ethiopia, it is worth mention the amount of collections in the National Library of Ethiopia. It was first established in 1944 with the name "Public Library Wemezekir". Out of the most invaluable documents which the National Library has collected in the last sixty years through purchase, donation and Proclamations, the followings are some:

- Ancient and historical manuscripts written in the fourteenth and fifteenth centuries as; The Four Gospels, Saint Paul's Message, and Passion Week;
- Books printed since 1520 as Psalm of David, Tobia, State and Public Governance, Wodajie Libie, and World History;
- Books and chronicles focusing on ancient, middle and modern Ethiopian political, Economical, Social and Cultural affairs written by Ethiopian and foreign historians;
- Governmental and non-governmental newspapers published since 1902 as: AEMERO, BERHANENA SELAM, DIMTS ERITREA, YEQESAR MENGIST, DIMTS, ADDIS ZEMEN, THE ETHIOPIAN HERALD ETC.
- Governmental and non-governmental magazines published since 1941 as:- BERHANENA SELAM, MENEN, YEKATIT, ETC.
- Videotapes, Magnetic tapes, Historical Photographs and several other prints abundantly available for study and research purposes.

According to the official web site of the National Library of Ethiopia (NLE), http://www.nale.gov.et/index.html, accessed on March, 03, 2010, the number of collection of books is indicated below:

Types of Collections	**Size of Collections**
Books	58,854
Periodicals & Doc.	33,627
Manuscript	859
Microfilms	9239
Bind News Paper	2650
CD, Audio, Video	4408
Total Collection of Library	109,990

Table 3.1. Number of collections of books on Ethiopian National Library (adopted from official website of National Library of Ethiopia, http://www.nale.gov.et/index.html, retrieved on March 03,2010).

3.4. CHALLENGES IN BUILDING AN OCR FOR AMHARIC DOCUMENTS

Character recognition from document images that are printed in Ethiopic script is a challenging task. To develop a successful character recognition system for Amharic script, the following issues need to be addressed.

3.4.1. Degradation of Documents

Document images from printed documents, such as books, magazines and newspapers are extremely poor in quality. Popular artifacts in printed document images include (Million and Jawahar, 2005):

(a) Excessive dusty noise,

(b) Large ink-blobs joining disjoint characters or components,

(c) Vertical cuts due to folding of the paper,

(d) Cuts at arbitrary direction due to paper quality or foreign material,

(e) Degradation of printed text due to the poor quality of paper and ink,

(f) Floating ink from facing pages etc.

This is the main issue where most character recognition research even for Latin and non-Latin scripts also fails. Careful design on an appropriate representational scheme and classification method is needed so as to accommodate the effect of degradation (Million and Jawahar, 2005).

3.4.2. Printing Variations

Amharic printed documents vary in fonts, sizes and styles. Building character recognition system is challenging in this situation. For example, some of the commonly used fonts in Amharic printed documents include `PowerGeez', `VisualGeez', `Alphas', and `Agafari'. Each of these fonts offers several stylistic variants, such as normal, bold, italic, etc. They are also written in different point sizes, including 10, 12, 14, etc. These fonts, styles and sizes produce texts that greatly vary in their appearances within printed documents. Standardizing the variation in size by applying normalization techniques and extract suitable features is a challenging task so that the representation is invariant to printing variations.

3.4.3. Large number of characters in the script

The total number of characters in Amharic script is more than 300. Existence of such a large number of Amharic characters in the writing system is a great challenge in the development of Amharic character recognizer. Memory and computational requirements are very intensive (Million and Jawahar, 2005). One needs to design a mechanism to compress the dimension of character representation so as to come up with computationally efficient recognizers.

3.4.4. Visual similarity of most characters in the script

There are a number of very similar characters in Amharic script that are sometimes difficult for humans to distinguish them easily (examples are presented in Figure 3.1). Robust discriminant features needs to be extracted for classification of each of the character into their proper category or class.

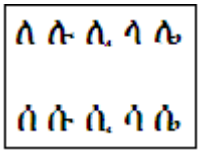


Figure 3.1. Samples of visually similar characters in Amharic writing system.

3.4.5. Language related issues

The Amharic language poses many additional challenges. Some of these are (Million and Jawahar, 2005):

- lack of support from browsers and keyboard, and
- lack of language processing routines.

These issues add to the complexity of the design and implementation of an optical character recognition system. In this thesis work, on the course of developing OCR engine for real-life documents, emphasis was given in developing robust noise removal module for repairing degraded images.

3.5. THE AMHARIC OCR SYSTEM

Amharic is the official and working language of Ethiopia and the most commonly learnt language next to English throughout the country. Amharic is written in the unique and ancient Ethiopic script (inherited from Geez, a Semitic language), now effectively a syllabary requiring over 300 glyph shapes for representation.

Character recognition from document images that are printed in Amharic characters is a challenging task due to the reasons mentioned in section 3.4. However, various studies are being

conducted since recent times. The studies, in general , aimed at the recognition of machine printed, typewritten and handwritten Amharic documents and, so far, the recognition of computer printouts on real-life documents such as books, magazines and newspapers cases are not considered. Therefore, there is a need to exert much effort to come up with better and workable OCR for Amharic characters on real-life documents. Hence, it is worth to mention some of the related works reported at home and in the diaspora.

3.5.1. Printed Amharic Character Recognition

As the first major attempt, Worku (1997) built an Amharic OCR by applying experience elsewhere to investigate the extent to which suggested OCR algorithms to work with other scripts would apply to recognizing Amharic characters. Worku did not apply binarization, noise removal, skew detection and normalization techniques due to time limitation. Once pages are preprocessed, then they are segmented into individual components. Worku adopted the stage by stage algorithm suggested by Pal and Chaudhuri (1995) for the reason that the nature of the Amharic scripts permitted the application of the segmentation algorithm to operate very well. In the stage by stage segmentation algorithm, a text is segmented in three successive steps: line, word and character segmentation. Concerning recognition algorithm, Worku adopted a recognition algorithm that uses a tree classification scheme built by using topological features of a character.

Concerning recognition algorithm, Worku adopted a recognition algorithm that uses a tree classification scheme built by using topological features of a character.

To determine the accuracy and speed of an OCR algorithm, Worku conducted testing from the main test case and additional test cases. The main test case includes all words which were selected for statistical computation was encoded as a plain text using the WashRa front as size of 12 points and with normal typestyle. Additional test case include printout WashRa bold typestyle, printout WashRa italic typestyle, original text from the "Addis Zemen" newspaper, a text from a fiction entitled "□□ □□□" and Nebar font with the normal typestyle. The test results are presented in Table 3.2 below.

Document	**Test Data (in characters)**	**Accuracy (%)**
WashRa at size of 12 points with normal font style	20,259	97.14
WashRa printout bold typestyle with Bold font style	1064	98.87
Text from "Addis Zemen"	860	28.14
Text from Amharic fiction	1194	75.71
Text from Nebar font of NCI	1995	75.34

Table 3.2. Test results of the recognition engine by Worku (1997)

The test results from italic text are not included because the result is found almost 0 % (Worku, 1997). From the results, Worku reached two conclusions. One is that, the line segmentation works properly even if the text is not on a white paper. The second is, the result of character segmentation was found very poor in case of italic typestyle.

3.5.2. Formatted Amharic Text Recognition

Worku's algorithm was not capable of recognizing texts written in different font sizes and styles (such as italics and underline). Ermias's (1998) work, tried to do further by introducing some preprocessing techniques so that his algorithm recognizes texts written in different sizes and styles.

For the purpose of thinning, Ermias considered two algorithms. These were the algorithms suggested by Zang and Suen (1984), and the one suggested by Hilditch (1969). According to the test made on the basis of the forgoing considerations, Ermias selected the algorithm suggested by Zang and Suen (1984). To underline detection and removal Ermias considered two methods for implementation. The first method was directly derived from the line segmentation algorithm adopted by Worku. The other method was the one used by Pal and Chaudhury (1995) to remove the matra line which is found in most of the Bangla characters. The second method was applied in Ermias's work to remove underlines. After applying the selected underline detection and removal and thinning algorithms, Ermias reported a recognition rate of 89.25%.

3.5.3. Typewritten Amharic Text Recognition

In reality, most Amharic typewritten documents that need to be converted into machine-readable format are typewritten and printed on non-white paper. In Dereje's (1999) work an attempt was made to explore the possibilities of developing an OCR system for typewritten Amharic text.

On the course of segmentation, Dereje observed that the Amharic typewritten text has many connected characters. This leads to misclassification of the characters by the recognition

algorithm (Dereje, 1999). To accommodate these characteristics of typewritten Amharic documents Dereje modified the stage by stage segmentation algorithm. Concerning the image restoration, Dereje used Binary morphological filtering algorithm, first suggested by Liang *et. al.*, (1996) for the removal of subtractive and additive noise in degraded images, since it is found to work better.

To test the selected algorithm, Dereje took 5 pages containing 5,172 characters, out of the 19 pages Worku generated to be written in the typewriter for testing. The modified recognition algorithm is tested on five pages of typewritten Amharic text with isolated characters and 61% accuracy is registered. When the recognition algorithm is tested on documents without intentionally isolating the characters, the accuracy of the recognition algorithms dropped significantly 48%. Dereje's modified stage by stage segmentation algorithm is tested along with the recognition and a better performance is achieved. This segmentation algorithm improved the OCR system by 5%.

3.5.4. Artificial Neural Network based Amharic Character Recognition

The use of ANN for the recognition of Amharic texts was a study as a thesis work by Birhanu (1999). This is, in fact, a different approach than those discussed so far.

Regarding the development of the segmentation algorithm, Birhanu implemented the segmentation part into two steps: line segmentation and character segmentation. For line segmentation, the algorithm adopted by Worku and originally suggested by Pal and Chaudhuri (1995) is used. Birhanu used MATLAB image processing tool box, which extracts the character image in its bounding box, for character segmentation. On the other hand , since character

segmentation using image spatial histogram (as suggested by Pal and Chaudhuri (1995)) was reported to be typically only about 60% accurate (especially for connected characters), Birhanu ignored this algorithm from further consideration.

The approach used by Birhanu for recognition of printed Amharic characters using neural network in such a way that after characters are segmented using suitable segmentation algorithm they are fed to a feed forward neural network. A feed forward neural network with 256 inputs, 40 hidden and 8 output nodes was designed and trained using the back propagation algorithm on selected Amharic characters as training sets. The neural network was tested on printed Amharic document, which is written using ALXethiopian font with 16 point size and a recognition accuracy of 35% is observed.

3.5.5. A Generalized Approach to Amharic OCR

The main aim of Million's (2000) work is to enable the Amharic OCR system recognize characters written in different fonts to generalize the previously adopted recognition algorithm insensitive to the different font types. Thus using different test cases, first, the efficiency of algorithms were tested individually and, then, both thinning, segmentation and feature extraction/detection algorithms are integrated together and tested on actual cases to evaluate the performance of the system on Amharic documents written using Agafari, WashRa and Visual Geez fonts.

Million tested his recognition algorithm using different test cases written in WashRa, Agafari_Addis Zemen and Rejim and Visual Geez font types. The test results are 49.38% for WashRa, 26.04% for Agafari-Addis Zemen and 27.64% for Rejim and 15.75% for Visual Geez.

3.5.6. Versatile Character Recognition System for Amharic Text

Yaregal (2002) attempted to recognize Amharic text written with different font sizes. To this end, the segmentation algorithm implemented in previous researches is tested and a remarkable result is found. Thus, he used the same algorithm in his study to implement character segmentation.

Yaregal used a hybrid system of syntactic/ structural and neural network approaches. The developed system has two parts: pattern extraction and recognition. The pattern extraction consists of primitive extraction, primitive tree building and pattern generation.

Yaregal's classification/recognition of the developed OCR system used neural networks. The network has 64 input nodes, 64 hidden nodes (of 1 layer), and 8 output nodes (Yaregal, 2002). Yaregal used a supervised learning with a back propagation algorithm to train and select the best network model.

The developed system is trained with Amharic characters written with VG 2000 Agazian font sizes of 10 and 12 and tested with sample documents of font sizes 8, 12, and 14. For the 8 font size, the developed system correctly recognized 62.02% of the characters included in the training set. It also correctly classified 62.87% of the characters not included in the training set as unknown. For the 12 font size, the system correctly recognized 74.68% of the characters included in the training set were also correctly classified as unknowns. The results for font size 14 was also not far from the above two cases. The system correctly recognized 73.18% of the

characters included in the training set. For characters not included in the training set 70.04% was correctly classified as unknowns.

These researchers tried to solve some problems of the Amharic OCR development. The common ball for all these works is that all based their investigation on printed, either by typewriter or printer, Amharic texts prepared systematically. Most of these documents are free from noises and hence the researchers do not consider the integration of noise detection and removal algorithms in the development of Amharic OCR. Dereje (1999) on the other hand attempts to use binary morphological filtering algorithm for the removal of subtractive and additive noises in typewritten document images. However, when we closely investigate real life documents there are various types of noises we encounter. Such noises cannot be detected and removed by applying only morphological filtering algorithms. Thus, in this thesis, the OCR technology with particular emphasis on noise detection and removal is investigated for Amharic texts taken from the real-life documents.

CHAPTER FOUR

EXPERIMENTATION

4.1. INTRODUCTION

This chapter presents the results of the experiment carried out to develop a recognition system for printed Amharic characters for real-life document. The preprocessing, segmentation, and the neural network recognition techniques discussed in the previous chapter were implemented and tested on a set of printed Amharic real-life documents. The steps used to collect the data, preprocessing and model building are presented hereunder.

4.2. DATA COMPILATION

In research like character recognition, the primary task is collecting required data and preparing it for further processing. Accordingly, gathering of text documents written in different styles, fonts, and sizes has been carried out. On the course of collecting text written in different styles, fonts, and sizes, difficulty was faced in choosing the representative real-life documents.

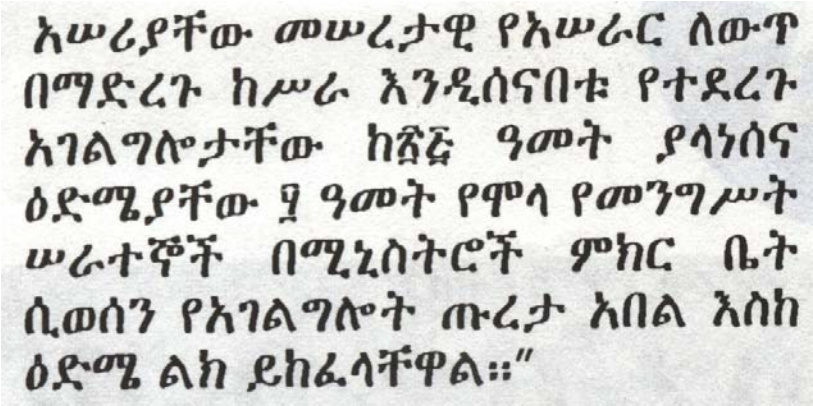
Since the research is dealing with pattern recognition, or more specifically characters recognition, the data collected are of two types. The first being data for training the neural network (recognition engine), the second will be for testing the performance of the recognizer. Towards this end, the approach followed in the data collection considered to have 275 (231 core and 44 labialized) printed Amharic characters for the purpose of training due to time limitation.

Although, there are a number of sources to collect data from, such as Amharic books (fictions, sciences, arts, textbooks, etc); newspapers (private and government); and Amharic dictionaries, etc), it was considered reasonable to select representative real-life documents from different sources to increase the generalization ability of the ANN. Since the population size is unknown, sample size determination was a problem.

The technique used for sampling was purposive sampling. By taking the limitations of time and cost into consideration, it was decided to take a sample of four sources for the problem domain. These are texts from the popular government Amharic newspaper 'Addis Zemen', texts from the Bible, texts from the 'Federal Negarit Gazeta', and the fiction 'Fiker Eskemekabir'. These samples were chosen for the following reasons:

- The diversity of their content (politics, religious, arts, sport, science, culture, advertisements, etc.).
- Words of different authors in these samples are expected to show the exact frequency distribution of characters.
- Amharic characters in these sources are with a variety of writing styles, sizes and fonts which is important for the identified problem domain to be dealt.
- With the assumption of the real-life representatives of Amharic printed documents, they are believed to have real-life features like noises which in turn is appropriate for the problem domain to be addressed.
- They were accessible.

As a result, considering the time limitation, two page documents from each source is considered for training. For testing the recognition engine developed, a one page document is prepared from each source. Sample character set from ‘Federal Negarit Gazeta’ for the training purpose is given in Figure 4.1 below. Refer to Annex II and III, for the character sets used for training and testing.


አሠሪያቸው መሠረታዊ የአሠራር ለውጥ
በማድረጉ ከሥራ እንዲሰናበቱ የተደረጉ
አገልግሎታቸው ከሰፔ ዓመት ያላነሰና
ዕድሜያቸው ፱ ዓመት የሞላ የመንግሥት
ሠራተኞች በሚኒስትሮች ምክር ቤት
ሲወሰን የአገልግሎት ጡረታ አበል እስከ
ዕድሜ ልክ ይከፈላቸዋል።"

Figur 4.1. Sample Amharic text taken from “Federal Negarit Gazeta”

4.3. PREPROCESSING TASKS

As described in section 2.3, preprocessing is necessary for efficient recovery of the text information from scanned image. The processing algorithms employed on the scanned image depend on paper quality, resolution of the scanned image, the amount of skew in the image, the format and layout of the images and text and so on. In this work, the researcher applies digitization, noise removal, binarization, thinning, and normalization techniques.

4.3.1. Digitization

As described in section 2.3.1, in OCR systems, the first electronic task that follows after data collection and before any preprocessing task is digitizing the printed or handwritten document.

For the problem at hand, printed real-life documents collected from ‘Addis Zemen’, Bible, ‘Federal Negarit Gazeta’, and ‘Fiker Eskemekabir’ are scanned and prepared for further manipulation.

As digitization is the process of converting a printed or handwritten document to an electronic format, the printed documents with varying styles, fonts, and sizes from real-life samples are scanned. The scanning device used is “BENQ Scanner” which is flat bed scanner with “BENQ Scanner Driver”. The scanning dpi (density per inch for vertical as well as horizontal dimensions) parameter for the sample documents was 300 dpi and with gray scale color format as recommended by Mori and Yamamoto, (1992). 300 dpi is found efficient for real-life documents because it does not tend to break thin lines or fill gaps, thus validating features.

The scanned grayscale color images are then put as an input in MATLAB working folder. These images then are processed using different preprocessing algorithms. The following sections describe the experiments done on the digitized documents inside the MATLAB programming environment.

4.3.2. Noise Detection and Removal

Sample scanned real-life documents collected from different representative sources contain noise that mainly arises due to printer, scanner, paper quality and age of the document. Application of binarization on these noisy scanned real-life documents to convert gray scale image (with 256 possible different shades of gray from black to white pixels) to binary colors (with just black and

white pixels) to ease image processing. In due course, binarization removes some of the noises in images. However, because of the level of degradations in digitized document images of magazines, books, and newspapers, there is a need to apply noise filters so as to reduce the effect of degradation during the recognition process.

As described in section 2.3.2, literatures reveal that there are three types of noise removal methods (Ntogas and Ventzas, 2008). These are Linear filtering, Median filtering and Adaptive Filtering. The MATLAB Image Processing Toolbox provides a number of different ways to remove or reduce noise in an image. Different methods are better for different kinds of noise. For the problem domain in this work, i.e., printed real-life Amharic images with different kinds of noise, these noise removal methods are tested. Below are details of the implementation of each filtering methods.

A. Linear Filtering

Linear filtering is used to remove certain types of noise. Mean filter is useful for removing Gaussian noise from a scanned document for historical manuscripts with noises as used by Ntogas and Ventzas (2008). Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by Gaussian are reduced. Gaussian noise is characterized by each image pixel having value from a zero mean Gaussian distribution.

Through experimentation, this method is ignored from further processing because it blurred edges and remove details while filtering Gaussian noises which are found in real-life documents. See Annex IV (a) for detail implementation.

B. Median Filtering

The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove Salt and pepper noises without reducing the sharpness of the image. Salt and pepper noises consist of random pixels' being set to black or white (the extremes of the data range) (MathWorks, 2010). Although median filter is a useful image filtering technique, it also has some disadvantages while applying on real-life Amharic documents. The median filter removes both the noise and the fine detail since it can not tell the difference between the two. Anything relatively small in size compared to the size of the neighborhood will have minimal effect on the value of the median, and will be filtered out. In other words, the median filter can not distinguish fine detail from noise (Vernon, 1991). See Annex- IV (b) for detail implementation.

C. Adaptive Filtering

Wiener filter applies to an image adaptively, tailoring itself to the local image variance. Where the variance is large, wiener filter performs little smoothing. Where the variance is small, wiener filter performs more smoothing. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image.

In addition, in the adaptive filter used in this thesis work, *wiener2* MATLAB function, there are no design tasks i.e., handles all preliminary computations and implements the filter for an input image. *wiener2*, however, does require more computation time than linear filtering (Math Works, 2010).

The code below applies *wiener2* to the sample image given above in the problem domain.

```
Nonoiseimg = wiener2 (Grayimg); % applies wiener2 filtering
figure, imshow (Nonoiseimg); % Show binary image without noise
title ('INPUT IMAGE WITHOUT NOISE')
```

The above image (Figure 4.1) is passed through adaptive filtering module and produced the adaptive filtered image. The detail is documented in Annex-IV (C). Hence, adaptive filtering is tested and works well through experimentation as it is more selective than a comparable linear filter and median filtering, preserving edges and other high-frequency parts of characters. Hence, it is further considered for this work. Accordingly, all the binarized documents are filtered using adaptive filtering method.

4.3.3. Binarization

Binarization, which is the task of discriminating the digitized document pixels into foreground and background image, is performed immediately after the noise is removed. This is because, Cheoi (2007) explained that if we binarize using only just global binarization algorithm, even if the execution time would be fast, the quality of the binarized image could be bad. However, by applying filtering module that enhances images' quality for better binarization; we can overcome the shortages of the global binarization algorithm.

Research shows that the binarization methods available could be either global or adaptive. For the case of this problem domain methods from both approaches were tested.

For adaptive thresholding, one of the toolbox in the MATLAB environment, named Two-Dimensional Variance Adaptive Thresholding of Wavelet Coefficients was tested. This

capability is available through graphical interface tools throughout the MATLAB Wavelet Toolbox. This tool allows defining, level by level, thresholds, and then increasing the capability of the de-noising strategies handling nonstationary variance noise. The noise variance can vary with time. There are several different variance values on several time intervals. The values as well as the intervals are unknown. This can be supported by one of the graphical interface tool (SWT De-noising 2-D) to illustrate this capability. This toolbox found to be inefficient for the problem at hand because it affects the basic features of the characters while trying to isolate the character pixels from background pixels. In addition, it is complex in adapting threshold values for further processing. A sample of binarized image using local thresholding algorithm is depicted in Appendix V.

Global thresholding on the other hand select one threshold value to work for all pixels in the image. MATLAB Image Processing Toolbox creates a binary version of the images loaded in the working directory by using thresholding. To binarize, the MATLAB Image Processing Toolbox uses global image threshold using Otsu's method.

In a binary image, each pixel assumes one of only two discrete values. Essentially, these two values correspond to *on* or *off*. A binary image in the MATLAB Image Processing Toolbox is stored as a logical array of *0's* (off pixels) and *1's* (on pixels).

The code that creates the binary version of the above sample image is:

```
threshold = graythresh (Nonoiseimg);
Binaryimg = im2bw (Nonoiseimg,threshold);
Figure, imshow (Binaryimg)
title ('INPUT BINARY IMAGE WITH OUT NOISE')
```

Accordingly, the function *graythresh* automatically computes an appropriate threshold to use to convert the intensity image to binary. *threshold* is a normalized intensity value that lies in the range [0, 1]. The *im2bw* function performs the conversion from intensity image to binary. *im2bw* produces binary images from indexed, intensity, or RGB images. To do this, it converts the input image to grayscale format (if it is not already an intensity image), and then uses thresholding in this case *graythresh* to convert this grayscale image to binary. The output binary image BW has values of 1 (black) for all pixels in the input image with luminance less than *level* and 0 (white) for all other pixels. Sample binarized image using Otsu's global thresholding method is shown in Figure 4.2 below.

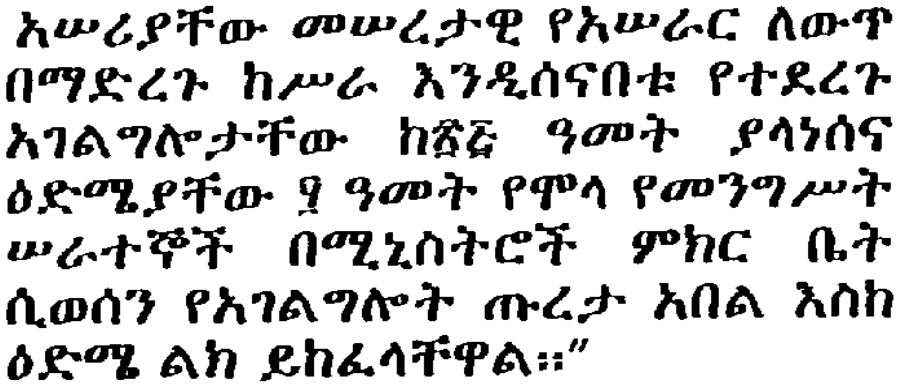


Figure 4.2. Sample Binarized Image

This method is found to be more effective in isolating the text pixels from background pixels from the image without affecting the basic features of the character. Therefore, it is considered for further tasks.

4.3.4. Segmentation

In OCR systems, image parts of interest to the recognizer, which could be sentences, words, characters or any type of strokes, should be isolated. For the character recognition problem domain, the point of interest or the low level abstraction to be extracted is the character. Thus, this part is devoted to describe the methods and approaches implemented to deal with character segmentation problem.

A. Line Segmentation

The segmentation algorithm suggested by Pal and Chaudhuri (1995) first constructs a histogram for the sum of values of pixels in each row and based on a selected threshold extracts a text line. Each text line is subjected to word segmentation, which in turn scans vertical lines and counts the sequence of lines whose gray level sum is less than a threshold, then decides to segment a word based on this count. Characters are also segmented in a similar manner as words are segmented.

As described in chapter three, in Amharic writing system, a line of Amharic script lies at the same level, having no ascent and descent features. This feature of the Amharic writing system makes the stage-by-stage segmentation algorithm to fit remarkably. In addition to this feature, printed Amharic real-life documents is produced with standard printing machines in which having no ascent and descent of characters that helps all characters in one line to fall on the supposed area.

The stage by stage segmentation algorithm suggested by Pal and Chaudhuri (1995) was used in this thesis work. A white row will have a maximum sum of pixels brightness values, while a row in which a text exists will have a smaller sum. Therefore, a set of sequential rows of pixels having maximum sum is considered as a line of text. A similar approach was successfully implemented by (Worku, 1997; Ermias, 1998; Dereje, 1999; Million, 2000; Nigussie, 2000).

For the experimentation as well as test text pages, no line segmentation error is observed when trying to isolate each logical text line from the documents. Thus, the performance of the line segmenter is found to be 100%. To illustrate the performance of the stage-by-stage line segmentation algorithm module as shown in Appendix VI, sample output of the procedure is depicted in Figure 4.3 below.

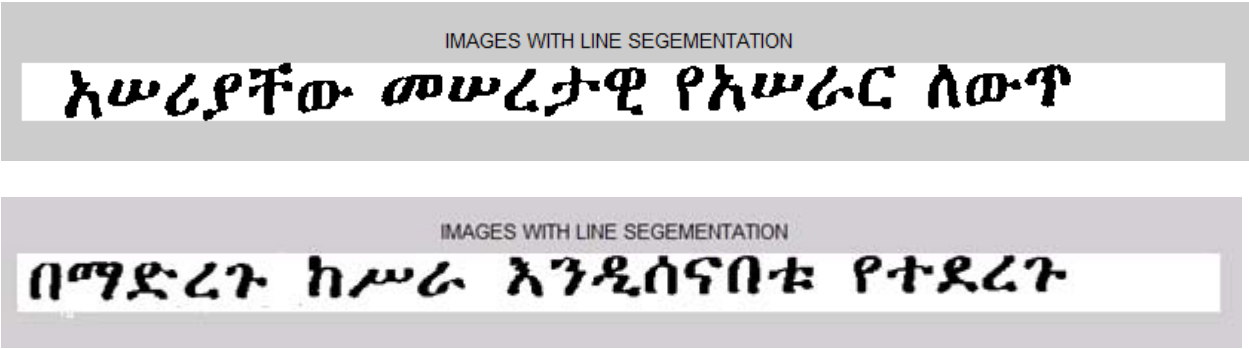


Figure 4.3. Sample Outputs of the Line Segmentation

B. Character Segmentation

The bounding box projection approach for character segmentation under stage by stage algorithm has advantages over the pixel projection approach. It is computationally less intensive (Liang *et. al.*, 1997). Therefore, this approach to character segmentation was implemented using MATLAB image processing toolbox, which extracts the character image in its bounding box. Each

character is considered as a connected object in the image, which was segmented as a line of text.

The bounding box projection approach works based on the assumption that each character is treated as a connected object. One of the features of the Amharic writing system is symbols are written in a disconnected manner. Due to this feature characters are segmented as a connected object while words are not because each character is disconnected. Hence, modification is needed on this approach in order to apply it for word segmentation.

After the lines of text found in the image are identified, the whole raster image is scanned to segment each character. For this purpose, as the stage-by-stage segmentation algorithm proceeds, each logical line is identified and the character images are segmented based on the assumption that each character can be treated as a connected object. Then each segmented characters are put into bounding box.

While extracting each character from the identified text line in its bounding box, an array is created and the features of each character in the bounding box are stored. This array is used to trace each isolated character for the coming steps, which will be imposed on the segmented character image areas.

After character segmentation module as shown in Annex-VII is executed, Figure 4.4 has the following pattern.

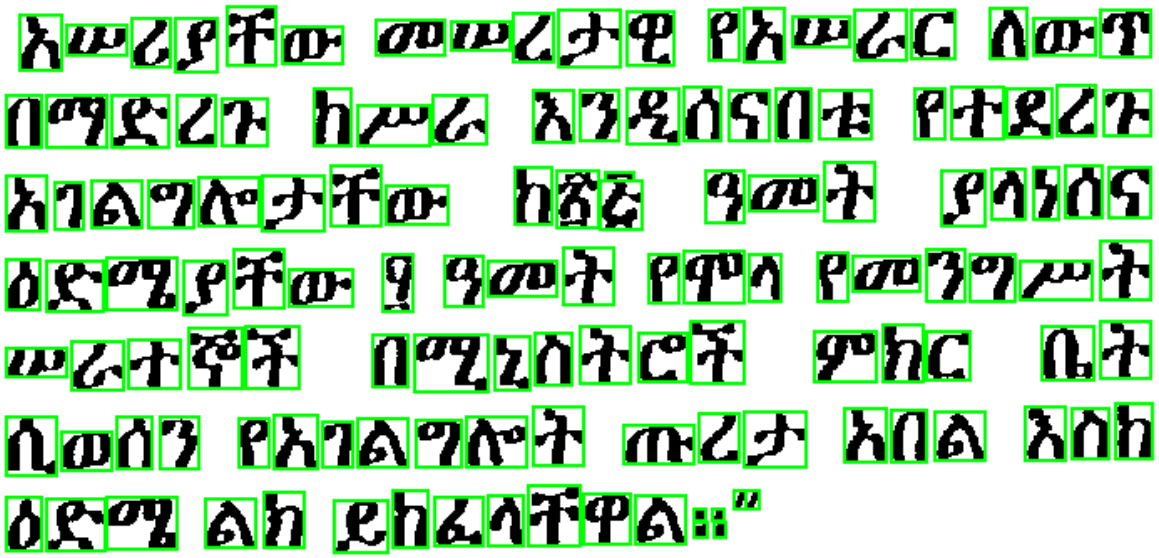


Figure 4.4. Sample segmented characters in the bounding box

On the course of isolating each character from the text line, it is observed that the assumption that each character can be treated as a connected object in the MATLAB labeling function holds true for the basic 231 characters. However, from experiment, it is observed that this approach did not work for Amharic characters constructed by disconnected strokes for a single character (special character (ቨ), punctuations, and numbers) on the effort to develop a complete OCR for the whole Amharic characters. Figure 4.5 shows the problem where disconnected strokes for single Amharic characters are segmented as a different character.

(a)

(b)

Figure 4.5. Sample segmentation errors (a) the number “  ” is taken as two distinct characters though it is not. (b) the four dots of “Arat Netib” (⁖) and to strokes of quotation marks (″) are recognized as distinct characters.

For both training and test cases, a total of 1898 characters (1262 from training and 636 from testing sets) printed characters from the four sample sources were segmented by this stage-by-stage character segmentation module. From these sets of characters, Table 4.1 reports correctly and wrongly segmented characters for the items on the Amharic Fidel set.

Items in the Document	Characters in the Documents		Wrongly Segmented Characters		Correctly Segmented Characters			
	Training	Testing	Training	Testing	Training	Accuracy%	Testing	Accuracy%
Basic Characters	1106	551	19	8	1087	98.28	543	98.55
Special Characters	1	0	1	0	-	0	0	-
Labialized Characters	3	5	-	0	3	100	5	100
Punctuations	149	80	27	46	122	81.88	34	42.5
Numbers	3	0	1	0	2	66.67	0	-
Total	1262	636	48	54	1214	96.20	582	69.62

Table 4.1. Number and accuracy of correctly and wrongly segmented characters.

Wrongly segmented characters are found to be segmented as distinct characters though they are not. The rate of segmentation for basic and labialized characters, which will be considered for

the coming recognition process, turns out to be 98.28% and 100% respectively for training character sets, 98.55% and 100% respectively for testing character sets, which is remarkable for real-life printed text processing.

The above problem found to propagate the inconvenience to the subsequent algorithms commanding a negative impact on the overall recognition. The binary representation produced for special characters, numbers and punctuations types of problems is traced down and removed from the binary file holding the representation of each identified character. After character segmentation part is over, the immediate procedure used is the size normalization that transforms each segmented character to some uniform height and width.

4.3.5. Normalization

In this part of the experimentation, all the characters segmented during the segmentation step are fed to the module that performs size normalization. Since the neural network, the selected recognition approach, accepts same size of input vector for the training as well as test data sets, the character images need to be normalized to some standard size.

Even though there exist a good many of size normalization algorithms like the Run Length Coded approach described and reported to perform well in Pandya and Macy (1995), the inbuilt function of the MATLAB environment is used to do the size scaling. The MATLAB built in function used to do the size scaling task uses pixel brightness interpolation method. This approach not only lowers the burden of the research but also overcome major problems like round off errors in using the RLC (Run Length Code) scaling algorithm. As discussed in chapter 2, interpolation works by using known data to estimate values at unknown points.

The amount of normalization is application specific. For the problem domain at hand, 16×16 and 20× 20 is tested. Segmented images are fed to the image normalization module. The result corresponding to the image at hand, with 16×16 and 20× 20 size normalization is shown below.

a b

Figure 4.6. Sample Normalized Characters by (a) 16×16 and (b) 20× 20.

As can be seen from Fig 4.6, character "ؤ" with 16×16 losses its connectivity due to normalization process. The same character has obtained its proper connectivity when its amount of normalization becomes 20× 20. On the other hand, disconnected character strokes for a single character are also normalized by the size normalization module taken as two distinct characters. As shown Figure 4.7 below, the number "  " is normalized as two different characters caused from the segmentation module.

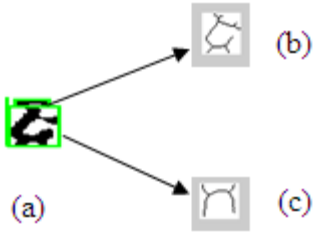


Figure 4.7. Erroneous Character Segmentation (a) Original character (b) a bottom stroke and middle character normalized as a single character. (C) top detached stroke is normalized as a distinct character.

The above demonstration reveals that the problem caused during character segmentation propagates to the size normalization phase and take component of one character as a unique one and proceed forward to normalization.

20×20 size representation is considered in this thesis work for further processing because in the context of printed character recognition problem at hand, through experimentation it is found that a 20×20 representation is sufficient to preserve the shape of the printed input page. The final result of size normalization process is a set of character components with their bounding box that are scaled to a size of 20 × 20. This data set is passed to thinning module in the next section.

4.3.6. Thinning

Character thinning is used as a preprocessing stage in OCR to simplify the subsequent recognition problem. However, specific defects in the data can cause thinning algorithms to destroy information and lead to misrecognition. The number and importance of such defects are particular to each problem and its application, and their effect on a thinning algorithm can be estimated by experiment.

Thinning as a morphological process moves across the input image, pixel by pixel and its neighbors are logically compared against a structuring element, to determine the output pixel's logical value. The structuring element is generally composed of square dimensions of size 3*3 pixels, 5*5 pixels and sometimes greater, depending on the application.

Bwmorph(); is a function available in Matlab, which is able to apply a specific morphological operation to a binary image. *bwmorph* requires (imageName, morphologicalOperation, numberOfAppliedOperations). In this case the operator is thinning and it is repeated 2,3,4,5 and *Inf* times through experimentation. The *Inf,* which is used in the MATLAB, refers to running the iterations until the skeleton no longer changes. The MATLAB script that performs thinning is attached in Annex VII.

(a) (b) (c) (d) (e) (f)

Figure 4.8: Thinned characters with different iteration levels. (a) Original character and thinned at (b) 2, (c)3, (d)4, (e)5 and (f) Inf iteration levels.

Through experimentation, an iteration level of *"Inf"* is found to be considered in this thesis work as this level maintains the connectivity of characters at a single pixel and did not cause excessive erosion of the region.

4.4. CREATING CHARACTER REPRESENTATION

After the normalized characters are thinned the binary representation of each character must be created and saved to a file. This file containing a binary representative of the identified characters will be fed to the neural network procedure that is coded in MATLAB programming environment. The 20×20 normalized character image in an enclosing box is stored in a vector for

each character during character thinning for further processing. Sample binary representation for the “ሚ” characters is presented as follows.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 4.9. Sample Binary Representation for “ሚ” Character

4.5. CHARACTER RECOGNITION

Artificial Neural Networks (ANN), with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques (Stergiou and Siganos, 2002).

While template matching, structural analysis and neural networks have been very popular classification techniques for character recognition, neural networks are now increasingly used in printed character recognition applications. No single neural network model appears to be

inherently better than others to a significant extent; instead, higher accuracy rates can be achieved by tailoring network models to the particular problem environment (Haykin, 1994).

The topology preferred to this work is the Multilayer Perceptron (MLP) topology; that yielded very accurate results and the execution time was shorter as compared to other structures that were evaluated (Jain *et.al.*, 1996).

This section describes how the artificial neural network is created, trained and refined to classify an input pattern as one of the target output results. The overall task of the recognizer is to be able to clarify new input to one of the possible output values. Since the neural net tool used for the recognizer is the MATLAB Neural Network tool Box, the discussion on how to design, creates, train and test the network will be in the domain of the tool and its corresponding functions.

In creating and training a neural network, the primary task is coming up with the architecture of the network. The architecture of the neural network created for the recognition is a multilayer perceptron that accepts the binary representation of the segmented character as an input vector. The binary representation of each segmented character has equal size after the normalization procedure, which creates a standard size that corresponds to the number of input nodes in the input layer of the network.

The following graphical presentation gives the visual impression of the overall architecture of the recognition system starting from the image input to the recognition phase. In this graphical representation, the development tool used for the respective task is also indicated.

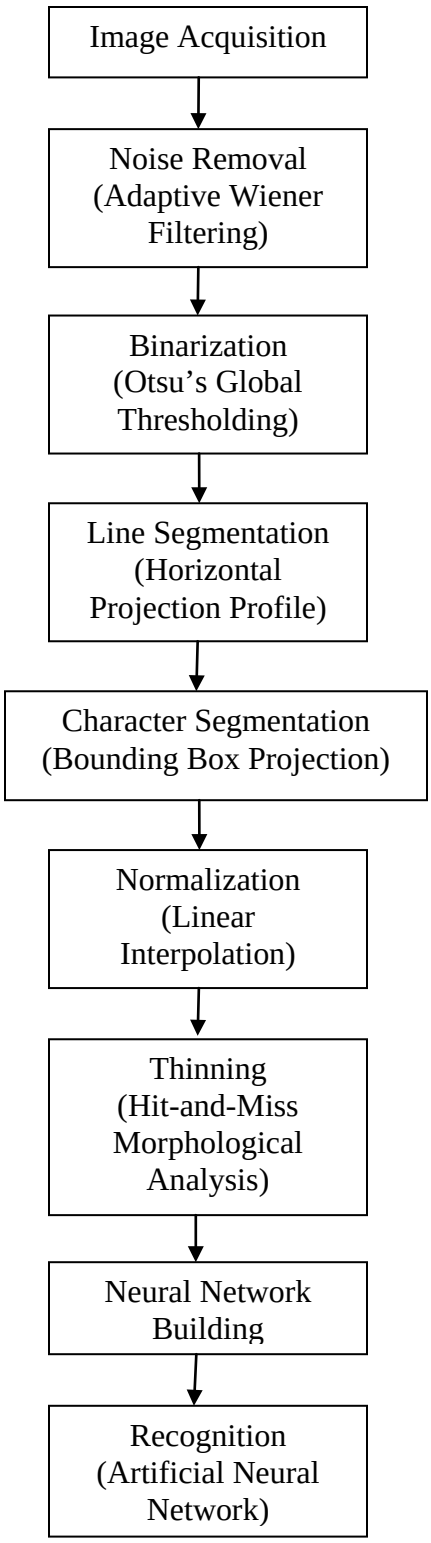


Figure 4.10. General Architecture of the Recognition Network.

4.5.1. The Neural Network

The network created is a feed forward neural net with backpropagation algorithm. The approach adjusts the strength of connection between nodes at different layers after computing the error between the desired and the actual output of the network at each training iteration. The learning method is then supervised learning where the desired output of the network for the know patterns of input is also fed to the network during training. By doing so the network takes inputs, performs some operation and provides the output using the feed forward approach. Then the mean squared error between the desired and the actual output is calculated. If this value is found to be greater than the threshold, the error propagated back to adjust the weights of each connection in the course of finding the optimal weight for each connation.

4.5.2. Character Input Method for the Neural Network

The first step in training the network is rearranging the input pattern into a column vector containing elements equal to the number of neurons at the input layer. For this purpose, the binary representation of the characters extracted during the preprocessing stages, is converted into one column vector where each preceding column of the original character matrix will be appended to the previous one. For 20x20 character representation we will have 400 input elements and 400 binary values to represent a single character.

In deciding the number of nodes at the output layer, since the number of characters to be recognized is the 33 basic characters with their 6 forms (that is a total of 7x33 = 231) and 44 labialized characters, totally 275, it can be represented by a continuous value between 0 and 1. While assigning 275 characters a continuous value, unique continuous values for each character

are created between 0 and 1. Accordingly, 0 is assigned for the first value and 1 is for the last 275^{th} character. Thus, the number of neurons at the output layer is fixed to be 1. After deciding on the possible number of nodes at each layer, the next phase is creating the network and proceeding with the training part, which increases the generalization power of the network.

4.5.3. Creating the Neural Network

The network created is equipped with these two parameters, learning rate and momentum, for two major reasons. The inclusion of learning rate is to specify how to make adjustment to the weights after calculating the errors on the output layer. The principle behind learning rate is that, instead of adjusting the weight after each input pattern, why not make the adjustment after set of patterns are fed to the network. For example, we can set the learning rate to be 0.02, and then it will take 200 patterns to make a 20% adjustment on the strength of connections between nodes of different layers.

Likewise, the purpose served by the momentum parameter is that, once the learning network knows to which direction to move on the learning curve, we can adjust the speed of the network for fast convergence. The principle is that, once the direction of convergence of the learning model is known moving fast towards the minima point will speedup the learning process which helps to minimize the total time required by the network to learn different patterns. In addition to fast convergence, the momentum constant will help the network not to get stuck at local minima. A local minimum is a point in the learning curve where the network error is the minimum only to points in its vicinity and not to the total learning curve.

If the momentum constant is set too low, the network will be slow and may get stack somewhere in the learning curve due to local minima. On the other hand, if it is set to be a large number, the network will have less probability of getting trapped in local minima and will converge fast, but it may also swing from one point to the other in the curve without touching the global minima (that is called an oscillation).

In MATLAB environment this facility is accessible by training the network with TRAINGDX keyword. The TRAINGDX is a network training function that updates weight and bias values according to gradient descent momentum and with adaptive learning rate. There were other network-training functions like TRAINGD, which do not have the power of controlling the momentum while navigating through the learning curve. To create such network using MATLAB, the following command is used.

Network=newff(input_range1,[40 1],{'tansig','purelin'},'traingdx');

Where:

Network; *the network is created with 40 hidden and 1 output layers with activation function 'tansig' and 'purelin' respectively.*

input_range1; *is the minimum and maximum values at the input nodes.*

[40 1]; *is size of neurons at hidden and output layer respectively.*

{'logsig','purelin'}; *is the activation function at the hidden and output layers.*

*'**traingdx';** is the learning function of the network.*

4.5.4. Training the Neural Network

After creating the neural network, the next step is making the network to be introduced to different patterns of the whole character set. After different network parameters are adjusted as per the performance of the network, the training input patterns along with matrix of target values will be fed to the network for training. The line of instruction to perform this task is:

[Network,tr]=train(Network,TrainSet,TargetSet);

Where:

Network; *is the network created using the instruction in section 4.5.3.*

TrainSet*; is matrix where each column represents one character from the set.*

TargetSet*; is pattern of target output for each corresponding input.*

tr; *training record (epoch and performance).*

For training the network with different patterns of the Amharic character set, eight (two from each) sets of characters are collected from the "Addis Zemen" newspaper, "Federal Negarit Gazeta", from the Bible ,and "Fikir Eskemekabir". For training the network, from eight training sets, a total of 1090 (1087 core and 3 labialized) characters are fed to the network. Out of the 275 (231 core and 44 basic) characters to be considered in the problem domain of this work, only 143 characters (52% of the 275 characters) are found in the training set. This is because the inclusion of few number of training sets due to time constraint. After creating the network and training it with the training samples available, the performance is tested on four testing sets from each sample. On the course of testing the recognition engine from the new testing sets, the characters

that are not included in the training sets fall in the testing set. This problem commands a negative impact on the overall recognition.

4.5.5. Testing the Neural Network

After training the artificial neural network is accomplished, it is required to test the performance of the modeled network with different test cases. In testing the network, two approaches were implemented. The first one is testing the network with the pattern that the network is trained with. This testing scenario helps to see if the network can recognize the original training pattern correctly. The second approach is evaluating the network performance with new input patterns. This testing scheme as well helps to see how powerful the network is in generalization when faced with new character patterns.

To create a match-up between the output and the target values, the 33 basic characters with their 6 forms (that is a total of 7x33 = 231) and 44 labialized characters, totally 275 characters, are assigned a number between 0 and 1 starting from the first character. The value of each fraction between 0 and 1 for 275 characters is assigned. Thus, for all the test cases what the network does is; it provides a fraction value between 0 and 1 as the number of output node is set 1 in the network definition. These fraction values are converted to its decimal equivalent value and the character with that decimal number representation is taken as an output of the network for that specific input pattern. Thus the testing of the network created and trained at sections 4.5.3 and 4.5.4 is tested using the following code:

Rst=sim (Network, TrainSet);

Where:

Network; *is the network created and trained for the training sets.*

TrainSet; *is the testing input pattern.*

Rst; *returned output in matrix format.*

4.5.5.1. Test with Training Set

To see whether the network can recognize the character set used for training, from eight training sets, a total of 1090 (1087 core and 3 labialized) characters are fed to the network. The output of the network is presented in the following table:

Test Sets	Performance /Rate of Recognition
Training Set from Addis Zemen	97.88 %
Training Set from Negarit Gazeta	97.84 %
Training Set from the Bible	96.41 %
Training Set from Fikir Eskemekabir	95.38 %
Average Accuracy	96.87 %

Table 4.2. Network Performance for Training Set

4.5.5.2. Test with New Test Sets

After the network is tested for the set of characters it is trained with, the network is also tested with new text taken from the four sources. This test helps to see the response of the network to new character input patterns that the network never faced during the training phase. Thus, the network is tested for a total of 548 (543 core and 5 labialized characters) new characters, 542 of them are those included in the training set and the other 6 are found not included in the training set. The outputs of these 6 characters are negative numbers which are not assigned to the characters in the training set. The recognition rate of the network is presented in the following table.

Test Sets	Performance /Rate of Recognition
Test Set from Addis Zemen	14.88 %
Test Set from Negarit Gazeta	10.25 %
Test Set from the Bible	11.04 %
Test Set from Fikir Eskemekabir	9.44 %
Average Accuracy	11.40 %

Table 4.3: Network Performance for New Test Set

4.6. FINDINGS

The errors encountered in the test results of the developed recognition system can be seen in two broad views: segmentation error and classification /recognition error. Segmentation error is an error occurred due to the segmentation algorithm implemented for this study. The segmentation error existed in the document was considering one character as more characters. This type of error exists in numbers and punctuations.

The classification/recognition error is an error occurred when the developed system wrongly classifies/recognizes characters. The developed Amharic OCR system classified characters in to one of the following. The system:

- may recognize the characters correctly as expected.
- May recognize the characters incorrectly, but the output having similar shape with the expected.
- may recognize the characters incorrectly, with no similarity between the output and the expected.
- may classify the characters unknown (for characters not included in the training set).

In testing the developed system from the training set, from eight training sets, a total of 1090 (1087 core and 3 labialized) characters are fed to the network. The system correctly recognized averagely 96.87 % of the characters. That means, 3.13 % of the data was recognized incorrectly because of the existence of structurally similar characters. For instance, ኳ was recognized as ከ; ሰ was recognized as ለ.

On the other hand, in testing the developed system using new test set, on the average the system correctly classified 11.40% of the characters. Most of the characters are misclassified and basically we observe two types of errors. The first type of error happens because of shape similarity between characters. This error constitutes 73.6% of the total errors. For instance, ላ was recognized as ሳ. The second type of error is related to character deformation during highly corrupted image noise removal. For instance, ሕ was recognized as ሐ. This type of error accounts for 15% of the total errors. Correcting such errors require reconstructing the image after noise removal.

In general, observation of the test results show that the performance of the system is greatly affected by the similarity of the shape of Amharic characters and effect of the application of noise removal for cleaning highly degraded document images. Such challenges require to further explore an invariant to shape feature extraction techniques and advanced noise detection and removal algorithms.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. CONCLUSION

The introduction of the printing press around 1900 encouraged the proliferation of books. Books and newspapers were printed using type-setting technology. With the advent of computer technology, it became possible to efficiently produce printed documents having Ethiopic characters without the use of a printing press. According to the official web site of the National Library of Ethiopia (http://www.nale.gov.et/index.html) the total numbers of collection of books in the National Library of Ethiopia are about 109,990.

Character recognition from document images that are printed in Ethiopic script is a challenging task. To develop successful character recognition system for Amharic script, there are some issues need to be addressed. These are the degradation of documents, printing variations, large number of characters in the script and visual similarity of some characters in the script.

In this study, different preprocessing algorithms like adaptive noise removal, global Otsu thresholding, stage-by-stage segmentation, normalization, thinning techniques and neural network classifier methods has been experimented and used to the best of the recognizer network. During experimentation of the applicability of these algorithms and approaches, the wiener adaptive filtering method for noise removal, Otsu global thresholding method for

binarizing the digitized image, linear interpolation techniques for normalization and hit-and-miss morphological analysis for thinning are found to work very well for the problem of interest.

In Amharic writing system a line of Amharic script lays at the same level, having no ascent and descent features. This feature of the Amharic writing system makes the stage-by-stage segmentation algorithm to fit remarkably. As a result, stage by stage segmentation algorithm suggested by Pal and Chaudhuri (1995) was used in this thesis work. The bounding box projection approach for character segmentation under stage by stage algorithm has advantages over the pixel projection approach. It is less computationally intensive (Liang *et. al.,* 1997). Therefore, this approach to character segmentation was implemented using MATLAB image processing toolbox, which extracts the character image in its bounding box. Each character is considered as a connected object in the image, which was segmented as a line of text. The method is found to segment for basic and labialized 98.28% and 100% efficiency respectively for training character sets, 98.55% and 100% efficiency respectively for testing character sets.

For classifying different input patterns as one of the 275 Amharic characters, an artificial neural network approach is implemented. The network developed is with feed forward backpropagation error correction scheme where desired and actual output errors are computed and used for refining the network. In MATLAB programming environment, different training methods are explored and the one with adaptive learning and momentum parameters (TRAINGDX) is used. A performance average recognition rate of 11.40% is observed for new test sets. This is due to high similarity between Amharic characters and deformation of the shape of characters during noise removal.

5.2. RECOMMENDATIONS

In order to develop a real-life printed Amharic OCR system, the following recommendations are forwarded for further research.

1. The performance of the system is greatly affected by the similarity of the shape of Amharic characters. Such challenges require for further exploring an invariant shape feature extraction techniques.
2. The present research attempts to integrate noise detection and removal techniques for the recognition of real-life documents. However, as the level of noise increases the removal scheme designed in this study may not produce satisfactory result. Hence there is a need to apply advanced noise detection and removal algorithms for highly degraded Amharic document images.
3. The segmentation algorithm used in the current study worked reasonably for basic and labialized characters. But it fails to segment special character |___ |, punctuations and numbers that are formed by two or more strokes, loops and lines in between which there is a space. Therefore, a better segmentation algorithm that tolerates space between connected characters.
4. Skewed and slant correction for the problem domain in this work was done using Microsoft paint rotation toolbox. Additional preprocessing algorithms for skew detection and slant correction should be considered for better performance.
5. As implemented in this work, only line and character segmentation are considered. Therefore, it is worth to implement word segmentation for a more meaningful result.

REFERENCES

1. Abass Alemneh (1994). "***The Need for a Standardization of Ethiopian Script***": Proceedings of EthCITA E-mail Conference, Volume III paper 3.
2. Aishy Amer (2002). ***"New Binary Morphological Operations for Effective Low-cost Boundary Detection***": International Journal of Pattern Recognition and Artificial Intelligence, Vol. 17, No. 2, pp.1-13.
3. Bender et *al*. (1976). ***"Language in Ethiopia"***: London, Oxford University Press,UK.
4. Berhanu Aderaw (1999). ***"Amharic Character Recognition using Artificial Neural Networks"***: (Masters Thesis), Department of Electrical Engineering, Addis Ababa University, Addis Ababa.
5. Casey, R.G. and Nagy, G. (1982). ***"Recursive Segmentation and Classification of Composite Character Patterns"***, International Conference on Pattern Recognition. Vol.2, No. 32, pp.1023-1026.
6. Chen Y. and G. Leedham (2005). ***"Decompose Algorithm for Thresholding Degraded Historical Document Images"***: IEEE Proceedings on Visual, Image Signal Processing, Vol. 152, No. 6. pp. 989-996.
7. Cheoi Kyung Joo (2007). ***"A New Binarization Method of Text Images Using Image Enhancement Techniques"***: Journal of the Research Institute for Computer and Information Communication, Vol.15, No.2, pp.33-40.
8. Chi Chang-yanab *et.al.*, (2005). ***"Study on Methods of Noise Reduction in a Striped image"***: College of surveying and Mapping, Shandong University of Science and Technology, Qingdao, P.R. China.

9. Cowell, J. and Hussain, F. (2003). "***Amharic Character Recognition Using a Fast Signature Based Algorithm***": Proceedings of the Seventh International Conference on Information Visualization, pp. 384-389.
10. Dawit Bekele (2003). ***"The Development and Dissemination of Ethiopic Standards and Software Localization for Ethiopia"***: The ICT Capacity Building Programme of the Capacity Building Ministry of the FDRE and United Nations Economic Commission for Africa, Addis Ababa.
11. Dereje Teferi (1999). ***"Optical character Recognition of Typewritten Amharic Text":*** (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa**.**
12. Ermias Abebe (1998). ***"Recognition of Formatted Amharic Text using Optical Character Recognition"***: (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa.
13. Genovese, J. A. (1970). ***"Character Recognition***": Encyclopedia of Library and Information Science, Vol. 4, New York, Marcel Deker Inc, USA.
14. Gonzalez, C.R. and E.R. Woods (2002). ***"Digital Image Processing"***: 2nd edition, Prentice Hall, Upper Saddle River, NJ, USA.
15. Green, W. B. (1993). "***Introduction to Electronic Document Management System***": Boston, Academic Press Inc, USA.
16. Haykin S. (1994). ***"Neural Networks: A comprehensive Foundation"***: Prentice Hall, Inc., New Jersey, USA.

17. Hilditch C.J. (1969). ***"Linear skeletons from square cupboards,"***: Machine Intelligence, Vol. 4,No.4, pp. 403-420, B. Meltzer and D. Michie, eds., New York, American Elsevier, USA.
18. Liang *et al.* (1997). ***"Document Layout Structure Extraction using Bounding Boxes of Different entities"***: Elsevier Publishers, USA.
19. Jain *et al*. (1996). ***"Artificial Neural Network: A Tutorial"***: IBM Almaden Research Center, Michigan, USA.
20. Karunanayaka *et al*. (2005). ***"Thresholding, Noise Reduction and Skew Correction of Sinhala Handwritten Words"***: IAPR Conference on Machine Vision Applications, Tsukuba Science City, Japan.
21. Lam, *et al*. (1992). ***"Thinning Methodologies- A Comprehensive Survey"***: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 9, pp. 869-885.
22. MathWorks, (2010). ***"MATLAB 7 Image Processing Users' Guide"***: The MathWorks, Inc,USA.
23. McCulloch and W. Pitts (1993) "A Logical Calculus of Ideas Immanent in Nervous Activity" : Bulletin Mathematical Bio-physics, Vol.5,No.1, pp.115-133.
24. Mesay Hailemariam (2003). "***Line fitting to Amharic OCR*: *The Case of Postal Address***": (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa, Ethiopia.
25. Million Meshesha (2000). "***A Generalized Approach to Optical Character Recognition of Amharic Texts***": (Masters Thesis), School of Information studies for Africa, Addis Ababa University, Addis Ababa, Ethiopia.

26. Million Meshesha and Jawahar, C. V. (2005). "***Recognition of Printed Amharic Documents***": Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR), Vo. 21, No.8-10, pp. 784-788.
27. Min Hu, Jieqing Tan , Feng Xue, (2005). ***"A New Approach to the Image Resizing Using Interpolating Rational-Linear Splines by Continued Fractions"***: Journal of Information & Computational Science: Vol.2, No. 4, pp. 681-685.
28. Motwani *et al.* (2004***). "Survey of Image Denoising Techniques"***: Proceedings of GSPx, Santa Clara Convention Center, Santa Clara, CA.
29. Mori, S., Suen, C., and Yamamoto, K. (1992). "***Historical Review of OCR Research and Development***": Proceedings of the IEEE, Vol. 80 No.7, pp. 1029-1058.
30. Negussie Taddesse (2000). "***Handwritten Amharic Text Recognition Applied to the Processing of Bank Cheques***": (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa, Ethiopia.
31. Ntogas Nikolaos and Ventzas Dimitrios (2008). ***"A Binarization Algorithm for Historical Manuscripts"***: 12th International Conference on Communications, Heraklion, Greece.
32. Otsu, N. (1979). ***"A Threshold Selection Method from Gray-level Histograms"***: IEEE Trans. Systems, Man, and Cybernetics, Vol. 9, No. 1, pp 62-66.
33. Pal and Chaudhuri (1995). "***A Complete Printed Bangla OCR System"***: Pattern Recognition, Vol.31, No.5, pp.531-549.
34. Pandy Abhijits S., Macy Roberts B. (1996). ***"Pattern Recognition with Neural Networks in C++"***: CRC Press,USA.

35. Qing Chen (2003). ***"Evaluation of OCR Algorithms for Images with Different Spatial Resolutions and Noises"***: Journal of the Research Institute for Computer and Information Communication, Vol.15, No.3, pp. 45-51.
36. Ralston *et al.* (2000). ***"Optical Character Recognition"***: Encyclopedia of Computer Science, 4th ed., Nature Publishing Group,USA.
37. Seong-Whan Lee *et al.,* (1996). ***"A New Methodology for Gray-Scale Character Segmentation and Recognition"*** IEEE Transactions on pattern analysis and machine intelligence, Vol. 18, No. 10, pp. 102-109.
38. Stergiou, C. and Siganos, D. (2002). "***Neural Networks***": URL: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/Vol4/cs11/report.html. Retrieved on March, 2010.
39. Srihari, S. and Lam, S. (1996). "***Character Recognition***": Amherst, NY, Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo. URL:http://www.cedar.buffalo.edu/TechReps/OCR/ocr.html. Retrieved on April, 2010.
40. Taylor J.,G. (1995). ***"Neural Networks"*** Unicom Ltd., London,UK.
41. Teshome Alemu (2008). ***"Recognition of Amharic Braille"***: (Masters Thesis), Department of Information Science, Addis Ababa University, Addis Ababa, Ethiopia.
42. Verlag Harrassowitz (2003). ***"Amharic Literature"***: Encyclopedia Aethiopica, Vol.1, A-C, Wiesbaden, Germany.
43. Vernon D. (1991). ***"Machine Vision"***: Prentice-Hall, USA.

44. Wondwossen Mulugeta, (2004). "***OCR for Special Type of Handwritten Amharic Text ("YKUM TSEFET")***: (Masters Thesis), Department of Information Science, Addis Ababa University, Addis Ababa, Ethiopia.
45. Worku Alemu (1997). "***The Application of OCR Techniques to the Amharic Script***": (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa, Ethiopia.
46. Worku, A., and Fuchs, S. (2003). "***Handwritten Amharic Bank Check Recognition Using Hidden Markov Random Field***": Document Image Analysis and Retrieval Workshop, pp. 28.
47. Yaregal Assabie (2002)."***Development of Versatile Character Recognition System for Amharic Text***": (Masters Thesis), School of Information Studies for Africa, Addis Ababa University, Addis Ababa, Ethiopia.
48. Yaregal, A. and Bigun, J. (2006). "***Ethiopic Character Recognition Using Direction Field Tensor"***: 18th International Conference on Pattern Recognition (ICPR), pp. 284-287.
49. Zhang T.Y. and C.Y. Suen (1984). ***"A Fast Parallel Algorithm for Thinning Digital Patterns,"*** : Communication ACM, vol. 27, No. 3, pp. 236-239.

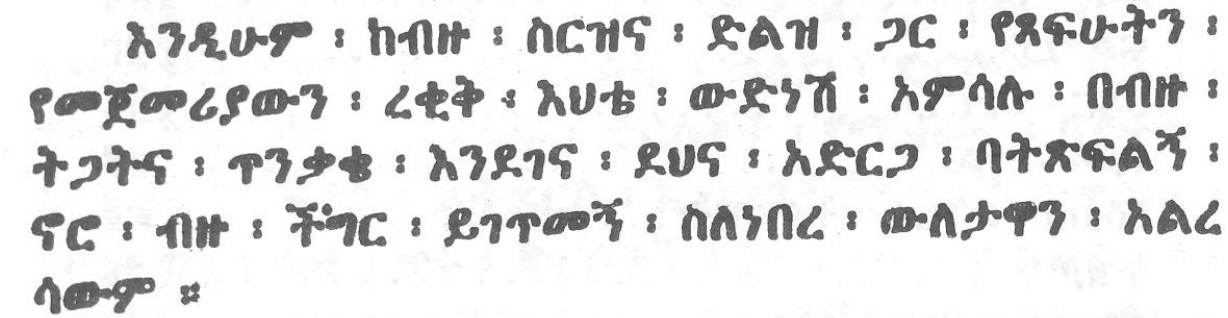
APPENDICES

Annex- I. The Amharic Character Set (Bender et al., 1976)

Order							Labialized				
1st	2nd	3rd	4th	5th	6th	7th					
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ			ሏ		
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ					
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ			ሟ		
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ			ሯ		
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ			ሷ		
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ			ሿ		
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቊ	ቋ	ቌ	ቍ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ			ቧ		
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ			ቷ		
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ			ቿ		
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኊ	ኋ	ኌ	ኍ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ			ኗ		
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ			ኟ		
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ					
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኲ	ኳ	ኴ	ኵ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ					
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ			ዟ		
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ			ዧ		
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ			ዷ		
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ					
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጒ	ጓ	ጔ	ጕ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ			ጧ		
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ			ጯ		
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ			ጷ		
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ					
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ			ፏ		
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ					

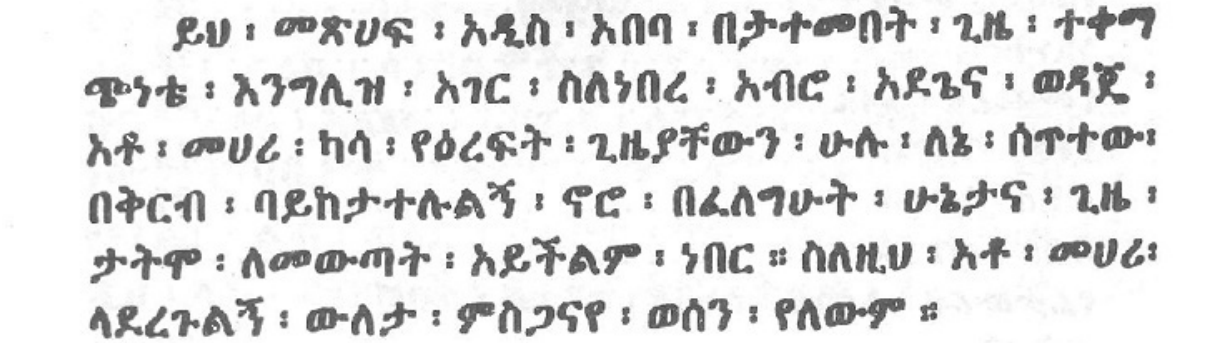
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Annex- II. The Amharic Texts Included in the Training Set



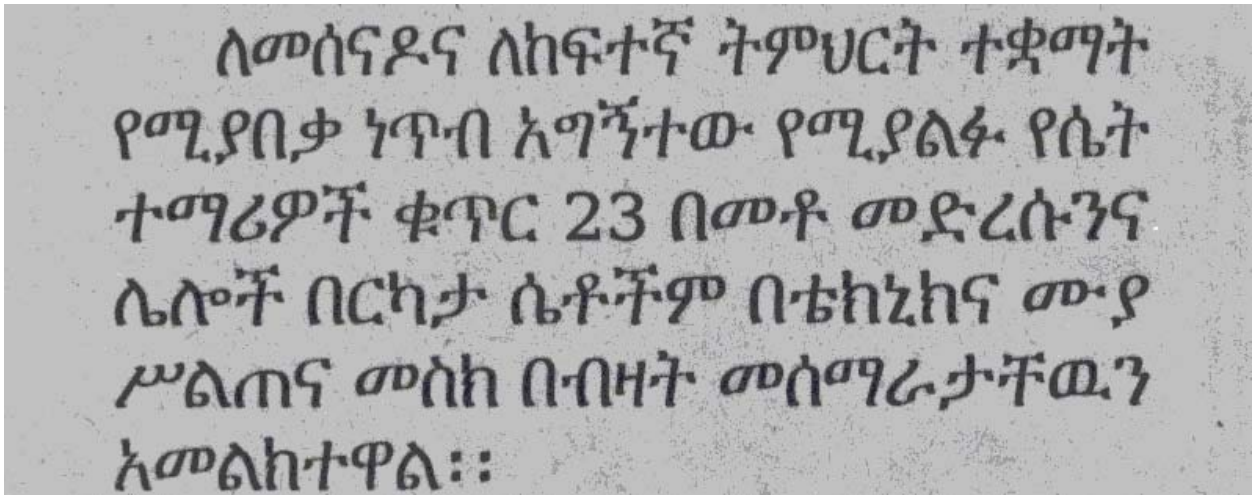
እንዲሁም ፡ ከብዙ ፡ ሰርዝና ፡ ድልዝ ፡ ጋር ፡ የጻፍሁትን ፡ የመጀመሪያውን ፡ ረቂቅ ፡ እህቴ ፡ ውድነሽ ፡ አምሳሉ ፡ በብዙ ፡ ትጋትና ፡ ጥንቃቄ ፡ እንደገና ፡ ደህና ፡ አድርጋ ፡ ባትጽፍልኝ ፡ ኖሮ ፡ ብዙ ፡ ችግር ፡ ይገጥመኝ ፡ ስለነበረ ፡ ውለታዋን ፡ አልረሳውም ።

Train Set 1. Text from “Fikir Eskemekabir”



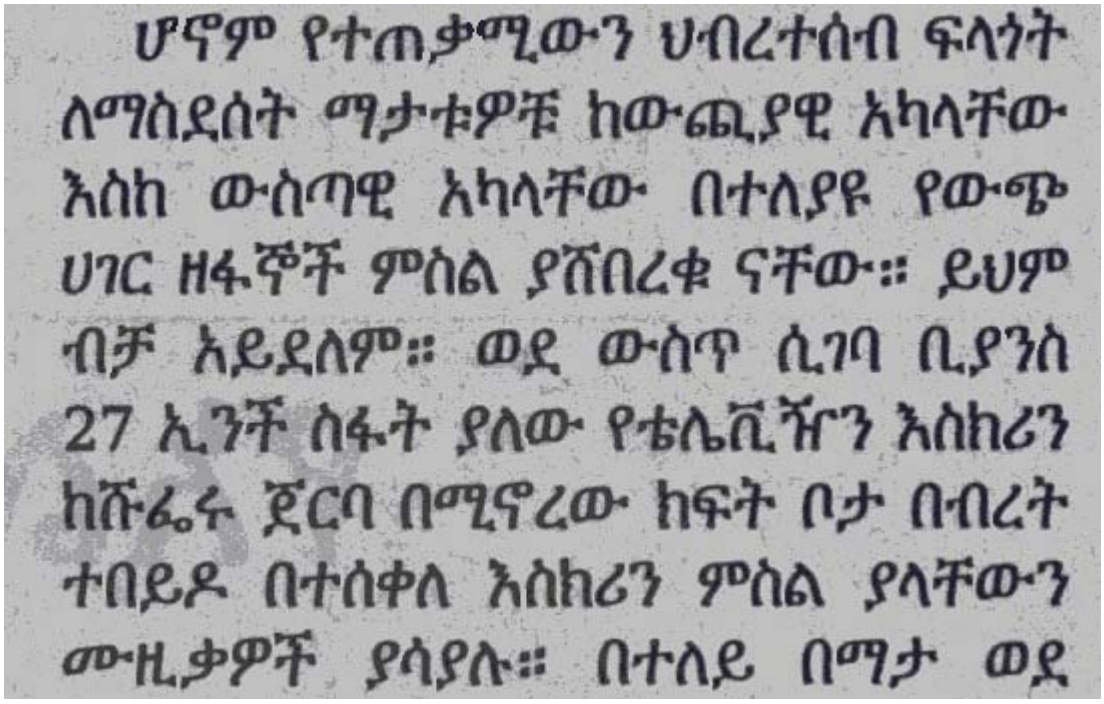
ይህ ፡ መጽሀፍ ፡ አዲስ ፡ አበባ ፡ በታተመበት ፡ ጊዜ ፡ ተቀማጭነቴ ፡ እንግሊዝ ፡ አገር ፡ ስለነበረ ፡ አብሮ ፡ አደጌና ፡ ወዳጄ ፡ አቶ ፡ መሀሪ ፡ ካሳ ፡ የዕረፍት ፡ ጊዜያቸውን ፡ ሁሉ ፡ ለኔ ፡ ሰጥተው፡ በቅርብ ፡ ባይከታተሉልኝ ፡ ኖሮ ፡ በፈለግሁት ፡ ሁኔታና ፡ ጊዜ ፡ ታትሞ ፡ ለመውጣት ፡ አይችልም ፡ ነበር ። ስለዚህ ፡ አቶ ፡ መሀሪ፡ ላደረጉልኝ ፡ ውለታ ፡ ምስጋናየ ፡ ወሰን ፡ የለውም ።

Train Set 2. Text from “ Fikir Eskemekabir”

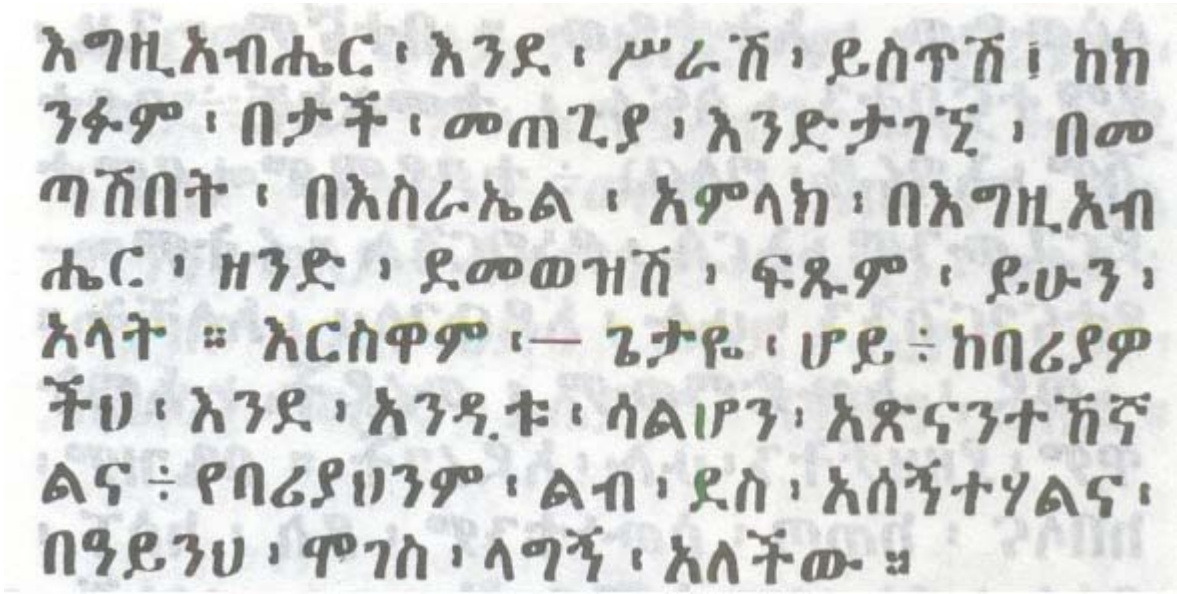


ለመሰናዶና ለከፍተኛ ትምህርት ተቋማት
የሚያበቃ ነጥብ አግኝተው የሚያልፉ የሴት
ተማሪዎች ቁጥር 23 በመቶ መድረሱንና
ሌሎች በርካታ ሴቶችም በቴክኒክና ሙያ
ሥልጠና መሰክ በብዛት መሰማራታቸዉን
አመልክተዋል::

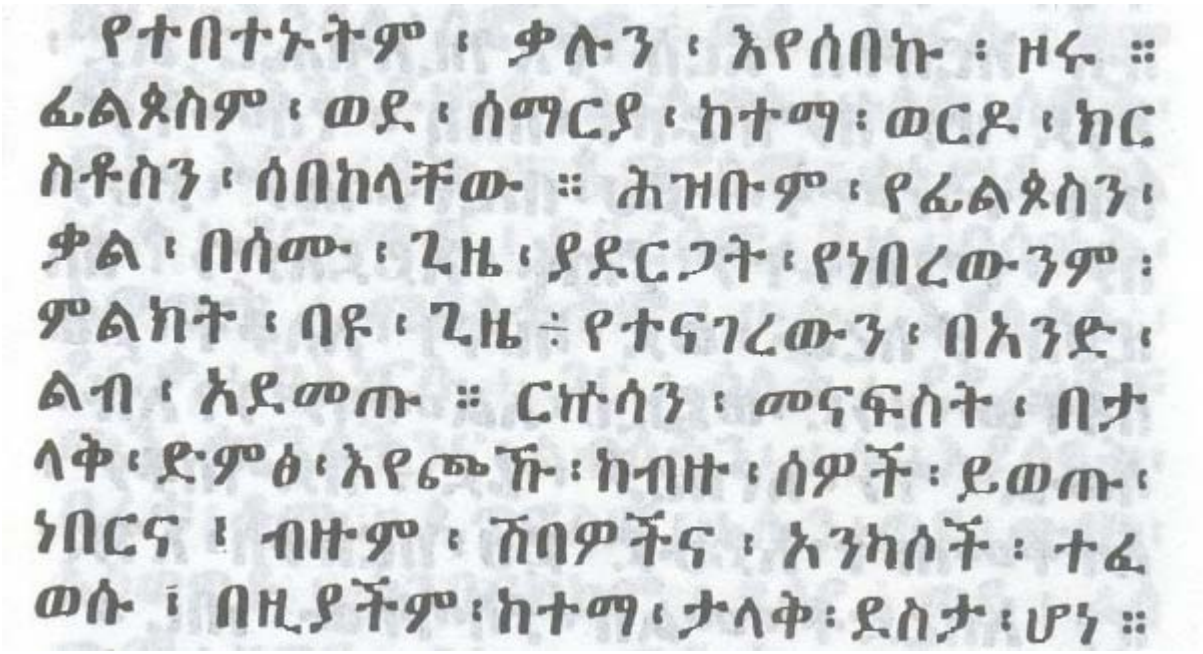
Train Set 3. Text from “Addis Zemen”


ሆኖም የተጠቃሚውን ህብረተሰብ ፍላጎት
ለማስደሰት ማታቱዎቹ ከውጪያዊ አካላቸው
እስከ ውስጣዊ አካላቸው በተለያዩ የውጭ
ሀገር ዘፋኞች ምስል ያሸበረቁ ናቸው። ይህም
ብቻ አይደለም። ወደ ውስጥ ሲገባ ቢያንስ
27 ኢንች ስፋት ያለው የቴሌቪዥን እስክሪን
ከሹፌሩ ጀርባ በሚኖረው ክፍት ቦታ በብረት
ተበይዶ በተሰቀለ እስክሪን ምስል ያላቸውን
ሙዚቃዎች ያሳያሉ። በተለይ በማታ ወደ

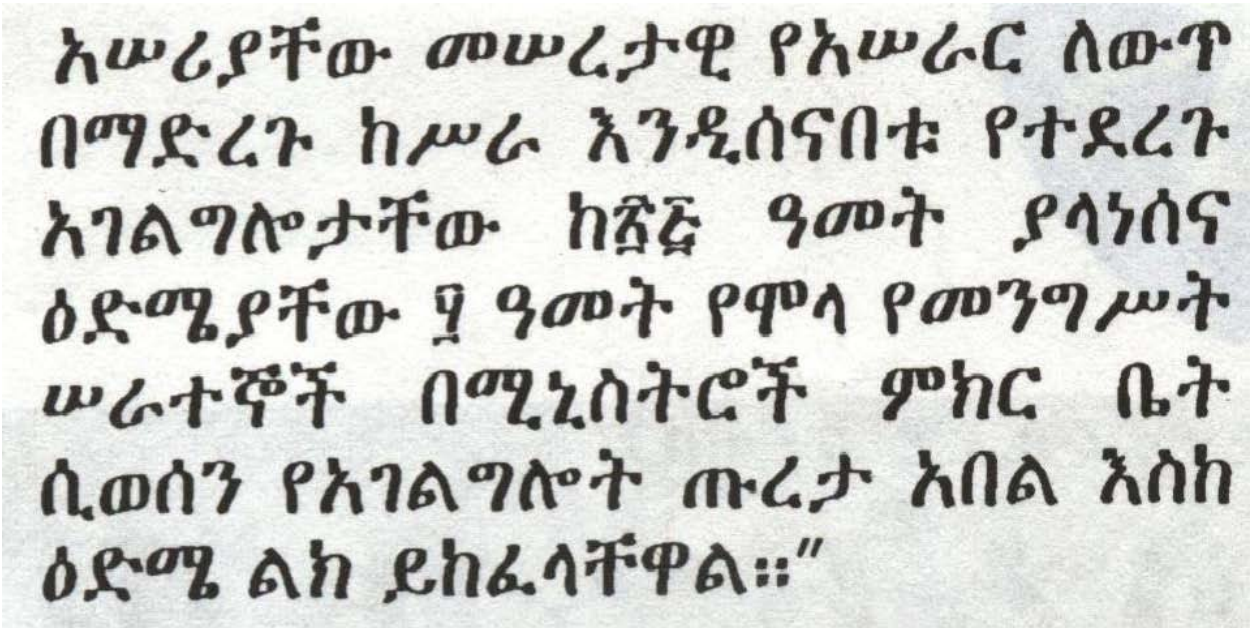
Train Set 4. Text from “Addis Zemen”


እግዚአብሔር ፣ እንደ ፣ ሥራሽ ፣ ይስጥሽ ፣ ከክ
ንፉም ፣ በታች ፣ መጠጊያ ፣ እንድታገኚ ፣ በመ
ጣሽበት ፣ በእስራኤል ፣ አምላክ ፣ በእግዚአብ
ሔር ፣ ዘንድ ፣ ደመወዝሽ ፣ ፍጹም ፣ ይሁን ፣
አላት ። እርስዋም ፣— ጌታዬ ፣ ሆይ ÷ ከባሪያዎ
ችህ ፣ እንደ ፣ አንዲቱ ፣ ሳልሆን ፣ አጽናንተኸኛ
ልና ÷ የባሪያህንም ፣ ልብ ፣ ደስ ፣ አሰኝተሃልና ፣
በዓይንህ ፣ ሞገስ ፣ ላግኝ ፣ አለችው ።

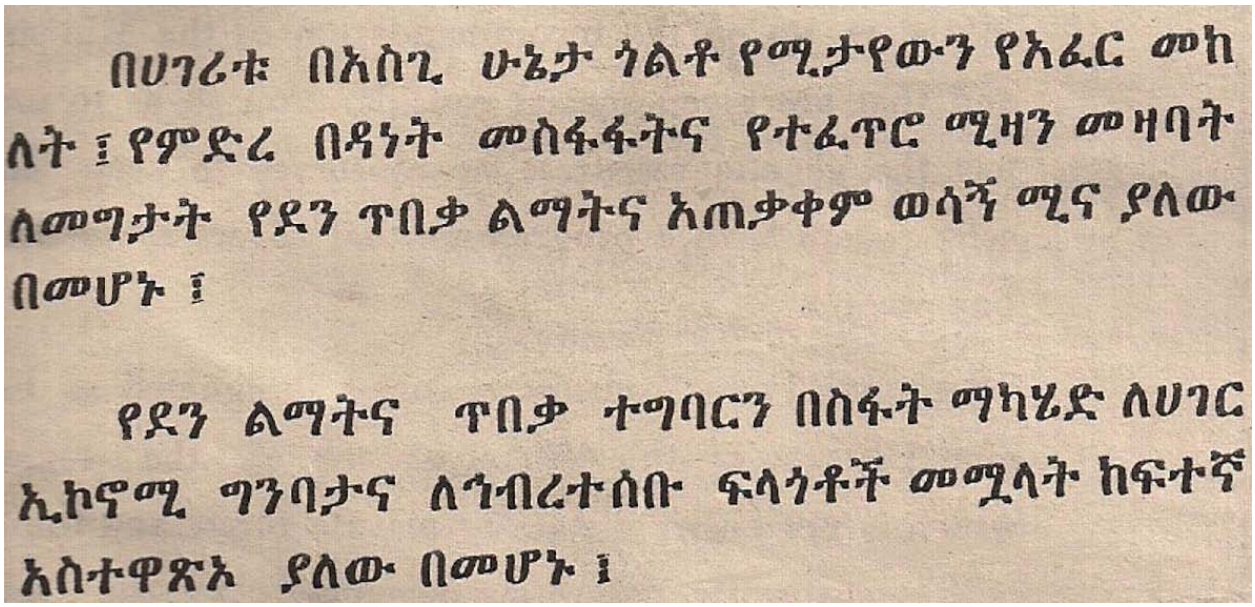
Train Set 5. Text from “Bible”


የተበተኑትም ፣ ቃሉን ፣ እየሰበኩ ፡ ዞሩ ።
ፊልጶስም ፣ ወደ ፣ ሰማርያ ፣ ከተማ ፡ ወርዶ ፣ ክር
ስቶስን ፣ ሰበከላቸው ። ሕዝቡም ፡ የፊልጶስን ፡
ቃል ፣ በሰሙ ፣ ጊዜ ፣ ያደርጋት ፣ የነበረውንም ፡
ምልክት ፣ ባዩ ፣ ጊዜ ÷ የተናገረውን ፣ በአንድ ፣
ልብ ፣ አደመጡ ። ርኵሳን ፡ መናፍስት ፣ በታ
ላቅ ፣ ድምፅ ፡ እየጮኹ ፡ ከብዙ ፡ ሰዎች ፡ ይወጡ ፡
ነበርና ፤ ብዙም ፡ ሽባዎችና ፡ አንካሶች ፡ ተፈ
ወሱ ፣ በዚያችም ፡ ከተማ ፡ ታላቅ ፡ ደስታ ፡ ሆነ ።

Train Set 6. Text from " Bible"


አሠሪያቸው መሠረታዊ የአሠራር ለውጥ
በማድረጉ ከሥራ እንዲሰናበቱ የተደረጉ
አገልግሎታቸው ከሿኜ ዓመት ያላነሰና
ዕድሜያቸው ፶ ዓመት የሞላ የመንግሥት
ሠራተኞች በሚኒስትሮች ምክር ቤት
ሲወሰን የአገልግሎት ጡረታ አበል እስከ
ዕድሜ ልክ ይከፈላቸዋል።"

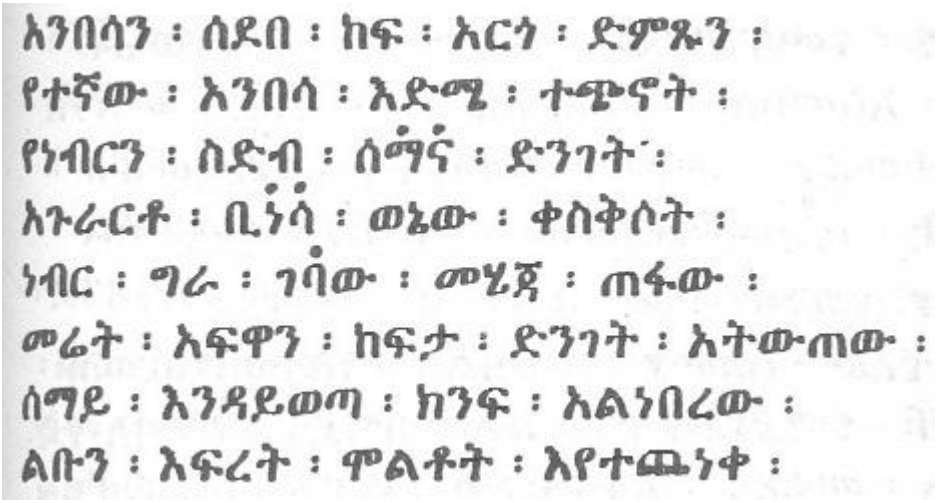
Train Set 7. Text from " Federal Negarit Gazeta"


በሀገሪቱ በአሰጊ ሁኔታ ጎልቶ የሚታየውን የአፈር መከ
ለት ፤ የምድረ በዳነት መስፋፋትና የተፈጥሮ ሚዛን መዛባት
ለመግታት የደን ጥበቃ ልማትና አጠቃቀም ወሳኝ ሚና ያለው
በመሆኑ ፤

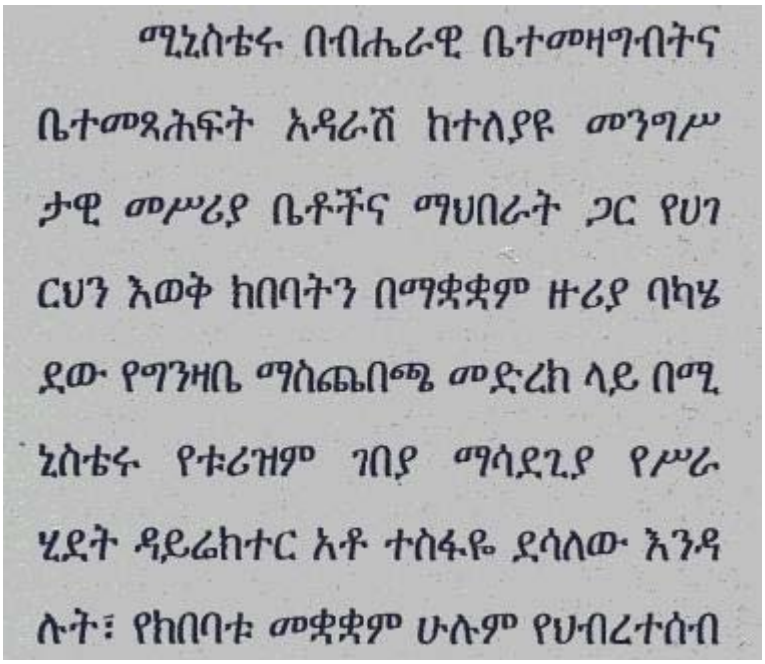
የደን ልማትና ጥበቃ ተግባርን በስፋት ማካሄድ ለሀገር
ኢኮኖሚ ግንባታና ለጎብረተሰቡ ፍላጎቶች መሟላት ከፍተኛ
አስተዋጽኦ ያለው በመሆኑ ፤

Train Set 8. Text from “Federal Negarit Gazeta”

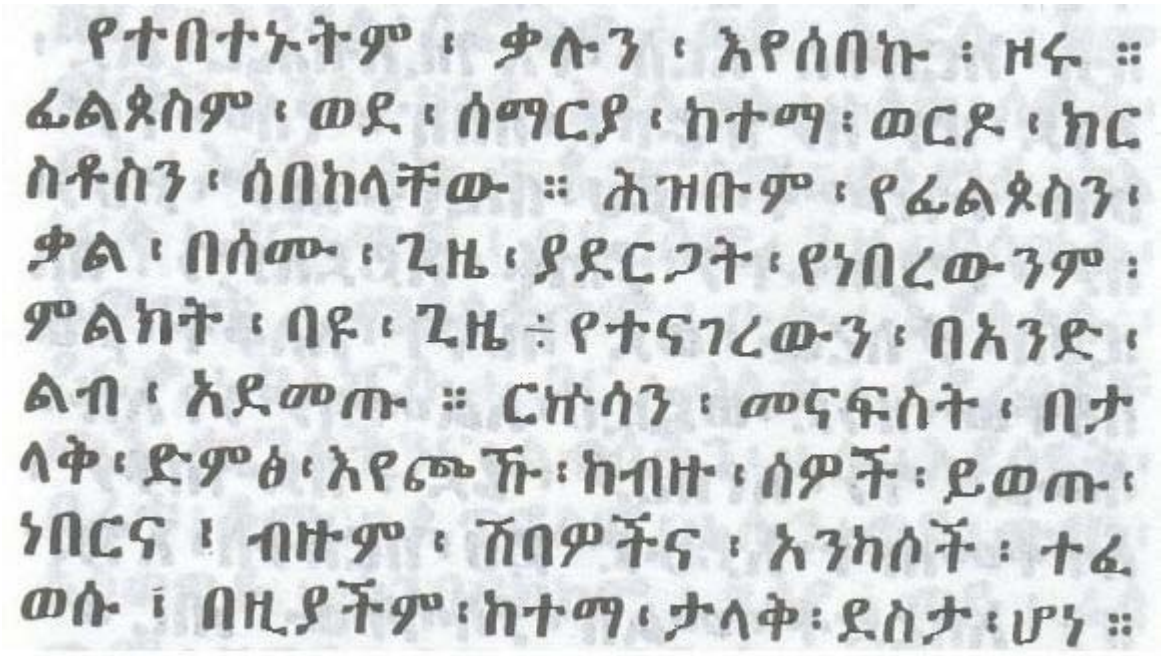
Annex- III. The Amharic Texts Included in the Test Set


አንበሳን ፡ ሰደበ ፡ ከፍ ፡ አርጎ ፡ ድምጹን ፡
የተኛው ፡ አንበሳ ፡ እድሜ ፡ ተጭኖት ፡
የነብርን ፡ ስድብ ፡ ሰማና ፡ ድንገት ፡
አጉራርቶ ፡ ቢነሳ ፡ ወኔው ፡ ቀስቅሶት ፡
ነብር ፡ ግራ ፡ ገባው ፡ መሄጃ ፡ ጠፋው ፡
መሬት ፡ አፍዋን ፡ ከፍታ ፡ ድንገት ፡ አትውጠው ፡
ሰማይ ፡ እንዳይወጣ ፡ ከንፍ ፡ አልነበረው ፡
ልቡን ፡ እፍረት ፡ ሞልቶት ፡ እየተጨነቀ ፡

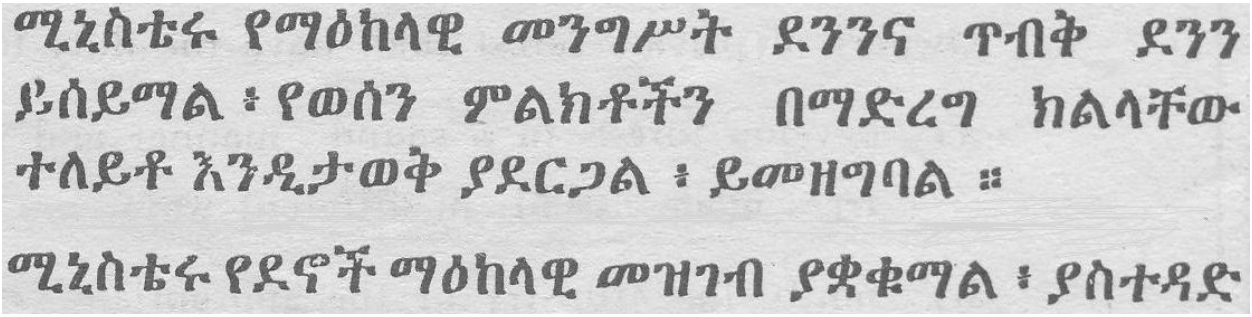
Test Set 1. Text from “Fikir Eskemekabir”


ሚኒስቴሩ በብሔራዊ ቤተመዛግብትና
ቤተመጻሕፍት አዳራሽ ከተለያዩ መንግሥ
ታዊ መሥሪያ ቤቶችና ማህበራት ጋር የሀገ
ርህን እወቅ ክበባትን በማቋቋም ዙሪያ ባካሄ
ደው የግንዛቤ ማስጨበጫ መድረክ ላይ በሚ
ኒስቴሩ የቱሪዝም ገበያ ማሳደጊያ የሥራ
ሂደት ዳይሬክተር አቶ ተስፋዬ ደሳለው እንዳ
ሉት፣ የክበባቱ መቋቋም ሁሉም የህብረተሰብ

Test Set 2. Text from “Addis Zemen”


የተበተኑትም ፣ ቃሉን ፣ እየሰበኩ ፡ ዞሩ ።
ፊልጶስም ፣ ወደ ፣ ሰማርያ ፣ ከተማ ፡ ወርዶ ፣ ክር
ስቶስን ፣ ሰበከላቸው ። ሕዝቡም ፡ የፊልጶስን ፡
ቃል ፣ በሰሙ ፣ ጊዜ ፣ ያደርጋት ፡ የነበረውንም ፡
ምልክት ፣ ባዩ ፣ ጊዜ ÷ የተናገረውን ፣ በአንድ ፣
ልብ ፣ አደመጡ ። ርኩሳን ፡ መናፍስት ፣ በታ
ላቅ ፡ ድምፅ ፡ እየጮኹ ፡ ከብዙ ፡ ሰዎች ፡ ይወጡ ፡
ነበርና ፤ ብዙም ፡ ሽባዎችና ፡ አንካሶች ፡ ተፈ
ወሱ ፤ በዚያችም ፡ ከተማ ፡ ታላቅ ፡ ደስታ ፡ ሆነ ።

Test Set 3. Text from " Bible"


ሚኒስቴሩ የማዕከላዊ መንግሥት ደንንና ጥብቅ ደንን
ይሰይማል ፥ የወሰን ምልክቶችን በማድረግ ክልላቸው
ተለይቶ እንዲታወቅ ያደርጋል ፥ ይመዘግባል ።

ሚኒስቴሩ የደኖች ማዕከላዊ መዝገብ ያቋቁማል ፥ ያስተዳድ

Test Set 4. Text from "Federal Negarit Gazeta"

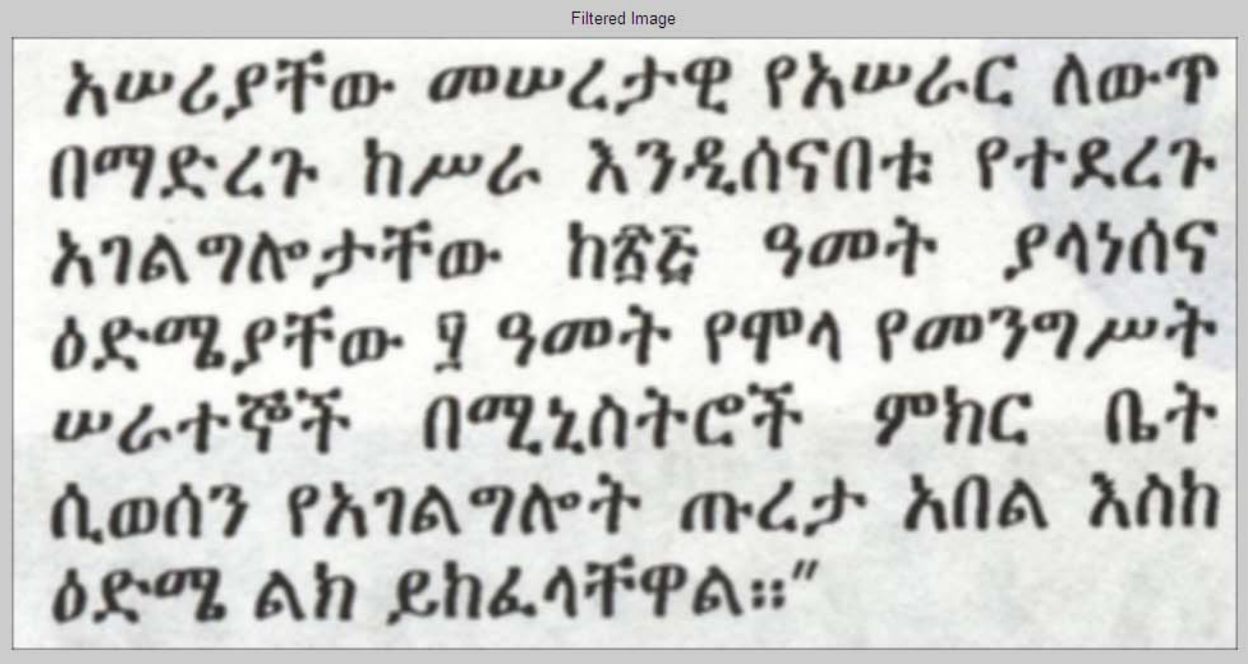
Annex- IV. Noise Removal Implementation

A. Linear Filtering

Linear filtering of images, either by correlation or convolution, can be performed using the toolbox function ***imfilter*** in the MATLAB. The image which was used for median and adaptive filtering in the main part of the thesis work (image from Negarit Gazeta) will be demonstrated here to compare results among filtering methods. It uses a 5-by-5 filter containing equal weights. The script that does linear filtering is:

```
Originalimg = imread('NegaritImg.jpg');
OriginalOnesimg = ones(5,5) / 25;
LinearFilteredimg = imfilter(Originalimg,OriginalOnesimg);
imshow(Originalimg), title('Original Image');
figure, imshow(LinearFilteredimg), title('Linear Filtered Image')
```

The output of linear filtering is shown below.

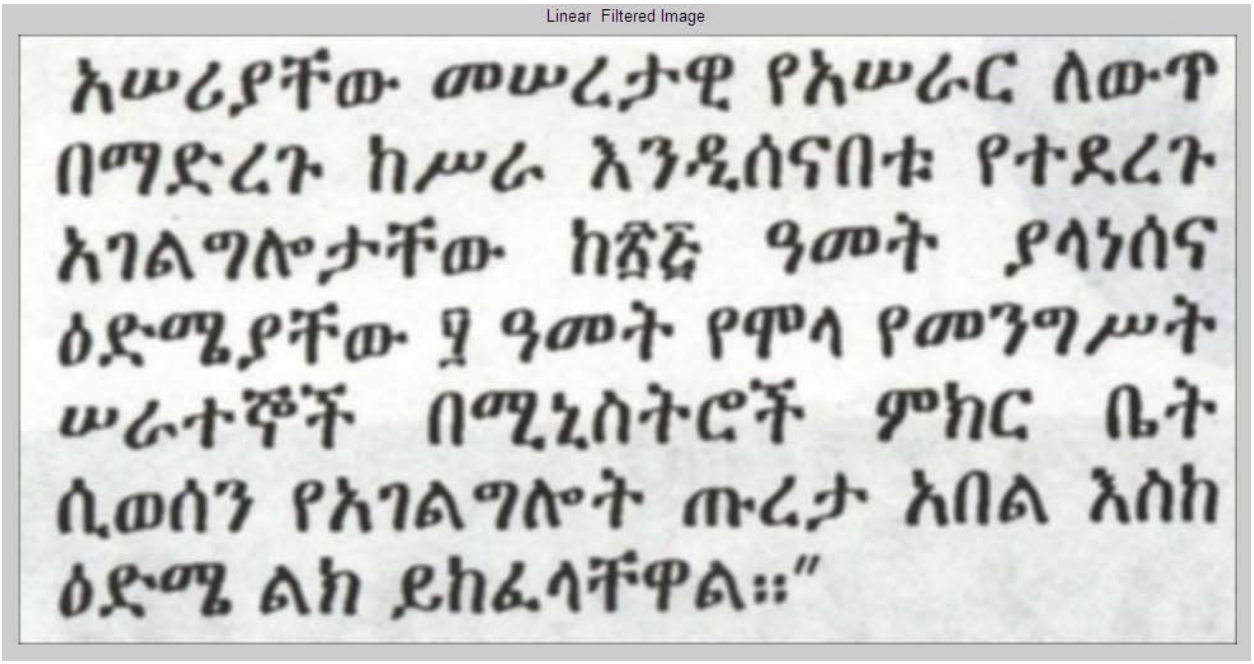


B. Median Filtering

The same picture used in the linear filtering will be used for the seek of comparing using a *medfilt2,* MATLAB function which performs median filtering to remove salt and pepper noise. In this case the size of the neighborhood used for filtering is 3-by-3. The script for median filtering is listed below.

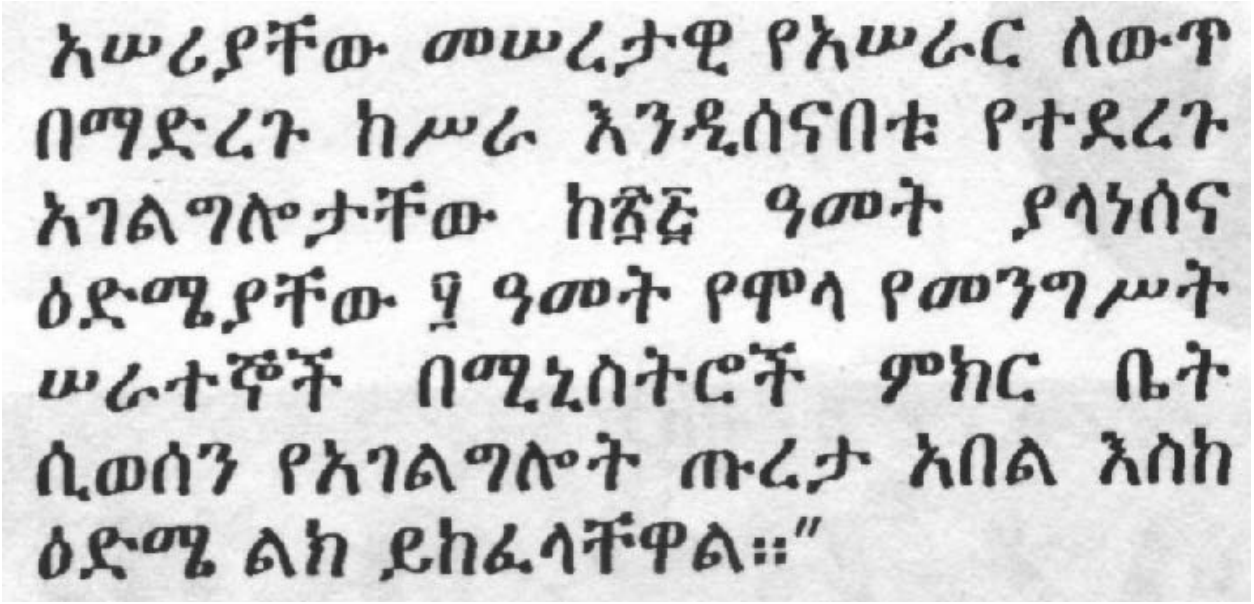
```
Originalimg = imread('NegaritImg.jpg');
Noisedimg = imnoise(Originalimg,'salt & pepper',0.02);
Medianfilteredimg = medfilt2(Noisedimg,[3 3]);
imshow(Originalimg), title('Original Image');
figure, imshow(Noisedimg), title('Original Image with Noise ');
figure, imshow(Medianfilteredimg), title('Median Filtered Image');
```

The output of linear filtering is shown below.



C. Adaptive Filtering

The above picture is passed through adaptive filtering module and produced the following adaptive filtered image. The script for adaptive filtering is discussed in section 4.3.2 under ‘C’ part.

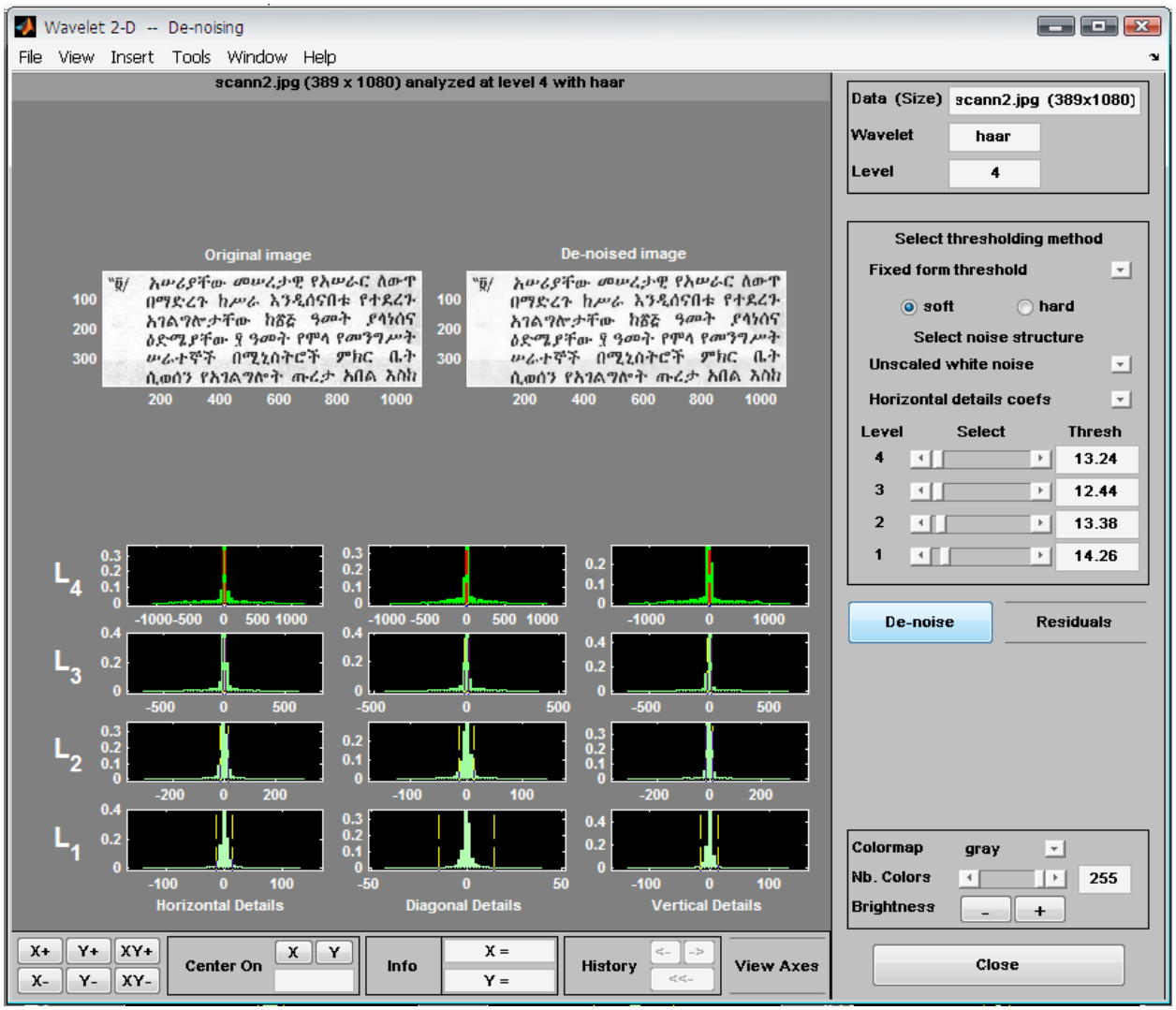


Annex- V. Two-Dimensional Variance Adaptive Thresholding of Wavelet Coefficients Binarization Implementation

The graphical interface tool throughout the MATLAB Wavelet Toolbox allows to define, level by level, time-dependent (x-axis-dependent) thresholds, and then increase the capability of the de-noising strategies handling nonstationary variance noise. More precisely, the model assumes that the observation is equal to the interesting signal superimposed on noise. The noise variance can vary with time. There are several different variance values on several time intervals. The values as well as the intervals are unknown. This thesis will use one of the graphical interface tool (SWT De-noising 2-D) to illustrate this capability. Procedures are reported below.

1. From the MATLAB prompt, type wavemenu, The Wavelet Toolbox Main Menu appears.
2. Click the SWT De-noising 1-D menu item in the Wavelet Toolbox Main Menu.
3. Load data (Negaritimg.jpg).
4. Perform data decomposition.
5. Generate interval-dependent thresholds.
6. De-noise with interval-dependent thresholds.

The output of the method is shown below.


Wavelet 2-D -- De-noising
File View Insert Tools Window Help
scann2.jpg (389 x 1080) analyzed at level 4 with haar
Original image
De-noised image
Data (Size) scann2.jpg (389x1080)
Wavelet haar
Level 4
Select thresholding method
Fixed form threshold
soft
hard
Select noise structure
Unscaled white noise
Horizontal details coefs
Level Select Thresh
4 13.24
3 12.44
2 13.38
1 14.26
De-noise
Residuals
Colormap gray
Nb. Colors 255
Brightness
Horizontal Details
Diagonal Details
Vertical Details
X+ Y+ XY+ X- Y- XY-
Center On X Y
Info X = Y =
History
View Axes
Close

Annex- VI. Source Code for Line Segmentation

```
% segline.m
% Basic function
% segments lines of text image from a scanned text image
% description
% This program should be given a binary preprocessed image "Removedimg".
% the program first gets a column vector of sum of pixel brightness values (0 or 1) for each row.
% Then the row having maximum sum is considered white space between text lines.
% The coordinates of each text  line are then saved. Finally, each text line image is saved in a
% cell array.


% Determine number of lines on the scanned image.
RemovedimgSum=sum(Removedimg,2); %  sums along the dimension 2 of the given image.
[m,n]=size(Removedimg);   %  returns the number of rows and columns of the given image in
separate output variables m and n.
RemovedimgMaxSum = max(RemovedimgSum);  %  returns a value in a row vector of the
given image containing the maximum element from each column.
RemovedimgMaxSumVector=[ ];   % RemovedimgMaxSum vector

% identify lines with text
k=1;
for i=1:m,
        if RemovedimgSum(i)~= RemovedimgMaxSum
                RemovedimgMaxSumVector=[RemovedimgMaxSumVector;i];
   end
end
% save line structure line structure "LineStructure"
RemovedimgLineimgVector=[];
l=1;
LineStructure=struct('lineno',{0},'RemovedimgLineimgVector',RemovedimgLineimgVector);
LineStructure(1).lineno=1;
[r,c]=size(RemovedimgMaxSumVector);  % to determine # of rows with text in
RemovedimgMaxSum
for i=1:r,
        if i==r & RemovedimgMaxSumVector(i)-RemovedimgMaxSumVector(i-1)==1
                RemovedimgLineimgVector
=[RemovedimgLineimgVector;Removedimg(RemovedimgMaxSumVector(i),:)];
      LineStructure(l).RemovedimgLineimgVector=RemovedimgLineimgVector ;
      l=l+1 ;
                LineStructure(l).lineno=l;
        elseif RemovedimgMaxSumVector(i+1)-RemovedimgMaxSumVector(i)==1; % row
sequential pixels
```

```
RemovedimgLineimgVector=[RemovedimgLineimgVector;Removedimg(RemovedimgMaxSum
Vector(i),:)];
   else

RemovedimgLineimgVector=[RemovedimgLineimgVector;Removedimg(RemovedimgMaxSum
Vector(i),:)] ;  % take into account the last pixelline
                    LineStructure(l).RemovedimgLineimgVector=RemovedimgLineimgVector;
      %  figure,imshow(RemovedimgLineimgVector)
      %  title('IMAGES WITH LINE SEGEMENTATION ')
      %  pause (0.5)
      l=l+1;
                    LineStructure(l).lineno=l;
                    RemovedimgLineimgVector=[ ];
   end
end
% figure, imshow(RemovedimgLineimgVector)
% title('IMAGES WITH LINE SEGEMENTATION')
% pause(0.5)
ldl=LineStructure(l).lineno;
ld=double(ldl);
```

Annex-VII. Source code for Character Segmentation, Normalization and Thinning

```
%%%%%%%%%%%%%%%%%% charcter segementation, thinning and size normalization
%%%%%%%%%%%%

% initialize variables
cnt=0 ;     % total character count
ChrVector=[ ];  % character vector
ConcChrVector=[];       % Concatenated character vectors
ChrOutputVec=[ ];     % corresponding characters output vector

% start segmenting characters from segmented lines in 'segline.m'above
for TextLine=1:l,  % 1 = # of text lines in image
        xx=LineStructure(TextLine).RemovedimgLineimgVector;
        [lc4,num]=bwlabel(~xx);         % identify/label each connected chr in a line
   stats=regionprops(lc4,'all');   % get feature including bounding box image
   % Plot Bounding Box
for n=1:size(stats,1)
   rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)
   end
% get and scale each character
for cc=1:num,
        ChrInBox=stats(cc).Image ; % character image in an enclosing box
   ThinnedimgInBox=bwmorph(ChrInBox,'thin',Inf); % thin charcter images in an inclosing box
   imshow(~ThinnedimgInBox);
   xc=double(xc);
   % begin resizing
        ResizedimgInBox=imresize(ChrInBox,[20,20],'bilinear'); % 16*16 resized image in an
enclosing box
   figure(2);
   imshow(~ResizedimgInBox);
   pause(0.5);
   begin thinning
   ThinnedimgInBox=bwmorph(ResizedimgInBox,'thin',Inf); % thin charcter images in an
inclosing box
   figure(6);
   imshow(~ThinnedimgInBox);
   pause(1);
   ResizedFidel= ThinnedimgInBox(:,1);
for t=2:16
      ResizedFidel=vertcat( ResizedFidel,ResizedimgInBox(:,t));
   end
   ChrVector=[ChrVector;ChrInBox(:)] ; % store each character in a line for writing data to file
        ConcChrVector=horzcat(ConcChrVector, ResizedFidel);
```

```
    cnt=cnt+1;
end
end
% write training pair data to a file

 inputtobewritten=double(ConcChrVector);
 xlswrite('NegaritInput.xls',inputtobewritten);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
```

Annex- VII. MATLAB code for creating and training and Testing the Network

```
% This function will take the binary representation of the four training
% character sets with either 400x with the total number of characters in the training.
% The code first train the network with the original training sets, tests with the training set and
% test with new test sets.

% load inputs
input1=xlsread('FikirEskemekabirInput.xls'); %FikirEskemekabirOutput.xls
input2=xlsread('AddisZemenInput.xls'); %AddisZemenOutput.xls
input3=xlsread('BibleInput.xls'); %BibleOutput.xls
input4=xlsread('NegaritInput.xls'); %NegaritOutput.xls
input5=xlsread('TestBibleInput.xls'); %TestBibleOutput.xls
input6=xlsread('TestFikirEskemekabirInput.xls'); %TestFikirEskemekabirOutput.xls
input7=xlsread('TestAddisZemenInput.xls'); %TestAddisZemenOutput.xls
input8=xlsread('Img1Input.xls'); %Img1output.xls
TrainSet= [input1,input2,input3,input4,input5,input6,input7,input8];

% find inputs
input_range1=minmax(input1);
input_range2=minmax(input2);
input_range3=minmax(input3);
input_range4=minmax(input4);
input_range5=minmax(input5);
input_range6=minmax(input6);
input_range7=minmax(input7);
input_range8=minmax(input8);
input_range_all=[ input_range1, input_range2,
input_range3,input_range2,input_range5,input_range6,input_range7,input_range8 ];

% Network with 40 hidden and 1 outpul Layer is created with activation function
% 'logsig'and'purelin'

Network=newff(input_range1,[40 1],{'tansig','purelin'},'traingdx');

Time_Start=clock;
Network.performFcn='mse';
Network.trainparam.lr=0.01;
Network.trainparam.goal=0.000001;
Network.trainparam.show=50;
Network.trainparam.epochs=1000;
Network.trainparam.mc=0.8;
Network.trainParam.mu_max = 1e10;
```

```
Network.trainParam.mu=0.05;
Network.trainParam.mu_dec=0.01;
Network.trainparam.min_grad=1.0000e-006;

target1=xlsread('FikirEskemekabirOutput.xls');
target2=xlsread('AddisZemenOutput.xls');
target3=xlsread('BibleOutput.xls');
target4=xlsread('NegaritOutput.xls');
target5=xlsread('TestBibleOutput.xls');
target6=xlsread('TestFikirEskemekabirOutput.xls');
target7=xlsread('TestAddisZemenOutput.xls');
target8=xlsread('Img1output.xls');

target1=target1';
target2=target2';
target3=target3';
target4=target4';
target5=target5';
target6=target6';
target7=target7';
target8=target8';

TargetSet=[target1,target2,target3,target4,target5,target6,target7,target8];

%Training the network with the original training charcater sets
[Network,tr]=train(Network,TrainSet,TargetSet);

% Testing with the new data for Bible
TestBibleimg=xlsread('Test2_BibleInput.xls');
TestSetBibleimg=[TestBibleimg];
TestRst1=sim(Network,TestSetBibleimg);

% Testing with the new data for Fikir Eskemekabir
TestFikirEskemekabirimg=xlsread('Test3_FikirInput.xls');
TestSetFikirEskemekabirimg=[TestFikirEskemekabirimg];
TestRst2=sim(Network,TestSetFikirEskemekabirimg);

% Testing with the new data for Addis Zemen
TestAddisZemenimg=xlsread('Test4_AddisInput.xls');
TestSetAddisZemenimg=[TestAddisZemenimg];
TestRst3=sim(Network,TestSetAddisZemenimg);

% Testing with the new data for Negarit
TestNegaritimg=xlsread('Test1_NegaritInput.xls');
TestSetNegaritimg=[TestNegaritimg];
TestRst4=sim(Network,TestSetNegaritimg);
```

DECLARATION

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

_____

Abay Teshager Birhanu

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

_____

Wondwossen Mulugeta (Ato)

July 2010, Addis Ababa, Ethiopia.