

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

OCR FOR SPECIAL TYPE OF HANDWRITTEN AMHARIC TEXT
(“YEKUM TSIFET”)

NEURAL NETWORK APPROACH

By

WONDWOSSEN MULUGETA GEWE

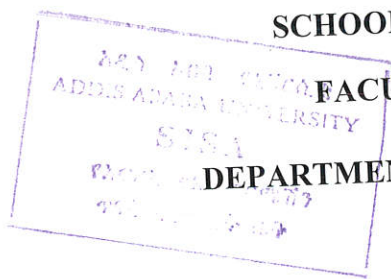
A thesis submitted to the School of Graduate Studies of Addis Ababa University

In partial fulfillment of the requirements for the Degree of

Master of Science in Information Science

June 2004

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES



FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

OCR FOR SPECIAL TYPE OF HANDWRITTEN AMHARIC TEXT ("YEKUM TSIFET")

NEURAL NETWORK APPROACH

By

WONDWOSSEN MULUGETA GEWE

Signature of the Board of Examiners for Approval

Dr. Gashaw Kebede
Chairman, Examining Board

Dr. Atul Negi
Advisor

Ato Nigussie Tadesse
Advisor

Dr. N. Kaiv Murthy
External Examiner

ACKNOWLEDGMENT

First of all I would like to praise God who has been there with me all the time, even at the lowest moment of my life. Next, I would like to thank my advisors Dr. Atul Negi and Ato Nigussie Tadesse for their constructive comments and encouragement on my work. This work would not be possible without the unreserved and careful technical assistance of my advisors.

I would also like to thank my parents Ato Mulugeta Gewe and W/o Fetlework Bekele and also my sister Haregewoin Mulugeta and brother Mesfine Mulugeta for their wholehearted financial and moral support throughout my stay at the university.

The technical and related material support I get from Dr. Getu Hailu, Dr. Tewodros Arage, and Ato Menasse Zewdu helped me a lot for the completion of this work. I owe them a lot.

How could I forget the motive and the encouragement I get hold from Ms Meseret Degefu to join the school and during the two years study? I'm also indebted to my classmates who have been there with me all the time when I needed them.

I am greatly indebted to Ms Frehiwot Getahun for editing and commenting the original scripts.

Thank You All

Wondwosesen Mulugeta

DEDICATION

To My Whole Family

And

Meseret Degefu

Work

Table of Contents

ABBREVIATIONS.....	v
LIST OF FIGURES.....	vi
LIST OF TABLES.....	vii
ABSTRACT.....	viii
CHAPTER ONE.....	1
INTRODUCTION.....	1
1.1 Background.....	1
1.2 Why Amharic OCR System?	3
1.3 Historical development	4
1.4 Statement of the Problem.....	5
1.5 Justification of the study	7
1.6 Objective of the study.....	10
1.7 Scope of the Study.....	11
1.8 Methods of the Study.....	11
1.9 Application of results	13
1.10 Organization of the study.....	14
CHAPTER TWO.....	16
REVIEW OF LITERATURE ON OCR SYSTEMS	16
2.1 Introduction	16
2.2 Preprocessing Methods	16
2.2.1 Mandatory Preprocessing Steps	18
2.2.1.1 Digitization	18
2.2.1.2 Image Binarization/Thresholding.....	19
2.2.1.3 Segmentation.....	22
2.2.2 Optional Preprocessing Steps.....	28
2.2.2.1 Thinning.....	28
2.2.2.2 Size Normalization.....	29
2.3 Techniques in Pattern Recognition.....	30
2.4 Artificial Neural Network for Character Recognition.....	33
2.4.1 The Natural/Biological Neural Network	34
2.4.2 The Artificial Neural Network	35
2.4.2.1 Design of an Artificial Neural Network.....	36
2.4.2.2 Layers of an Artificial Neural Network.....	37
2.4.2.3 Artificial Neural Network Architecture	39
2.4.2.4 Communication and Types of Connections	40
2.4.2.5 Training the Artificial Neural Network.....	42
2.4.2.6 Neural Network Learning Rules	43
2.4.2.7 Learning Rates.....	46
2.4.2.8 Learning laws	47
2.4.2.9 Multi Layer Perceptron (MLP).....	47
2.5 Attempts on Amharic OCR.....	51

CHAPTER THREE	57
THE AMHARIC WRITING SYSTEM	57
3.1 <i>Introduction</i>	57
3.2 <i>The Amharic writing system</i>	57
3.3 <i>Amharic Handwriting</i>	60
3.4 <i>Writing style of "Yekum Tsifet"</i>	61
CHAPTER FOUR	66
EXPERIMENTATION	66
4.1 <i>Introduction</i>	66
4.2 <i>Data Compilation</i>	66
4.3 <i>Digitization</i>	70
4.4 <i>Binarization</i>	71
4.5 <i>Image Segmentation.....</i>	73
4.6 <i>Character Normalization.....</i>	80
4.7 <i>Creating Character Representation</i>	83
4.8 <i>Character Recognition.....</i>	86
4.8.1 <i>Artificial Neural Network Architecture</i>	87
4.8.2 <i>The Neural Network</i>	88
4.8.3 <i>Character Input Method for the Neural Network.....</i>	89
4.8.4 <i>Creating the Neural Network</i>	90
4.8.5 <i>Training the Neural Network</i>	92
4.8.6 <i>Testing the Neural Network.....</i>	96
4.9 <i>Conclusion</i>	100
CHAPTER FIVE	101
CONCLUSION AND RECOMMENDATION	101
5.1 <i>Introduction</i>	101
5.2 <i>Conclusion</i>	101
5.3 <i>Recommendation.....</i>	104
REFERENCES	106
APPENDIXES	109
<i>Annex-1: Training set one.....</i>	109
<i>Annex-2: Training set two.....</i>	110
<i>Annex-3: Training set three</i>	111
<i>Annex-4: Test Case one</i>	112
<i>Annex-5: Test Case two</i>	113
<i>Annex-6: The Amharic Character Set.....</i>	114
<i>Annex-7: Sample Randomly Generated Noisy Character</i>	115
<i>Annex-8: MATLAB code for creating and training the network</i>	116
DECLARATION	117

LIST OF TABLES

Table 1: Layers and their Activities.....	38
Table 2: Summary of Amharic OCR Attempts.....	56
Table 3: Network Performance for the Training Set	97
Table 4: Network Performance for Training Set with Noise.....	98
Table 5: Network Performance for New Test Set.....	99

ABSTRACT

Verbal and written communications, which are integral components of human society, have been transformed by the development of the respective communication devices. Through the swift development in processing devices, the need and access to digitize printed information items by means of Optical Character Recognition (OCR) became possible. Despite the fact that most world languages are beneficiaries of this technology, the application of character recognition technology to Amharic text is a recent experience, and in its infancy stage when handwritten recognition is considered.

This study is then an attempt made to develop a recognition engine for Amharic handwritten text written in a special type of writing style, which is called "Yekum Tsihuf" ("ሃቂግ ጽሁፍ"). Before mechanical and electronic text processors were introduced in Ethiopia, information used to be recorded on natural materials by hand writing, animal skin being the dominant one. Those handwritten documents, written in this writing style, hold vital information about history, tradition, religion, nature and etc., which render undeniable contribution to current and future studies. The availability of this information in an electronic form would greatly help preservation and communication.

In this study, the application of handwritten character recognition with Artificial Neural Network implementation for the 231 main character set of Amharic language is attempted. The training and test data sets are produced by scribes who are trained to write text using the writing style. The study used various techniques at each phase from digitization to recognition levels. Preprocessing methods like image binarization, character segmentation, and size normalization and neural network recognition is made using Visual C++.Net and MATLAB programming environments. While segmentation rate of 95.96% is attained using stage-by-stage segmentation algorithm, recognition rate that ranges from 98.8% to 20.3% is obtained for different test cases. Based on the findings and the knowledge acquired during the experimentation, topics for future research are also identified.

CHAPTER ONE

INTRODUCTION

1.1 Background

Verbal and written communications, which are integral components of human society, has been transformed by the development of the respective communication devices. Through the swift development in processing devices, the need and access to digitize printed information items by means of Optical Character Recognition (OCR) became possible.

In the forthcoming modern society as well as in the existing era, information, which is capable of being processed electronically, is critical. The concern of the information society is not just the storage of information with the representation of bits and bytes. The apprehension is the ability of those electronic representations being processed by means of intelligent applications like word processing, spreadsheet, and information retrieval systems.

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). OCR is the process of examining printed characters on paper and determining their shapes by detecting patterns of dark and light. Once the scanner or reader has determined the shapes, character recognition methods and pattern matching with stored sets of characters are used to translate the shapes into

computer text [Encarta, 2001]. Such systems have the capability to convert the bit wise representation of scanned document into its equivalent computer-readable and computable file formats.

In character recognition systems, an OCR engine takes data, which is either handwritten or typewritten text to convert it into computer readable format [Pal and Chaudhuri, 1995]. And handwritten format is the very natural means of human communication next to speaking. In the case of printed characters, shapes vary in stroke width, ornament, style, and the like. In the case of handwritten characters, there are many more nonlinear transformations [Mori, 1999], which make the recognition as well as character segmentation task even harder.

There are two types of OCR systems with respect to the way the input is provided to the system. The first being on-line OCR where the recognition task is performed concurrently with the writing process [Eric, 1993]. This type of recognition has the advantage of capturing the temporal or dynamic information of the character in case of handwriting. The information consists of the number of strokes, the order of the strokes, and the speed of the writing within each stroke. [Jun Fan, 2003]

The other type of OCR system is off-line OCR where the recognition process is performed after the whole text, which is a scanned bit map representation of a text, is supplied to the system [Eric, 1993]. When contrasted with the on-line recognition, the off-line case is much more difficult because it lacks all the knowledge acquired in the case of on-line recognition [Jun Fan, 2003]. In on-line OCR there is a need to use pen-based interface, while in off-line OCR equipments

like scanner and preprocessing algorithms like segmentation and noise removal are required.

At each era of its development, OCR technology frees acres of storage space once given over to file cabinets and boxes full of paper documents. Today, OCR software can recognize a wide variety of fonts, but handwritten and script fonts that mimic handwriting are still problematic [Eric, 1993]. This is due to the reason that handprint recognitions should be capable of taking care of distortions of characters. The distortions of characters are evident with the writing of a single individual let alone between different writers.

1.2 Why Amharic OCR System?

Before and after the emergence of computer use for office and personal activities in Ethiopia, information is recorded using various types of materials available for writing by means of handwriting and typewriter. As the working language of the Federal Government and some regional governments in Ethiopia is Amharic, most documents are written in Amharic characters, which are unique to the country. In addition, the majority of other local languages use Amharic characters for writing purpose [Berder et al, 1976]. These printed and handwritten documents hold vital information about history, tradition, religion, nature and etc., which render undeniable contribution to current and future studies. The availability of these information first electronically and then at character level will fill the information access gap.

Before typewriters were introduced to Ethiopia, information used to be recorded by specialized writing style. The experts who do the lettering are called scribes and are trained for many years. There is a huge collection of manuscripts written with this writing style found in IES (Institute of Ethiopian Studies) and churches. These manuscripts are not only spiritual and religious in content but also render various types of information at their time of scribing. Thus making available this information on computer and providing the facility of editing and indexing will help the information storage, retrieval, indexing and searching to be at the tip of one's hand, permitting enormous information input to users by revealing the content of the scripts.

As a result, this thesis aims at developing an OCR engine, which converts a special type of handwritten Amharic text into a standard computer editable format. The typestyle is traditionally called "Yekum Tsehuf" ("የቁጥ አሁፍ") which was used to preserve knowledge and events on animal skin before the introduction of typewriters and computers to Ethiopia.

1.3 Historical development

OCR technology, as presented by Gray [1977], was introduced to the world of science in the early 1800s. Gray also noted that the practically applicable OCR system was invented in 1950's to assist character reading as a replacement for keypunching systems. From then onwards, OCR imposes a huge influence around the business world and governments towards electronic data processing. According to Srihari [1996], the modern OCR technology is, said to have been,

born in 1951 with the invention of GISMO, a Robot Reader Writer. As reported by Blumenstein & Kulkarni, J. Rainbow developed a machine capable of reading uppercase typewritten letters. And the prototype is claimed to do the task in a very sallow manner. By 1967, companies such as IBM finally marketed OCR systems [Blumenstein & Kulkarni 1998].

At each stage of its development, OCR system developers and researchers try to encapsulate the-at-then modern technology to make the systems accurate and efficient. Researchers in the area of pattern recognition believe that OCR problems can be decomposed into levels, and each will have a contribution to the field when solved accordingly [Mori, 1999]. Thus, the scanning hardware, the preprocessing algorithms and techniques, character segmentation, recognition and classification methods are being modified at different instances. These modifications and new inventions make the OCR technology to attain a high level of accuracy and efficiency in terms of recognition time. All these indicate that OCR has grown as an industry, aided by the rapid development of computer technology [Mori, 1999].

1.4 Statement of the Problem

Amharic language writing system is composed of a total of more than 330 characters. The basic writing character set is composed of 33 symbols that are core characters. Each of these core characters occur in seven orders; one basic form and six non-basic forms representing syllable combinations consisting of a consonant and vowel. In addition to the 231-character set, which are the 33 core and its 7 forms for each, there are additional symbols to represent numbers, punctuations

and other rarely used characters, which make up a number of characters in Amharic writing system to be more than 330. [Bender et al, 1976].

The application of OCR technology for Amharic characters is a recent research focus. Even though investigations have started, there is no complete and full-fledged system to perform the image-to-text conversion. Literature shows that languages in Europe, Far East Asia and America developed such systems to facilitate knowledge transfer. These languages include Latin, Japanese, Chinese, Deranagari, Bengali, Arabic, Swedish, Russian, Tibetan, and the likes.

In case of Amharic OCR development, postgraduate students from Addis Ababa University have conducted researches on this area. The work of Worku [1997], Ermias [1998], Dereje [1999], Berhanu [1999], Nigussie [2000], Million [2000], Yaregal [2002], and Mesay [2003] are those to mention. These researchers tried to solve some problems of the Amharic OCR development. The common ball for all these works, except that of Nigussie and Mesay, is that, all based their investigation on printed, either by typewriter or printer, Amharic text.

To the best of the researchers knowledge, only Nigussie and Mesay, attempted handwritten OCR attempt and their work was on the application of processing bank checks and postal address processing respectively [Nigussie, 2000, Mesay, 2003]. Thus, in this thesis, the OCR technology will be investigated more for a special type of handwritten Amharic text using all the 231 basic character sets. The writing style under consideration was used to record historical events, social development, religious and cultural activities and beliefs in early times.

and other rarely used characters, which make up a number of characters in Amharic writing system to be more than 330. [Bender et al, 1976].

The application of OCR technology for Amharic characters is a recent research focus. Even though investigations have started, there is no complete and full-fledged system to perform the image-to-text conversion. Literature shows that languages in Europe, Far East Asia and America developed such systems to facilitate knowledge transfer. These languages include Latin, Japanese, Chinese, Deranagari, Bengali, Arabic, Swedish, Russian, Tibetan, and the likes.

In case of Amharic OCR development, postgraduate students from Addis Ababa University have conducted researches on this area. The work of Worku [1997], Ermias [1998], Dereje [1999], Berhanu [1999], Nigussie [2000], Million [2000], Yaregal [2002], and Mesay [2003] are those to mention. These researchers tried to solve some problems of the Amharic OCR development. The common ball for all these works, except that of Nigussie and Mesay, is that, all based their investigation on printed, either by typewriter or printer, Amharic text.

To the best of the researchers knowledge, only Nigussie and Mesay, attempted handwritten OCR attempt and their work was on the application of processing bank checks and postal address processing respectively [Nigussie, 2000, Mesay, 2003]. Thus, in this thesis, the OCR technology will be investigated more for a special type of handwritten Amharic text using all the 231 basic character sets. The writing style under consideration was used to record historical events, social development, religious and cultural activities and beliefs in early times.

Those documents written using the writing style render vital information which will make different parties to be beneficiary for research and knowledge acquisition. But this is not possible unless the content is found in computer editable format and suitable for searching and retrieval task. For this, the first step to be accomplished is optically recognizing each character and convert it to its equivalent electronic representation.

1.5 Justification of the study

The benefit offered by computable electronic information is not controversial. The potential of OCR systems is enormous because they enable users to harness the power of computer to access and manipulate printed documents. OCR has unquestionable benefit in legal profession where storage, retrieval and searching are frequent tasks and the target information is in hardcopy level. OCR can be integrated with text retrieval systems to makes possible searches that once require hours or days to be accomplished in a few seconds. [Hammerstrom, 1993]

OCR is important in the information era since it is still true that printed documents are dominant and the information contained in them is more useful in electronic format. Haigh [1996] noted that, the problem with scanned images or texts is that processing at that level is impossible. A document image can be displayed onscreen or printed to the computer, however, the image is not “readable”, and it cannot be “understood” as anything other than collection of dots or raster image [Haigh, 1996] OCR will translate text in bitmap or other image formats into machine-readable form.

Therefore OCR technology should be considered if:

- *The text is to be reused, edited or reformulated,*
- *The text should be available for full-text information retrieval,*
- *The text should be available to adaptive equipments for the visually impaired,*
- *File size is of concern in terms of storage and bandwidth to transmit.*

Especially in Ethiopia, a country that has its own writing system and having Amharic as the working language of the federal government, and most local languages using the same character set, lots of documents are produced and stored in hardcopy form. The production of such handwritten documents was in a high rate before text processing devices are introduced in the country. The translation of these printed and handwritten documents into machine understandable format is vital to best exploit the information content.

The effort made by other researchers showed that there is a great potential for Amharic hardcopy documents to be electronically transformed into computer readable format. As there are huge collections of literature, which are typical handwritten texts before the coming of typewriter or printer to Ethiopia, developing an Amharic OCR for handwritten documents will lessen many of the problems related with the digitization of handwritten manuscripts.

Among these wide ranges of handwritten documents, those written and preserved by Ethiopian Orthodox Church, ancient historians, and around the royal family are

the major ones. Observing the collection of manuscripts scribed using this writing style can also support this fact. Revealing information content of the documents written “Yekum Tsifet” will help to have a good impact on social research and historical studies. Moreover indigenous knowledge preserved using the writing style will benefit the academic as well as the business sectors.

Thus making an attempt towards developing a recognition engine, which translates these bitmapped representations of handwritten documents into computer readable format, will be a new venture in Amharic handwritten OCR development. When analyzing the direction of the world of pattern recognition researches, in the last decades, handwriting recognition has been a very active area of research. This is due to the demand of the information society and the advances in automation technologies to bring most activities to an electronic paperless world and knowledge preservation.

1.6 Objective of the study

This thesis work has particular general objective and some set of specific goals which will be attained to accomplish the general objective.

General Objective

The general objective of this thesis is to develop OCR engine for special type of handwritten Amharic text, which is traditionally called "Yekum Tsehuf" "የቁጥጥ አሁፍ".

Specific Objectives

Towards achieving the above-mentioned general objective, the study accomplishes the following specific objectives.

- Review previous works on handwritten OCR system and Amharic character recognition.
- Investigate the special features of Amharic characters and more thoroughly of the writing style ("Yekum Tsehuf").
- Review the application of Neural Network approaches for character recognition purpose.
- Assess different preprocessing, recognition and classification algorithms for the purpose of handwritten character recognition.
- Developing required algorithms for the recognizer using Visual C++.Net and MATLAB programming environment.
- Test and evaluate the performance of the developed recognition engine.
- Recommend future research focus.

1.7 Scope of the Study

The study is mainly concerned with optical character recognition for Amharic characters, which are written with special kind of layout, traditionally called Yekum Tsehuf "የቁጥጥ አሁፍ". Thus on this thesis an attempt will only be made to adopt different OCR algorithms for such kind of writing style. In addition to handwriting type, the experimentation and character tracing will only be made on text written on white paper. The study will go no further than experimenting and testing the applicability of some preprocessing algorithms and artificial neural network recognition for the handwriting style described above.

As handwritten character recognition is claimed to be very difficult due to many nonlinear transformations observed in characters written by different writers, the search limit the experimentation to very few number of scribes.

1.8 Methods of the Study

Since the aim of the thesis is to adopt and/or devise an algorithm, which performs Amharic handwritten character recognition, various methods of research will be employed. To achieve the general and the specific objectives, the following research approaches will be used.

Literature Review

Past researches are of assistance to identify what has been done in the area and which approaches fit best for specific problem domain, previous works in Amharic

addition, OCR system for other languages handwritten text will be reviewed to put light on the way towards developing Amharic handwritten OCR development.

In addition to studies in Amharic and handwritten OCR systems, recent general character recognition findings for other character sets will be reviewed so as to have good background on the ways of handling handprint text processing problems and appropriate approaches to tackle the bottlenecks.

Development Techniques

Various algorithms in the previous Amharic OCR development have been done using Microsoft Visual C++ programming language, which lower the burden of coding for the researcher. Moreover, Microsoft Visual C++.Net programming language provides the facility of using classes, which enables the project to use objects, which again closely model the real world objects in attributes and behaviors.

Thus the programming language, which will be used to develop the preprocessing steps, will be Microsoft Visual C++.Net to create consistency between the previous works.

For the purpose of classification, Artificial Neural Network techniques is adopted, which is categorized as one of the robust approaches to deal with uncertainty and noisy input data. For this classification purpose, MATLAB Neural Network toolbox is used mainly due to its availability and ease of constructing and training

the network. In addition MATLAB is powerful in dealing with matrices which is the way characters are represented for recognition purpose.

Training and Testing

Along with the development task of the work, the recognition system will be trained with some set of the special handwritten Amharic characters written by the experts/scribers on white papers. This means that the training of the recognition engine is going to be done by tracings on white paper. After training the neural network, the network will be tested on different new inputs to see the recognition accuracy and effect some required modification on the network parameters to increase the generalization power of the network.

1.9 Application of results

OCR is an application, which converts bitmap representation of a scanned document into computer “readable” formats. This application will assist the conversion of hardcopy documents, or documents in printed format into electronically editable format. This will facilitate minimized storage, fast indexing, full text retrieval, and text modification. Since the purpose of OCR is to decrease the data input and storage bottleneck, it enables to automate data entry and processing by reducing human involvement in transcribing data into the computer. The development of computers and other technologies is, indeed, to hand down the burden of the humans.

As Green [1993] put it in detail, the primary benefits offered by an OCR system are improved accuracy of data conversion, increased efficiency in data storage,

data retrieval and processing. Thus the development of Amharic handwritten OCR will enable handwritten manuscripts be converted into editable format with improved accuracy and less storage requirement.

The converted documents will provide vital information for historians, politicians, social researchers, and many other professionals. This will increase the usefulness of current and future studies by revealing the exact happenings and developments during the past.

1.10 Organization of the study

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 analysis. The section also discusses different pattern recognition techniques generally and character recognition methods specifically.

The third chapter discusses about the nature and characteristics of the Amharic writing system along with the target writing style of this research, “Yekum Tsifet”. The chapter also reviews the different approaches used so far to develop an OCR system along with the results obtained for the Amharic characters for both printed and handwritten text.

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.

The last chapter will summarize the over all task performed with the techniques used and the results achieved by the implemented techniques. The chapter also forwards a set of recommendations for further consideration and research guide.

CHAPTER TWO

REVIEW OF LITERATURE ON OCR SYSTEMS

2.1 Introduction

This part will be devoted to give a detailed description of the preprocessing steps in most OCR systems and the widely used algorithms adopted at different instances aiming at enhancing a recognition system.

Automatic recognition of printed characters through standard Optical Character Recognition (OCR) systems cannot be successfully applied to ancient texts, where aging of paper, diffusion of ink and other processes have strongly degraded the quality of the documents. Indeed, due to non-uniform reduction of the image contrast, non-uniform spacing of the characters, presence of broken, touching and merging characters, and strong background noise, the preliminary step of character segmentation is particularly difficult. Thus when dealing with historical documents, the attention given to preprocessing steps should be high.

In the coming sections, the preprocessing algorithms that should be implemented in all OCR systems and those optional ones will be discussed. Pattern recognition and specifically character recognition methods will also be reviewed in this chapter.

2.2 Preprocessing Methods

Character recognition has long been a research interest because it is a crucial component of an intelligent man-machine system. In this branch of intelligent

science, there are as many other steps of important subtasks as there should be to enhance efficiency. These subtasks are named preprocessing. The name comes from the idea that the scanned image containing the text to be recognized needs to be prepared with respect to different aspects before being supplied to the recognizer.

The need to deal with various preprocessing steps is that, in actual fact, the performance of recognition logic is very strongly influenced by the quality of the scanned image presented to it. Thus each preceding preprocessing will have a significant effect on the subsequently processes of the recognition system. Especially in handwritten systems, the noise or distortion of important parts of the image is very noteworthy. Thus, these kinds of recognition systems should tolerate a certain amount of distortion and misclassification of characters.

The preprocessing task of OCR systems starts with the acquisition of the text image using the available scanning and digitization equipment. After digitization of the text image, different noise removal tasks like slant correction, broken image repair, salt and pepper removal can be undertaken according to the need to do so. After these and other important processes, each logical character should be isolated from the rest by analyzing the foreground and background pixel colors. The segmented characters will be the point of interest for the next processing and recognition tasks. The isolated characters will go through various courses of actions, which will help to attain good recognition rate and efficient OCR system. Some of these processes are size normalization, thinning, image restoration, smoothing, slant correction and so on.

The very basic need to go after all the preprocessing steps in character recognition systems is to pave the way for the recognizer to be more accurate and efficient in its

processing and recognition. Thus, it is to be marked that preprocessing helps to reduce the amount of redundant information, simplify segmentation and classification and enhance performance of the recognition system.

The available preprocessing steps can be categorized into two broader classes. These classes are mandatory and optional preprocessing. The first class, mandatory preprocessing, are those steps that should be included in every OCR system. On the other hand, the optional preprocessing steps are those steps taken into consideration according to the need based on the requirement problem at hand. The optional processes may not be applied to a problem at hand depending on its characteristics.

2.2.1 Mandatory Preprocessing Steps

The subsequent sections of this part will be dedicated to the description of the preprocessing steps that are required in most character recognition systems. The steps are termed as mandatory since the recognition engine could be handicapped without these preceding processes.

2.2.1.1 Digitization

The basic idea behind any OCR system is digitizing a printed document to gain access to all the facilities provided by an electronic document. Thus for this purpose, the first step of any pattern recognition, particularly character recognition system, will be the acquisition of the hard copy of the document in a form of raster image. This step of an OCR system is called digitization or sometimes it may also be termed as process of scanning the document. Thus, converting paper document, which is the classical but dominant analog storage media, to electronically editable format will start from digitization [Srihari et al 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. The most commonly used format is the bitmap format, which is dominantly used for the case of OCR [Pandya & Macy, 1996]. The price of such computer peripherals is reducing and the quality of image produced by the device is getting better from time to time making the quality of inputs to recognition systems more precise.

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. 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 & Macy, 1996].

2.2.1.2 Image Binarization/Thresholding

After the process of appropriate digitization, 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. Practically, image binarization is the process of converting an image of up to 256 gray levels to a black and white image. 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 pixels rather than a set of true colors.

In practice not all pixels of a scanned or digitized document have same intensity of colors when the image is converted into gray-scale level. For the cases where the intensity of pixels in the gray-scale level is not uniform, the simplest way to use in image binarization is to choose a threshold value and classify all pixels with values below this threshold as black, and all other pixels as white. The problem in this approach is then how to select the correct threshold value. In many cases, this threshold value is selected by ~~experimentation~~. Different algorithms use various approaches to select this threshold value in global as well as local manner.

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.

As pointed out earlier, thresholding 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 conventional or global thresholding and the other being adaptive thresholding.

Whereas the conventional thresholding scheme uses a global threshold value for all pixels found in the image matrix, adaptive thresholding changes the threshold dynamically over the image. This change of threshold value is done for each specified area of the image. In conventional thresholding, in a single pass, each pixel in the image is compared with a global threshold value. If the pixel's intensity is higher than the threshold, the pixel is set to, say white, in the output and if it is less than the threshold, it is set to black.

For the case of adaptive thresholding, for each pixel in the image, 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.

Thus the mathematical representation of thresholding algorithm can be put as:

Thresholding Algorithm:

INPUT: Gray scale image $f(i,j)$; where i and j are row and column numbers of the target pixel.

OUTPUT: Binary image $g(i,j)$; where i and j are row and column numbers of the target pixel.

PROCESS

- For each pixel at position (i,j) in the INPUT gray scale image
- Get the intensity of the target pixel $f(i,j)$
- Compare the intensity value with some threshold value T
- Set the corresponding pixel in the Binary image as either foreground or Background pixel

If $f(i,j) > T$ then $g(i,j) = 1$ (*White*)

Otherwise $g(i,j) = 0$ (*Black*)

Figure 1: Thresholding Algorithm

As discussed above the threshold value, T could be computed using one of the approaches of threshold selection based on the input document quality and the intended binarization algorithm method. The trick is how to select the best threshold value that most of the time computed and used after intensive experimentation on different values of parameters.

2.2.1.3 Segmentation

In offline OCR systems, after the binarization process, one of the other critical preprocessing steps is image segmentation. There are various types of segmentation ranging from text block segmentation to character stroke segmentation. For the case

of character segmentation, at this stage of the preprocessing, the lower level of the writing system is isolated from the raster image.

Character segmentation is the process of extracting a single lower level abstraction of a writing system from a text, which requires further examination. This process is also called character tracing that involves locating of each character in a digitized document. Since the character segmentation subsystem will accept the binary form of the digitized image, discriminating the white and black pixels will do the tracing.

The character isolation progression in this preprocessing phase of a character recognition system involves the scanning of the document row by row from left to right and from top to bottom until all the characters in the document are extracted. Any failure or error in the segmentation step produces a negative effect on the character classification. Due to this reason, the segmentation process must be very robust and effective. As S.W. Lee et al [Seong-Whan Lee, et al, 1996] described, the complexity of character segmentation stems from the wide variety of fonts, rapidly expanding text style, and image characteristic such as poor quality printing and poor binary images. He also pointed out that touched; overlapped, separated, and broken characters are major factors for causing segmentation errors [[Seong-Whan Lee, et al, 1996].

While Character segmentation is often possible for machine-printed text, it is more difficult for handwritten material and is perhaps the major problem encountered in reading cursive scripts.

A very well known and successful segmentation algorithm for none cursive and unconnected characters is the stage-by-stage segmentation that involves isolating lines, words and characters respectively by use of projection profile. The histogram developed by the projection presents the frequency of grayscale values in an image either in row or column wise. The three successive steps of segmentation help to segment each line from the document; then each word from each line and every character from each word. For this purpose, a histogram, which counts the number of black pixels either in a row or column of the binary image, is used to analyze each region. Thus the projection profile in gray-scale image is used to segment the lines as well as the characters in the raster image.

The row wise projection and histogram construction is used to segment or isolate lines from a document while column wise projection and histogram construction is used to segment words and characters based on the number of white lines between consecutive objects or characters during the character tracing.

In general the different methods of character segmentation can be classified into three categories; straight segmentation method, recognition based or recursive segmentation method, and cut classification method.

Straight Segmentation Method

In straight segmentation method, each word is segmented into several characters and the isolated characters are fed to the character recognition engine. This method of character segmentation highly depends on the power of the segmentation point of

each logical unit [Kahan S., et al 1987]. The power of the segmentation point can be interpreted as the capability of the threshold value to segment all individual characters from the rest. The method can also be easily affected by noisy pixels. According to Pal and Chaudhuri [1995], the result obtained when implementing this kind of segmentation on printed Bangla script, the result is reported to be good and not sensitive to noise as long as the text is not skewed.

Recognition-Based Segmentation

In recognition-based segmentation, the raster image is analyzed and a set of possible segmentation points will be pointed out [Seong-Whan Lee, et al 1996]. These candidate segmentation points are discriminated based on a recognition result. This segmentation approach feed the area found by each candidate point of segmentation to the recognizer and if that region is correctly recognized it will continue to the next character taking the preceding as one individual character. If the recognizer could not classify the region, the next candidate point of segmentation will be taken for the same analysis. Thus, the recognition-based segmentation will highly depend on the performance and accuracy of the recognizer.

A similar approach is suggested by Casey and Nagy [1982], which merges the segmentation and recognition tasks. As described by the authors, the text line will first be identified and a sequence of pattern arrays where there is a sufficient white space will be identified. This process will segment untouched characters correctly and overlapping characters will be taken as one pattern. If a pattern array fails to be recognized, the pattern array will be narrowed down from the right and the truncated image will be sent for recognition. This process continues until the pattern is

recognized or the pattern array is less than a specified length. This algorithm is tested on different documents and the result is reported to be remarkable. The authors reported that for documents with 2000 characters for which seven characters were touching; only one error is found [Casey and Nagy, 1982].

Cut Classification Method

The third method, which is the cut classification method, is based on the classifiers decision whether it represents a cut hypothesis or not, for each column of the character image [Seong-Whan Lee, et al 1996]. In this approach, along with the classification result of the recognizer, some hypothesis will be set on how to segment the characters and the combination of the result of the recognizer and the hypothesis will be considered.

Most of the studies conducted on Amharic character recognition used stage-by-stage segmentation for isolating or extracting each character from the raster image. From the success of the previous researches, this method of character segmentation will also be used in this work. The reason for going for the stage-by-stage character recognition, which deals with line segmentation, word segmentation and character segmentation using vertical and horizontal projection is that, the scripts are written in such manner that no character touch one another and words are separated by special character, which is a colon (:). After the character isolation, different other preprocessing algorithms need to be used to achieve a higher rate of recognition.

Stage-by-Stage Segmentation Algorithm:

INPUT: Binarized scanned image document

OUTPUT: An electronic document with characters being isolated and identified from the image

PROCESS:

- Do horizontal projection profile to identify the lines
- Select the top and bottom line of a logical text line based on the number of black pixels.
 - If the top logical line is not identified and the number of black pixels is greater than some threshold value it is the indication of the top line of the logical line.
 - If the top line of the logical line is identified and the number of black pixels is less than the threshold vale it is the indication of bottom line of the logical line.
- For each of the identified logical text lines
 - Do Vertical projection profile
 - If the left border of a character is not identified and the number of black pixels is grater than the predefined threshold value it is the indication of left border of the character
 - If the left border of a character is identified and the number of black pixels is less than the predefined threshold value, it is the right border of the character.

Figure 2: Stage-by-Stage Segmentation Algorithm

2.2.2 Optional Preprocessing Steps

In this section of the paper, some of the major optional preprocessing steps will be described and their contribution towards making the recognition be efficient will be pointed out.

2.2.2.1 Thinning

In a writing communication system, the symbols and combination of them to represent the sounds in a language will be put on the writing material occupying some stroke width. Most of the time, the width of a character furnishes no significant information to the character recognizer. Due to this reason most pattern recognition systems involving image analysis will add one preprocessing step that erodes the character stroke to make the image of the character one pixel wide. This preprocessing step is called character thinning or skeletonization.

The motivation for character thinning arises from the fact that the stroke width of the characters obtained by the method of character segmentation and the preceding preprocesses is not uniform. Moreover there is unevenness on the stroke width due to the variation of the writers and the imprecision of the writers. In order to resolve these problems, smoothing is needed. And as portrayed above, the smoothing is done by thinning.

2.2.2.2 Size Normalization

As in the case of stroke width of handwritten characters, it has been pointed out that characters of any writing system will not have the same height and width. The case will be much more apparent when the problem domain is on handwritten text where dissimilarity is observed for a same writer let alone different writers. Research pointed out that the major sources of variation in handwriting are size (height), slant, skew (slope), stroke width and rotation [Senior A. W. and Robinson A. J., 1998] [Simon J. C., 1992]. If one system minimizes these and the like constraints the efficiency of the system will boost up, which is the demand of the information society.

Thus size normalization or scaling will help to make all characters encountered in the scanned image to be of same width and height. This preprocessing step will help to effect same action on all isolated characters in the coming stages of the recognition process. This normalization is more crucial when the classifier is a size non-variant neural network. This type of classifier will accept same size of input patterns for each character representation to classify the input characters according to the pattern observed at each selected area of the binarized image.

Just like any preprocessing algorithm, there are various size normalization approaches. These approaches are expected to be loss less algorithms where after the application of the algorithm on the bounded character all the features of the character must be preserved.

The general algorithm for size normalization schemes is shown as follows:

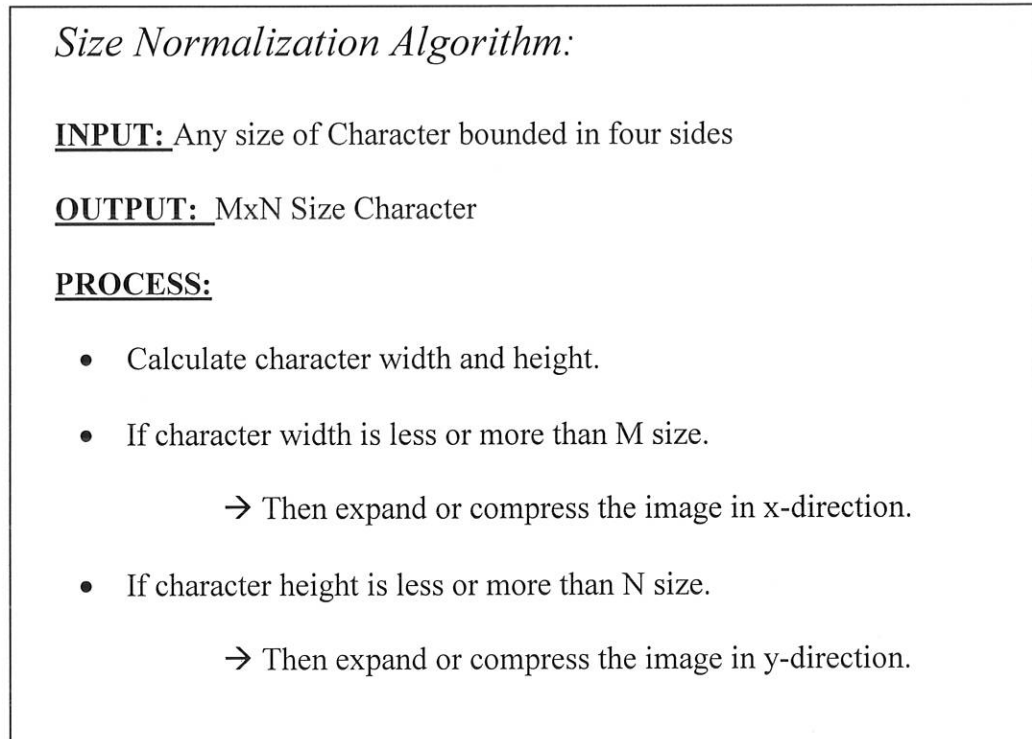


Figure 3: Size Normalization Algorithm

2.3 Techniques in Pattern Recognition

Being one of the branches in computational and machine intelligence science, pattern recognition has various approaches to solve problems in its domain. These different approaches of the branch could be termed as techniques of pattern recognition. Pattern recognition as a science is the study of how patterns in real world or in the brain of human being are processed and how to adopt the techniques to the machine. This way the machine will have the power of identifying patterns and making decisions based on the predefined features or past experience based on the type of the recognition approach.

One of the well-studied problems in pattern recognition is character recognition that deals with mapping bitmap representation of lower level abstraction of a written or printed language, which is a stroke or a character, to its computer system code.

Towards developing a robust and efficient character recognition systems for different languages, various techniques of Pattern Recognition such as Template Matching, Neural Networks, Syntactical Analyses, Wavelet Theory, Hidden Markov Models, Bayesian theory etc have been explored [Wang, 2001]. These different techniques use a good many of algorithms and approaches towards attaining same goal, which is better recognition rate and robust OCR system [Wang A.B. & Fan K.C., 2001].

Pujari et al [2002], describe one approach of character recognition, which is feature-mapped recognition as, the recognition task accomplished by extracting certain primitives, or distinctive features of the characters input image [Pujari, et al, 2002]. In this approach each input character will be subjected to a module or routine which extract some features or characteristics observed in the whole script and the result will be used to classify the input image as one of the set of possible characters or combination of strokes. Thus in feature mapped recognition approach; the individual characters are recognized based on a decision function that settles on the presence or absence of different primitive components in the character.

In contrast to feature-mapped recognition, for the case of Image Mapped approach the identification and the extraction of features are implicit processes within the recognition process. This means that the classification of the recognizer is embedded

inside some implicit modules. Burr [1988], point out that Neural Network based OCRs, which is taken as the most robust system architecture since it uses the concept of nervous system follow this principle.

There is another character recognition approach reported by Brown [1992]. The method approaches the problem by analyzing the way human beings identify single characters. The author claims that it is not the length or width of a character or a stroke that gives more information to the reader. It is the intersection between the lines and the relationship between lines. Based on the algorithm, such features like intersection points; holes and curves will be extracted from an input image and will be checked against a set of known feature points. The algorithm also claimed to consume much time and not efficient for images with greater than an 8x8 matrix representation.

Another recognition approach was also proposed by Pal and Chaudhuri [1995] which uses a tree classification method to the feature extraction approach. The algorithm works in such a way that a tree will be constructed using the features and an input image is checked for existence of a feature at each root node until a leaf node is reached. All characters of the language will be represented by a leaf node where the input image is traversed through nodes of the tree and will select one of the leaf nodes. The character represented by that node will be the best approximation to the input image. This algorithm is reported to have 96.55% recognition accuracy on Bangla scripts [Pal and Chaudhuri 1995].

Blumenstein & Kulkarni [1998] also share the idea of Burr. They described how handwritten OCR research is shifting to approach of neural network classification;

".....has led researchers to use more complex methods for preprocessing, feature extraction and classification. Such methods include the use of Neural Networks and Mathematical Morphology."

".....the task of finding a suitable classification technique (for individual characters and whole words) has been exhaustively pursued. However, again the variability of handwriting and the lack of reliable feature extraction and preprocessing techniques have impaired many unconstrained approaches. For most of the aforementioned problems, including feature extraction and classification, researchers have turned to complex and intelligent methods. The use of Neural Networks has become extremely popular, and may hopefully enable researchers to move closer to solving the handwriting recognition problem."

2.4 Artificial Neural Network for Character Recognition

In this section, a brief description on what neural network is and how it will be applied to pattern recognition in general and character recognition in particular will be given.

Neural networks were initially studied by computer and cognitive scientists in the late 1950s and early 1960s in an attempt to model sensory perception in biological organisms. Neural networks have been applied to many problems since they were first introduced, including pattern recognition, handwritten character recognition,

speech recognition, financial and economic modeling, and next-generation computing models. [Encarta, 2001]

2.4.1 The Natural/Biological Neural Network

Generally speaking, neural networks fall into two categories: artificial neural networks and biological neural networks. Artificial neural networks are modeled on the structure and functioning of biological neural networks. Since the motivation of the artificial neural network arise from the computational and pattern analysis of the biological neural network, it is important to have an insight about the biological neural network.

The most familiar biological neural network is the human brain. The human brain is composed of approximately 100 billion nerve cells called neurons that are massively interconnected. Typical neurons in the human brain are connected to on the order of 10,000 other neurons, with some types of neurons having more than 200,000 connections [Tom, 1997]. The extensive number of neurons and their high degree of interconnectedness are part of the reason that the brains of living creatures are capable of making vast number of calculations in a short amount of time.

The main characteristic of a biological neural network is that each neuron, or nerve cell, receives signals from many other neurons through its branching dendrites. The neuron produces an output signal that depends on the values of all the input signals and passes this output on to many other neurons along a branching fiber called an axon.

2.4.2 The Artificial Neural Network

In its general sense, Artificial Neural Network is a highly interconnected network of information-processing elements that mimics the connectivity and functioning of the human brain. Compared to a human brain, conventional programs are poor in dealing with uncertainty and new patterns. These features of the human brain can be modeled using the artificially designed neural network.

Neural networks are designed to address problems that are often very complex for traditional computers to solve, such as speech and pattern recognition. One of the most significant strength of neural networks is their ability to learn from examples. Motivations for modeling natural neural networks artificially rose from the learning and highly parallel computation capability of the biological neural network.

Neural networks differ greatly from traditional computers in the way they are programmed. Rather than using programs that are written as a series of instructions, as do all traditional computers, neural networks are taught or more formally trained with a set of training examples. The network is then able to “learn” from the initial examples to respond to information sets that it has never encountered before. This feature of Artificial Neural Network is called learning capability of the system.

There are some sets of real world problem that can robustly be solved by artificial neural network. These problems are kinds with data corresponding to noisy, complex, and subject to high degree of distortion.

According to Mitchell, problems with some specific characteristics can be tackled using artificial neural network computation. These characteristics of the problems are [Tom M. Mitchell, 1997]:

- Many attribute-value pairs represent instances of the problem, where the values could be highly correlated or independent of one another.
- The target function output may be discrete-valued, real-valued, or vector of several real- or discrete-valued attributes.
- The training as well as test examples may contain errors or noisy data.
- Longer training times are acceptable.
- Fast evaluation of the trained or learned target function may be required.
- Understanding the learning and decision function of the target by human is not important.

2.4.2.1 Design of an Artificial Neural Network

The design issues in neural networks are complex and are the major concerns of the system developers. Designing a neural network consist of the following set of decisions and activities.

- Deciding the number of layers in the network.
- Arranging neurons in various layers.
- Deciding the type of connections among neurons for different layers, as well as among the neurons within a layer.
- Deciding the way a neuron receives input and produces output.

- Determining the strength of connection within the network by allowing the network to come up with appropriate values of connection weights by using a training data set.

In designing an artificial neural network, one cannot expect to come up with the best model/network in few shots. The process is an iterative course of action that achieves an optimal result by adjusting the parameters of the network depending on the proximity of the output with the desired target.

2.4.2.2 Layers of an Artificial Neural Network

Any ANN could have any number of layers, where the neurons will be grouped into these layers. The network could be divided into three parts according to the activities of the neurons in the layers. The first part is the input part where the neurons in this layer are fully responsible for the task of interfacing the real world to receive its inputs. The second part is the set of layers; one or more hidden layers, responsible for performing some form of computation on the input received by the input layers. The last and third part will be the set of neurons in the output layer responsible for interfacing the real world by providing network output.

When the input layer receives the input its neurons produce output, which becomes input to the other layers of the system. The process continues until a certain condition is satisfied or until the output layer is invoked and fires their output to the external environment.

When described according to the activity that would be undertaken at each layer, the general description of each layer is as follows:

Layer	Activity
Input	The activity of the input units represents the raw information that is fed into the network. The input layer is the interfacing point between the outer environment and the network. These nodes are accessible for the training environment and may therefore have signals directly imposed on them from outside. The data to be trained has to be in the format, which is suited for the input nodes.
Hidden	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 hidden nodes reside between input and output layers. They are not accessible by the environment. It is the task of these units to develop an internal representation of the training set.
Output	The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units. The output layer is the interfacing point between the network and the outer environment. These nodes are the complementary of the input nodes.

Table 1: Layers and their Activities

2.4.2.3 Artificial Neural Network Architecture

The architecture of a neural network, as for any computational system, is the specific arrangement and connections of the neurons that make up the network. The most common neural network architectures have three layers. The first layer is called the input layer and is one of the layers exposed to external signals. The input layer transmits signals to the neurons in the next layer, which is called a hidden layer. The hidden layer extracts relevant features or patterns from the received signals. Those features or patterns that are considered important are then directed to the output layer, the final layer of the network. A typical three layer artificial neural network is presented in the figure below.

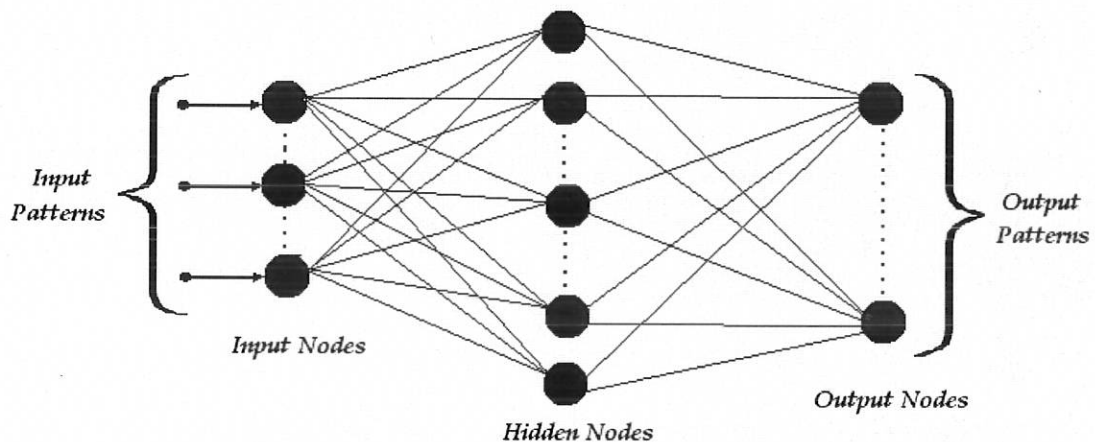


Figure 4 : Three Layer Artificial Neural Network Architecture

In an artificial neural network, input signals, such as pixel value of a segmented character, or one attribute value of a variable, fall on a layer of input nodes, or computing units. Each of these nodes is linked to several other “hidden” nodes between the input and output nodes of the network. Each hidden node performs some sort of calculation on the signals reaching it and sends a corresponding output signal to other nodes. The final output is a highly processed version of the input.

To mimic the way in which biological neurons learn patterns, the connections between artificial neurons in a neural network are given adjustable connection weights, or measures of importance. This value of the connection will keep changing during the training session to hold the optimal value so as to make the best approximation to new patterns.

When signals are received and processed by a node, they are multiplied by a weight, added up, and then transformed by a nonlinear function. The effect of the nonlinear function is to cause the sum of the input signals to approach some value, usually +1 or 0. If the signals entering the node add up to a positive number, the node sends an output signal that approaches +1 out along all of its connections, while if the signals add up to a negative value, the node sends a signal that approaches 0.

2.4.2.4 *Communication and Types of Connections*

Since neural network is the interconnection of many neurons, the connection behavior of the neurons needs to be studied to have good understanding on how the network works. Each input and hidden layer neurons are connected via a network of paths carrying the output of one neuron as input to another neuron. These paths are normally unidirectional, there might however be a two-way connection between two neurons, because there may be another path in reverse direction. A neuron receives input from many other neurons, but produces a single output, which is communicated to other neurons.

The neuron in a layer may communicate with each other, or they may not have any connections within their layer. But it is mandatory that the neurons of one layer be always connected to the neurons of at least another layer.

There are different types of connections used between neurons of different layers. These connections between layers are called inter-layer connections, while connection of neurons within one layer is called intra-layer connection. The type of connections between neurons of different layers could be:

- **Fully connected:** Each neuron on the first layer is connected to every neuron on the second layer.
- **Partially connected:** A neuron of the first layer does not have to be connected to all neurons on the second layer.
- **Feed forward/uni-directional:** The neurons on the first layer send their output to the neurons on the second layer, but they do not receive any input back from the neurons on the second layer.
- **Bi-directional:** There is another set of connections carrying the output of the neurons of the second layer into the neurons of the first layer.
- **Hierarchical:** If a neural network has a hierarchical structure, the neurons of a lower layer may only communicate with neurons on the next level of layer.
- **Resonance:** The layers have bi-directional connections, and they can continue sending messages across the connections a number of times until a certain condition is achieved.

2.4.2.5 *Training the Artificial Neural Network*

It is pointed out that the excellence of artificial neural network is its ability to learn with examples in the real world and capability to deal with new patterns. Learning in the context of ANNs means, that the network can adopt different behavior on the basis of the data that is given to the network. Unlike telling the network how to react to each data vector separately, as would be the case in the conventional programming, the network itself is able to find properties from the presented data. The network learning can be continued as new data becomes available. Learning is thus adaptive for the case of artificial neural networks.

Then how the network learns new things is the critical question that should be asked. Artificial Neural Networks are taught by presenting them with desired input-output combination of training sets. The input will be a set of attribute values or vector that can make the generalization power of the network to increase and the output is the set of values expected for that specific input pattern. After being supplied with input patterns and the desired output pattern for each input, the network will do some adjustment on its parameters to approximate the output of the network with the desired output.

Input and hidden neurons will be connected to the neurons of the immediate neighborhood layer with some connection strength or weight. Beginning with equal, or random, connection weights between the neurons, the inputs are digitally entered into the input layer of the neural network and an output signal is computed and compared to the target output. Small adjustments are then made to the connection weights to reduce the difference between the actual output and the desired output. Another input/output set is again presented to the network and further adjustments are

made to the connection weights so that the network will have a chance to encounter different patterns of inputs. This process will help the network to have an optimal strength between connections of neurons at different level.

The iterative training task is required for the reason that the first few times that the input is entered, the network will usually choose the incorrect output neuron value. After repeating the weight-adjustment process many times for all input-output patterns in the training set, the network learns to respond in the desired manner. If the network is trained with appropriate training set which has the power to represent the whole set, then the network is capable of classifying new patterns in their most probable class.

2.4.2.6 *Neural Network Learning Rules*

The brain basically learns from experience. Artificial neural networks are sometimes called machine-learning algorithms, because changing of its connection weights during training causes the network to learn the solution to a problem. The strength of connection between the neurons is stored as a weight-value for the specific connection. The system learns new knowledge by adjusting these connection weights. The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training.

To have a working network, the empty network must be trained to handle different events. To proceed with the learning process, there are various learning methods which are called learning rules. A number of different neural network learning rules, or algorithms, exist and use various techniques to process information. All the learning rules use some sort of system to adjust the connection weights between the

supervised learning, because it requires a teacher. The teacher may be a training set of data or an expert who evaluates the performance of the network results.

Both unsupervised and reinforcement suffers from relative slowness and inefficiency relying on a random shuffling to find the proper connection weights.

3. Supervised Learning

This method is proven highly successful in training of multilayered neural networks. The network is not just given reinforcement for how it is doing on a task. Information about errors is also filtered back through the system and is used to adjust the connections between the layers, thus improving performance. Back propagation is a typical case of supervised learning.

Basic steps in the learning process of supervised learning;

- A pair of input and output vectors presented to the neural network
- Forward pass with the input vector
- The errors between the desired and actual, computed by the forward pass, response for each node in the output layer are found
- These errors are then used to determine weight changes in the network according to the prevailing learning rule.
- This forth and back process will continue until an optimal solution is found

The most widely used scheme for adjusting the connection weights is called error back-propagation. The back-propagation learning scheme compares a neural network's calculated output to a target output and calculates an error adjustment for each of the nodes in the network. The neural network adjusts the connection weights

according to the error values assigned to each node, beginning with the connections between the last hidden layer and the output layer. After the network has made adjustments to this set of connections, it calculates error values for the next previous layer and makes adjustments. The back-propagation algorithm continues in this way, adjusting all of the connection weights between the hidden layers until it reaches the input layer. At this point it is ready to calculate another output.

2.4.2.7 *Learning Rates*

The learning rates are the rates at which artificial neural networks learn how to deal with new patterns. This depends upon several controllable factors that have their own effect on the performance of the system. Obviously, a slower rate means a lot more time is spent in accomplishing the off-line learning to produce an adequately trained system. With the faster learning rates, however, the network may not be able to make the fine discriminations possible with a system that learns more slowly.

Network complexity, size, architecture, type of learning rule or rules employed, and desired accuracy must all be considered. These factors play a significant role in determining how long it will take to train a network. Changing any one of these factors may either extend the training time to an unreasonable length or even result in an unacceptable accuracy.

Most learning functions have some provision for a learning rate, or learning constant. Usually this term is positive and between zero and one. If the learning rate is greater than one, it is easy for the learning algorithm to exceed in correcting the weights, and the network will move back and forth. Small values of the learning rate will not correct the current error as quickly, but if small steps are taken in correcting errors, there is a good chance of arriving at the best minimum convergence.

2.4.2.8 *Learning laws*

As learning in artificial neural networks is the process of adjusting the connection strength between successive neurons, there should be some predefined law on how to adjust the connection weights. The learning laws are mathematical formulas or algorithms used to make adjustment on connection weights. Most of these laws are sort of variation of the best-known and oldest learning law, Hebb's Rule. The rule is defined as follows;

Hebb's Rule: The first and the best-known learning rule was introduced by Donald Hebb. The description appeared in his book "*The organization of Behavior*" in 1949. This basic rule is: "*If a neuron receives an input from another neuron, and if both are highly active, i.e. mathematically have the same sign, the weight between the neurons should be strengthened and vise versa*".

2.4.2.9 *Multi Layer Perceptron (MLP)*

The most common neural network model is the multilayer perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown.

The MLP and many other neural networks learn using an algorithm called backpropagation. With backpropagation, the input data is repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (backpropagated) to the neural network and used to adjust the weights such

that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as “*training*”.

Backpropagation

The multi-layer perceptron (MLP) 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 three-layered perceptron network training. The backpropagation algorithm learns the weights for a multilayer network given a fixed set of units and interconnections. This algorithm implements gradient descent to attempt to minimize the sum of square error between the network output and the desired output during training.

The learning process in backpropagation is an iterative action processing a set of training samples, comparing the network’s prediction for each sample with the actual known output. For each such input patterns, the weights at each connection will be modified so as the sum of mean squared error between the actual and desired output will be minimized. The modification starts from the back, weights between the output and last hidden layer, and propagates to the front, weights between the first hidden layer and the input layer.

The sum of mean squared error, at each iteration, will be calculated as follows.

$$E(\vec{w}) = \frac{1}{2} \sum_{(d \in D)} \sum_{(k \in \text{Outputs})} (t_{kd} - o_{kd})^2$$

Figure 5: Network Output Error Formula

Where:

$E(w)$ is the sum of mean squared error over the network

Outputs is the set of output units in the network

t_{kd} is the target value associated with the k^{th} output unit and training example d .

o_{kd} is the output value associated with the k^{th} output unit and training example d .

The general algorithm for Feedforward backpropagation learning algorithm is presented as follows:

Backpropagation Algorithm:

INPUT: A multilayer Feedforward network, the training sample, and the learning rate.

OUTPUT: A neural network trained to classify

PROCESS:

1. Initialize all weights and bias in the network
2. While terminating condition is not satisfied
 - For each training sample X from the sample set
 - i. For each hidden or output layer unit j

$$I_j = \sum_i w_{ij} O_i + \theta_j$$
 Net inputs of unit j with respect to the previous layer i.

$$O_j = \frac{1}{1 + e^{-I_j}}$$
 The output of each unit j
 - ii. For each unit j in the output layer

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$
 Compute the error at each output unit
 - iii. For each unit j in the hidden layers from the last to the first hidden layer

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$
 Compute the error for the next higher layer
 - iv. For each weight w_{ij} in the network

$$\Delta w_{ij} = lr * Err_j O_i$$
 Weight increment where lr is the learning rate

$$w_{ij} = w_{ij} + \Delta w_{ij}$$
 Update the weight
 - v. For each bias θ_j in the network

$$\Delta \theta_j = lr * Err_j$$
 Bias increment where lr is the learning rate

$$\theta_j = \theta_j + \Delta \theta_j$$
 Update the Bias

Figure 6: Backpropagation Algorithm

2.5 Attempts on Amharic OCR

Worku Alemu who was a postgraduate student at SISA (School of Information Studies for Africa) first tried out optical character recognition prototype for Amharic characters. Worku attempted to test the applicability of some OCR techniques on the Amharic script, with a view to adopt a working algorithm [Worku, 1997]. Worku studied the feature of the 231 core characteristics along with the various segmentation and recognition algorithms used to recognize scripts in other foreign languages. For the case of classification Worku used a tree classification scheme built using topological features of a character. He took onboard the stage-by-stage segmentation and topological features based recognition algorithms. After selecting and modifying the algorithms he wrote a Visual C++ program to develop the prototype and test the performance of the selected algorithms. The OCR system developed by Worku was able to recognize 97.31% of the main test case, which is a Laser printout of Amharic text with normal typestyle of WashRa font with 12-point font size.

In the year 1998, Ermias dealt with the problem of formatted Amharic text recognition. His work concentrated on underlined and italicized characters. In his work, Ermias used thinning algorithm to transform the input characters into one pixel wide format and he also implemented algorithms to remove underline. In his work, Ermias attempted to improve the Amharic OCR system developed by Worku by incorporating the facility of handling formatted Amharic text recognition. For this purpose, Ermias used preprocessing algorithms for thinning

and underline detection and removal. After analyzing the problem and the existing thinning algorithms, Ermias selected the Zang-Suen's thinning algorithm for future consideration and a recognition rate of 89.25% is reported [Ermias, 1998].

In 1999 two attempts in the area of Amharic OCR were made. Dereje, graduate of SISA considered the development of Amharic OCR prototype for type printed characters [Dereje 1999]. Dereje Teferi also tried to improve the capability of Amharic OCR by recognizing poor quality typewritten Amharic document. To this end, Dereje used preprocessing algorithms for image enhancement and character building. The algorithm adopted for image restoration is a binary morphological filter. In addition, segmentation and recognition algorithms originally adopted by Worku were modified to suit to the peculiar characteristics of typewritten Amharic characters. The result of the OCR system developed by Dereje produce a recognition accuracy of 53.47% for documents written with mechanical typewriters.

Birhanu made another attempt in a same year, 1999; he considered the case of computer printed characters to be recognized [Birhanu, 1999]. These attempts were considering characters with similar font type and font size. Berhanu, who was a postgraduate student at technology faculty of AAU, studied the use of ANN (Artificial Neural Network) for the recognition of Amharic text. In his work Berhanu used the approach that after segmenting characters using suitable segmentation algorithms, they are fed to a feed forward neural network. A feed forward neural network with 256 input, 40 hidden and 8 out put nodes was

designed and trained using the back propagation algorithm on selected Amharic characters as training sets. The system was tested on printed Amharic document, which is written using ALTEX Ethiopian font with 16-point size and a recognition accuracy of 35% was obtained. Berhanu used MATLAB image processing toolbox, which extracts the character image in its boundary box.

Yaregal Assabie also tried to explore the various OCR development approaches, techniques and methodologies. He as well tried to investigate the possibilities of adopting and/or developing algorithms for Amharic character recognition [Yaregal, 2002]. Yaregal showed the very basic concepts in OCR technology and put light on various character recognition techniques and preprocessing methods. The classification/recognition of the developed OCR system used neural networks. BrainMaker neural network software was used for this experiment. The network has 64 input nodes, 64 hidden nodes, and 8 output nodes. A supervised learning with a back propagation algorithm is used by Yaregal to train and select the best network model. This network model is integrated with the developed OCR system.

The system developed by Yaregal was trained with 10 and 12 font sizes of VG2000 Agazian font type. After the training the system was tested with documents of font size 8, 12, and 14. The system correctly recognizes 65.02% and 74.68% characters of size 8 and 12 respectively, which were included in the training set. It also classify 62.87% and 81.07% of the characters not included in the training as unknowns for the two font sizes respectively. Yaregal claimed that the result for documents with font size 14 was not far from the mentioned two font

size results. Generally the system developed by Yaregal correctly recognized 73.1% of the characters included in the training set and 70.04% of the characters not included in the training were classified as unknown

Nigussie, who is also a graduate of SISA had worked on the recognition of Handwritten Amharic legal amounts of checks, the purpose of which is investigating the application of neural networks as a tool to recognize handwritten Amharic characters for the purpose of processing bank checks [Nigussie 2000]. Nigussie used stage-by-stage character segmentation, and the characters were normalized after segmentation to 16x16 pixel matrix and converted to one-dimensional array to be input to the neural network. Nigussie claimed that even though some preprocessing steps were applied to the raster image, the classification result of the neural network was not found to be satisfactory. The reason for this unsatisfactory result was attributed to insufficient sample data supplied for the neural network.

In the year 2000, Million Meshesha, another graduate of SISA undertaken his thesis research work making his objective to generate a generalized approach to make previously adopted recognition algorithms to handle Amharic texts printed in different fonts. Million, in his research work, reviewed the works done by Worku (1997), Ermias(1998), Dereje(1999) and Brhanu(1999) and other literatures and considered some thinning and feature extraction algorithms. After taking different algorithms for further consideration, a Visual C++ code is developed for each algorithm and using different test case the efficiency for each

algorithm as well as the integrated system is tested. The test case Amharic documents were written using Agafari (Agafari-Addis Zemen, Agafari-Rejim), WashRa, and Visual Geez fonts.

In his work, Million integrate the Zang-Suen and Parallel thinning algorithms to come up with a hybrid thinning algorithm, which resulted in no loss of connectivity and only 4% of shape distortion. He also modified the stage-by-stage segmentation algorithm by modifying the threshold for the thinned characters. Finally, million reviewed the topological feature extraction algorithm and extracted some features and tested the modification on the different fonts selected for the case. The accuracy obtained by the recognition system for the test fonts are; 49.38% for WashRa, 26.04% for Agafari-Addis Zemen, 27.64% for Agafari-Rejim and 15.75% for Visual Geez fonts.

Recently Messay Hailemariam made an attempt to solve the problem of identifying postal addresses written in Amharic by applying OCR technology. Messay explored and test the application of simple geometric calculations and line fitting for handwritten Amharic Character Recognition by using characters used in writing of destination addresses in Amharic language. Messay used the statistical technique which is a regression line fitting approach by using the least square technique to solve the problem of handwritten character recognition for the case of postal address. The result obtained by the system was with accuracy rate of 91.9%, which is very high for Amharic handwritten text. He attempted to reduce number

generation to another by means of writing which uses those symbols of the language that represent either the alphabet or the phoneme.

In a language where it has its own writing system, every sound is represented by some set of strokes. These combinations of strokes make-up the character set of the language. The character sets or symbols used to represent the sounds of a language are called scripts of that language. After formulating and being modified through the set of characters through time, the speakers of the language will get used to these symbols so that they can use the written language approach to communicate as well as to preserve knowledge.

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]. In his book, "*Language in Ethiopia*", Bender et al [1976] noted that all middle-eastern scripts originate from Egyptian picture writing system, which is called the Hieroglyphics. The Geez script is one of these scripts, since it is derived, in its unvocalized shape, from the South Arabian alphabet (i.e. Sabaean script) [Ullendoff, 1973 as quoted by Dereje, 1999]. Ethiopic script is one of the ancient alphabets in the world that is used to write in languages such as Geez, Amharic, Tigrigna, etc.

According to Bender [Bender et al, 1973], the present writing system of Amharic is taken from Geez, which was brought to the highlands by immigrants from South Arabia in the first century A.D.

As a writing language, the layout of the characters and the direction of the writing will have one format. Concerning the direction of writing the characters, while Sabaean characters are written from right to left, Geez and Amharic and any language that uses the script for writing is written from left to right. In addition to the direction of the writing, unlike other scripts, in the Amharic scripts, the vowels are integrated with the consonants. Thus there are no separate vowel characters in the Ethiopic writing system. 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 than alphabetic. Except for some special characters every stroke in the Ethiopic script is connected. Since the Amharic writing style is syllabic the symbols or scripts used in Amharic writing system represent a phoneme and not a sound.

Unlike the English scripts, the concept of upper case and lower case characters are absent in Amharic script writing system. In addition to the capitalization absence, a line of Amharic script lies at the same level, having no ascent and descent features. This feature of Amharic writing system grant that there will always be a white line between two consecutive lines of characters, unless and otherwise the line is inclined.

When analyzing the number of symbols used in Amharic language writing system, it is composed of a total of more than 330 characters. The basic writing character set is composed of 33 symbols that are core characters. Each of these core characters occurs in 7 orders; one basic form and six non-basic forms representing syllable combinations consisting of a consonant and vowel. In addition to the 231-character set, the 33 core and its 7 forms, there are additional symbols to represent numbers and other rarely used characters, which make up a number of characters in Amharic writing system to be greater than 330. [Bender et al, 1976]. For the whole character set, refer to Annex 6.

As described by Bender et al [1976] the non-basic forms (vocalization) of each core character are derived from the basic forms by attaching small appendages (diacritic marks) to the right, left, top or bottom in more or less regular pattern for the first 5 forms. The last two forms are very irregular in the modification, which makes categorization of the characters difficult. In general it could be concluded that the Amharic characters, formally called “Fedel”, have gone through a serious of changes and shape modifications to have the present form.

3.3 Amharic Handwriting

Handwriting was, and still is, the most dominant means of written communication. Along with its old age and dominance in written communications, it brings difficulty to automation of handwritten documents. In handwritten documents, the

major challenge is the existence and possibility of having connected characters, overlapping strokes, skewed and slant letters along with variation on the size of different as well as same characters at different instances.

In handwriting, variations in spacing are generated from difference in writing style. The problem is even evident for same characters written by one user at different instances. Since handwritten text lacks uniform spacing, stroke layout, alignment, and etc. preprocessing steps are very challenging. In Amharic writing system there is an advantage that the structure of characters provide to the preprocessing steps of OCR engine. This advantage is the feature of Amharic characters being unconnected and the nonexistence of cursive writing unless and otherwise there is overlapping strokes due to the style of the writer.

3.4 Writing style of “Yekum Tsifet”

Before the importation of the devices used in printing presses, all documents used to be handwritten. Most of them were written for religion purpose and correspondence between different administrative organs of the at-then government. Since all the letters need to be read by the receiver, there used to be common writing style in all the writers. This special kind of traditional writing system of Amharic scripts is called “Yekum Tsifet”. This written communication means used to be painted on animal skin and of course is handwriting. The art of producing parchment and handwritten books and documents using this technique is a very significant feature of Ethiopian cultural heritage, and is as well a very old one.

There are different writing materials for handwritten text. But for the purpose of “Yekum Tsifet” the dominant one is vellum or parchment. Parchment is animal skin prepared from any domesticated mammals such as the ox, cow, calf, sheep, goat, horse and wild animals such as lion, tiger, gazelle, antelope, etc. but goat skin more preferred due to the fact that it is a fine white one and can take ink very well.

According to Sergew [1972], in early times, the inhabitants of the opposite side of the Red Sea, the Sabaeans, emerged to Ethiopia, and around 1000 BC brought with them a written language, which began to be used by the indigenous inhabitants. From then onwards, different writing materials were used for the purpose of putting ideas and knowledge on relatively long living materials. In addition to parchments, when we come to the third and fourth centuries A.D., we find a variety of writing materials like stone, metal, clay, and also wood [Sergew, 1972].

Among the materials used for writing on hard materials, ink is the major one. For the case of parchments, there are two kinds of mostly used inks, black and red, which were and still are used in Ethiopia for writing on parchments. The black ink is of a slightly different variety. Its colors are mainly two, charcoal black and greenish black. The former is prepared particularly for magical prayer document writing and the latter on for daily uses.

When analyzing the features of the special writing style, which is the concern of this paper, there are some additional characteristics attributed to it. Kefyalew Merahi, in his book *“The Contribution of Ethiopian Orthodox Church to Ethiopian Civilization”* [1987], put a clear explanation on how to write “Yekum Tsifet”. As per his elucidation, when writing “Yekum Tsifet”, the punctuations play a great role for the readability of the text. In addition to the punctuations, there are some pre-processes on the writing material, which is animal skin [Kefyalew, 1987].

The preparation of the animal skin will help to guide the layout of the characters and the whole structure of the document. According to his narrative, sequences of horizontal and vertical lines are drawn on the animal skin using sharp metal so that each character falls in one block. This feature of the writing style permits character as well as lines and words not to touch each other unless and otherwise a major noise is present.

This same description of how parchment are produced and handwriting on the parchment is proceeded is describe by Assefa Liban. He describes the process, as *“during the preparation of parchment, the writer or scribe will put lines on the parchment to guide the writing. The awls, used for ruling, have a dull lip to protect the parchment from being torn.”*[Assefa, 1987]

Learning how to write using the style of “Yekum Tsifet” is claimed to be difficult. The most difficult process in learning how to write “Yekum Tsifet” is the course of designing and painting the symbols. Because of the difficulty to learn the technique, priority used to be given by devoting more teaching hours. Painting by itself could be a career for the scribes. According to Sergew [1972], gifted and well-trained painters can have the chance to get full time job as a scribe.

The layout of the characters is somewhat different in old times. Quoting W. Wright, Sergew reported that; *“in the fifteenth century.... the handwriting is more angular than at a subsequent years.... The writing style of the sixteenth century.... is very different in appearance from that of the fifteenth. It is more rounded in many of its forms and in the examples known to me, is decidedly small in size and delicate. The changes since then are very minor”*

Concerning the use of punctuations, in this special type of Amharic handwriting system, every word in a sentence is separated from the other using special word separator punctuations. These word separating punctuations are: “Nekut” or colon (:) to separate words in a sentence, “Netela Serez” (፡) which acts as a comma in Latin scripts, “Dereb Serez” (፤) which acts as a semicolon in Latin script and “Arat Netib” (፡፡) which acts as a sentences terminator or period in Latin script [Kefyalew, 1987]. Based on these rules, every word in this special writing system, “Yekum Tsifet”, is separated from the other using one of these separators.

The other peculiar feature of “Yekum Tsifet” is the weight of stroke colors. The text is either written using red or black ink or the combination of two and each stroke is written in bold format. This feature of the writing style will help to identify the information bearing strokes from the text, which is flooded by noise due to the writing material and/or the animal skin.

CHAPTER FOUR

EXPERIMENTATION

4.1 Introduction

This part briefly describes the results of experimentation that took place in implementing different preprocessing algorithms and approaches towards coming up with a workable recognition engine. Thus, in this chapter, discussion at each stage of the development process will be described along with the respective result achieved and the difficulties faced.

As the aim of the research is to devise a recognition engine for a special type of Amharic handwritten text (“Yekum Tsifet”), all the preprocessing tasks and recognition is done on Amharic text document written in the above specified writing style. Hereunder, various tasks of the research starting from data collection to evaluation of the performance of the recognition engine will be discussed.

4.2 Data Compilation

In research like pattern recognition, the primary task is collecting required data and preparing it for further processing. According to this, gathering of text documents written in “Yekum Tsifet” has been conducted. On the course of collecting text documents written in “Yekum Tsifet”, difficulty faced in locating experienced scribes to write the training and test document is worth mentioning. The main reason attributed to this problem is that the writing system can be said obsolete, of

course with the writers, due to the advent of fast text processing machines like typewriter and computers.

As mentioned in section 3.4 the writing style of the problem domain was used to write different types of documents for various purposes, like spiritual books, government correspondence and historical books. And this was during those early times when the society deals with documentation without computer and typewriter. This situation reveals that the number of scribes will decrease from time to time since the introduction of these two mechanical and electronic writing machines, which elapsed some decades in Ethiopia. Thus, as declared in the previous paragraph, finding a scribe to write the training as well as test document was a big challenge. Using actual manuscripts was not also possible for testing due to its requirement of various preprocessing steps like robust noise removal; repair of degraded images and the effect of old age deprecation on the manuscripts. These tasks could not be covered due to the time constraint while conducting this study.

Since the research is dealing with pattern recognition, or more specifically characters pattern recognition, the data collected are of two types. The first being data for training the neural network designed and created as a recognition engine, the second will be for testing the performance of the recognizer. Towards this end, the approach followed in this work is to have the character set for the 33 basic and 6 forms of the basic character written by the scribe for the purpose of training.

It is described in section 2.4.2 that neural network requires as many sample data as possible to come up with a system having good generalization ability. There are three dominant reasons for not having large training sample written by professional scribes for teaching the neural network for our case. These are:

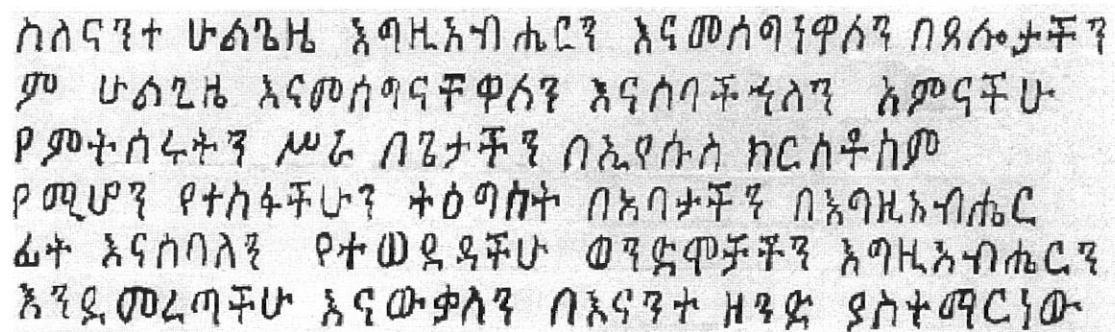
1. As described above for the case of “Yekum Tsifet”, finding many scribes and produce handwritten text documents was not possible.
2. Finding the 231 characters (the 33 basic and 6 forms of each) in real manuscripts was not possible as tried to look for the manuscript found at the Institute of Ethiopian Studies, AAU.
3. Extracting all the characters from one manuscript was not possible one due to the very noisy nature of the real manuscripts and the other since finding the whole character set in one document was not possible.

Thus, what is used to train the network is the method of generating characters with different noise level. The level of noise generated was restricted for 0.2 and 0.3 levels of random Gaussian noise as going for more noise show totally different pattern for same character and a level less than 0.2 has no significant effect on the character.

Using this method of random sample pattern generation, the data to train the network was created on the fly as the network trains. While generating patterns for testing the recognizer, two approaches were adopted. The first approach is to introduce any random Gaussian noise to the original character and test the

performance of the recognition engine. The other approach is that, some text document was extracted from the New Testament and given to the scribes who wrote the training character set and produce that text document for testing purpose.

Sample character set, the first two basic characters and their 6 other forms, written by one of the scribes for the training as well as for noise generation for testing and training purpose is given below. Refer to Annex-1, Annex-2 and Annex-3 for the full character sets used for training as scribed by two scribes.



ሰለናንተ ሁልጌዜ እግዚአብሔርን እናመሰግናለን በጸሎታችን
ም ሁልጌዜ እናመሰግናለን እናሰጣቸዋለን አምናቸው
የምትሰሩትን ሥራ በጌታችን በኢየሱስ ክርስቶስም
የሚሆን የተሰፋቸውን ተዕግስተ በአገታችን በእግዚአብሔር
ፊት እናሰጣለን የተወደዳቸው ወንጀሞቻችን እግዚአብሔርን
እንዲመረጣቸው እናውቃለን በእናንተ ዘንድ ያስተማርነው

Figure 8: Sample Amharic Characters as Written by a Scribe

As literature shows, characters in manuscripts are written in unconnected manner due to the grid on the skin and with uniform intensity of color. While experimenting on the training data, it should be reported that some of the characters in the test document written for testing the network performance were found to be overlapping and some with very low color intensity. In addition to that, the stroke width for characters among those written in the whole set is not found to be the same as that of similar character in the test document. This

indicates that the text document produced to test the accuracy of the recognizing engine is degraded compared to the character set produced to train the network.

4.3 Digitization

As described in section 2.2.1.1, 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, the hand written documents produced by the scribes are scanned and prepared for further manipulation.

As digitization is the process of converting a printed or handwritten document to an electronic format, the handwritten “Yekum Tsifet” documents are scanned. The scanning device used is Power Scan 1220 ST flat-bed scanner with MiraScan driver. The scanning parameter for the documents was of 300 dpi (density per inch for vertical as well as horizontal dimensions) and with grayscale color format. The image is prepared using Adobe Photoshop Image Processor and saved as a bitmap (.BMP) image format.

The resulting bitmap images are then put as a resource in Microsoft Visual C++.Net project folder. These images then are processed using different preprocessing algorithms. The following sections describe the experiments done on the digitized documents inside the Visual C++.Net Programming environment.

4.4 *Binarization*

Binarization, which is the task of discriminating the digitized document pixels into foreground and background image, is performed immediately after the image is loaded. Research shows that there are various methods that can be used to binarize digitized documents. As explained in section 2.2.1.2, the binarization or thresholding methods available could be either global or adaptive. For the case of this problem domain methods from both approaches were tested.

For the first method, which is adaptive thresholding, one of the algorithms, named Niblack algorithm was tested and found to be inefficient for the problem at hand. The Niblack algorithm works in such a way that local minimum threshold value is computed for each pixel by consulting the neighboring pixels. The algorithm uses the mean and the standard deviation of color intensities of the pixels in the vicinity of the target pixel. This algorithm is found to affect the basic features of the character while trying to isolate the character pixels from background pixels.

The second method, which is global thresholding as described in chapter two, will select one threshold value to work for all pixels in the image. The appropriate global threshold value (which is `color < 14000000` where 'color' is a 'COLORREF' variable in Visual C++.Net) is selected after repetitive experimentation by consulting the output binary image. In this approach, all the pixels from the input image will be mapped to output image either as foreground (white) or background (black) image in one pass through the image. This method is found to be more effective in isolating the text pixels from the image.

ለሰናፃተ ሁልጌዜ እግዚአብሔርን እናመሰግናለን በጸሎታችን
ም ሁልጌዜ እናመሰግናቸዋለን እናሰግባቸዋለን አመናችሁ
የምትሰሩትን ሥራ በጌታችን በኢየሱስ ክርስቶስም
የሚሆን የተሰፋችሁን ተዕግስተ በክብታችን በእግዚአብሔር
ፊት እናሰግላለን የተወደዳችሁ ወንጀዎቻችን እግዚአብሔርን
እንዲመረጣችሁ እናውቃለን በእናንተ ዘንድ ያለተማርነው

Figure 9: Sample Binarized Image

To illustrate the experiment at this level in depth, the property of the writing style is worth mentioning. In “Yekum Tsifet”, since all the characters are written in bold and heavy ink, unless and otherwise the noise is very visible, the binarization method used for noise removal can be relaxed. In addition to the intensity, by considering the fact that each Amharic character has fully connected strokes (except seven special characters, which are not considered in this work since they are not among the basic symbols) one can also use the strategy of analyzing the connectedness of the stroke with the other parts of a character, which is not covered in the current work.

After the binarization, or sometimes called thresholding, the scanned image will have the feature that the foreground image will have a black color and a background image will get hold of white color. This output of the binarization module will be fed to the next level of processing, which is image segmentation.

4.5 *Image Segmentation*

In OCR systems, as stated in the previous sections, image parts of interest to the recognizer, which could be sentences, word, character or any type of stroke, should be isolated. For the character recognition problem domain, the point of interest or the low level of 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.

After the binarization process is done, each logical line of the text is extracted by constructing the horizontal histogram. During this process, a top and bottom line of each logical text line is identified and marked with red color. This process of line segmentation uses some experimentally attained threshold value to declare the top and bottom of one text line. During experimenting the line segmentation phase, a threshold value of five (5) for the number of black pixels in a row is found to work well to isolate text lines. After getting done with the line segmentation, character segmentation took place.

To illustrate the performance of the stage-by-stage segmentation algorithm the module as well as the output of the procedure is presented below.

```

bool TOP=FALSE, BOT=FALSE;
int BotLine, TopLine, count;
for (i=0; i<=bm.bmHeight; i++)
{
    count=0;
    for (j=0; j<=bm.bmWidth; j++)
    {
        color= dcimage.GetPixel(j, i);
        if (color==m_Black)
            count++;
    }
    if ((count<5) && (!TOP))
        for (j=0; j<=bm.bmWidth; j++)
            dcimage.SetPixel(j, i, m_White);
    if (count>=5)
        if (!TOP)
        {
            for (j=0; j<=bm.bmWidth; j++)
            {
                dcimage.SetPixel(j, i-1, m_Red);
                TOP=TRUE;
                TopLine=i-1;
            }
        }
    if (count<5)
        if (TOP)
        {
            for (j=0; j<=bm.bmWidth; j++)
            {
                dcimage.SetPixel(j, i, m_Red);
                TOP=FALSE;
                BotLine=i;
            }
        }
}

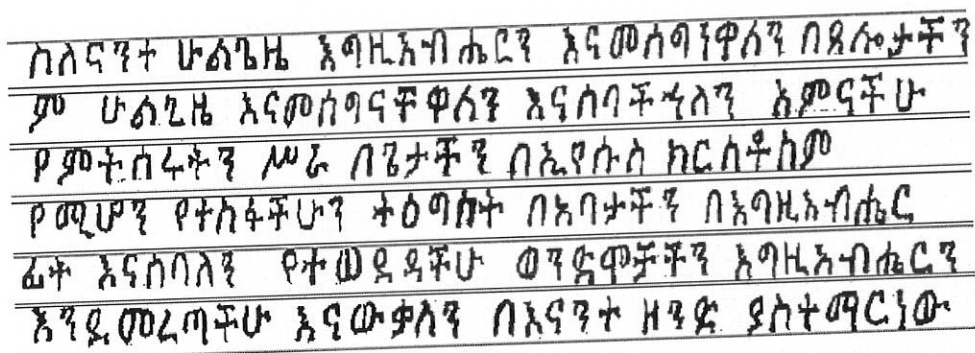
```

Figure 10: C++ Code for Line Segmentation

Starting from pixel found at the top first row, the procedure count the number of black pixels in each row. If the number of black pixels in one row is more than the specified number (which is set to be five black pixels for the binarized image) and if top of one line is not reached, that line of the image is taken as the top line of that specific text line. Otherwise, if the top line is found and the number of black pixels is less than the specified value (which is again five), then that line of pixels is taken as the bottom line of a specific text line. After finding the row number of the top and bottom line, the procedure will draw a red line bounding the text line. The number of black pixels in a line used to discriminate logical text lines is selected to five after different values are tested for efficiency.

The binarized image is presented to this module and the output of the procedure turn out to be very good.

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, "Yekum Tsifet" is written on the writing material after grid is formed on the skin that helps all characters in one line to fall on the supposed area. For the experimentation as well as test, five handwritten text pages were produced by scribes, and 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%.



ስለሰላም ሁልጊዜ እግዚአብሔርን እናመሰግናለን በጸሎታችን
 ም ሁልጊዜ እናመሰግናለን እናሰጥላለን አምናችሁ
 የምትሰጡን ሥራ በጌታችን በኢየሱስ ክርስቶስም
 የሚሆን የተሰጠላችሁ ተወካዮች በክብራችን በእግዚአብሔር
 ፊት እናሰጣለን የተወደዳችሁ ወንጌላቸውን እግዚአብሔርን
 እንዲመረጣችሁ እናውቃለን በእናንተ ዘንድ ያለተማርነው

Figure 11: Sample Output of the Line Segmentation with Horizontal Projection Profile

After the lines of text found in the image are identified and its top and bottom row is marked with red color, the whole raster image is scanned to segment each character. For this purpose, as the stage-by-stage segmentation algorithm proceeds, each logical line will be identified and the vertical projection profile will be consulted to segment each character.

```

for(i=0;i<=bm.bmHeight;i++)
{
    color= dcimage.GetPixel(0,i);
    if((color==m_Red)&&(!TOP))
    {
        TopLine=i;
        TOP=TRUE;
    }
    else if((color==m_Red)&& TOP)
    {
        BotLine=i;
        BOT=TRUE;
    }
    if(TOP&&BOT)
    {
        for(k=0;k<=bm.bmWidth;k++)
        {
            count=0;
            for(m=TopLine+1;m<BotLine;m++)
            {
                color= dcimage.GetPixel(k,m);
                if(color==m_Black)
                    count++;
            }
            if((count>0)&&(!Left))
            {
                for(int l=TopLine;l<=BotLine;l++)
                    dcimage.SetPixel(k-1,l,m_Red);
                Left=TRUE;
                LeftLine=k-1;
            }
            if((count==0)&&(Left))
            {
                for(int l=TopLine;l<=BotLine;l++)
                    dcimage.SetPixel(k,l,m_Red);
                Left=FALSE;
                RightLine=k;
            }
            //Creating a linked list of objects identified as a character from the image
            Current=new Fedeloch;
            Current->StartPoint.x=LeftLine+1;
            Current->StartPoint.y=TopLine+1;
            Current->EndPoint.y=BotLine-1;
            Current->EndPoint.x=RightLine-1;
            Current->NextFedel=NULL;
            Current->PrevFedel=NULL;
            if(!Start)
                Start=Current;
            else
            {
                Temp=Start;
                while(Temp->NextFedel!=NULL)
                    Temp=Temp->NextFedel;
                Temp->NextFedel=Current;
                Current->PrevFedel=Temp;
            }
            //End of creating the linked list
            if((count==0)&&(!Left))
            {
                dcimage.SetPixel(k,TopLine,m_White); //cleaning the red top and bottom
                dcimage.SetPixel(k,BotLine,m_White); //lines between characters set
            }
            during
        }
        //if the width of the character is less than 7 pixels delete that image area
        if((RightLine-LeftLine)<7)
        {
            for(j=LeftLine;j<=RightLine;j++)
            {
                for(int l=TopLine;l<=BotLine;l++)
                    dcimage.SetPixel(j,l,m_White);
            }
            TOP=FALSE;
            BOT=FALSE;
        }
    }
}

```

Figure 12: C++ Code for Character Segmentation

After presenting the line-segmented image to this module, the procedure will move vertically and look for two consecutive red pixels to indicate the top and bottom lines of a text line. Subsequent to identifying these two rows, vertical projection profile will be constructed within these two rows and a red line will be drawn at the right and left border of each character. This procedure also showed good result except that some few characters with disconnected strokes are regarded as one unique one. And some pair of characters are found to be considered as one.

After character segmentation module is executed, Figure 11 (the above figure showing line segmented image) will have the following pattern.



Figure 13: Sample Output of the Character Segmentation with Veridical Projection Profile

Overlapped characters and disconnected strokes are the two problems observed in the course of isolating each character from the text line. Most of the overlapping

occurs between the character “” and any other coming before it where both belong to same word. This scenario occurs since the scribes didn’t stop the stroke, left leg, of the character to be in its own block, which is expected in handwritten documents. The other character with frequent overlapping and as well detached stroke is “”. The top left stroke of the character found to overlap with the previous character or just broken from the original character.

Figure 14 and Figure 15 will give a visual impression on some of the erroneous outputs:

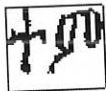


Figure 14: Sample Overlapped Characters

Figure 14 shows the case where, two characters are considered as one since the leg of the second character cross the border of the first character.



Figure 15: Sample Detached Character

Figure 15 shows the other problem where, the stroke to the left of the character is taken as another character since there is white space between the strokes.

For both training and test cases, a total of 2,254 characters written by two scribes were fed to this stage-by-stage character segmentation module. From these sets of characters, the procedure is capable of segmenting 2,163 characters correctly and

the rest are either taken together with another character or portion of it is detached. Thus the rate of the character segmentation turns out to be 95.96%, which is remarkable for handwritten text processing.

Among those wrongly segmented characters, 33 of them are found to be overlapped and one character which is “7” has a detached stroke 13 times. The total wrongly segmented characters in addition to the above are calculated to be 91 ($38 \times 2 + 15$). This value is found to be 4% of the total character set fed to the character segmentation module.

These two problems found to propagate the inconvenience to the subsequent algorithms commanding a negative impact on the overall recognition. The binary representation produced for these two sets 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.

4.6 Character 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, will accept 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 exists 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 Visual C++.Net environment is used to do the size scaling. 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.

A listing that performs size normalization to transform some selected image area to a 20 by 20 pixel size is presented below. The same procedure can be used to transform image area to any target size by modifying the output size on the listing.

```
int check=1;
Temp=new Fedeloch;
Temp=Start;
int X=5,Y=5;
while(Temp!=NULL)
{
    if(check==0)
    {
        for(j=Y;j<Y+35;j++)
            for(i=0;i<=bm.bmWidth;i++)
                dcimage.SetPixel(i,j,m_White);
        check=1;
    }
    //the inbuilt function to do the scalling (20x20, for our case)
    dcimage.StretchBlt(X,Y,20,20,&dcimage,Temp->StartPoint.x,
        Temp->StartPoint.y,(Temp->EndPoint.x-Temp->StartPoint.x),
        (Temp->EndPoint.y-Temp->StartPoint.y),SRCCOPY);

    dc.BitBlt(0,0,bm.bmWidth,bm.bmHeight,&dcimage,0,0,SRCCOPY);

    pOldBitmap=dcimage.SelectObject(&m_bitmap);
    dcimage.SelectObject(pOldBitmap);

    Temp->StartPoint.x=X;
    Temp->StartPoint.y=Y;
    Temp->EndPoint.x=X+19;
    Temp->EndPoint.y=Y+19;
    X=X+35;
    Temp=Temp->NextFedel;
}
```

Figure 16: C++ Code for Character Size Normalization to a 20x20 Representation

After the segmented characters are fed to the size normalization module, each character is set to have standard character width and height. The result corresponding to the image at **Error! Reference source not found.** with a 20x20 size normalization is shown below.

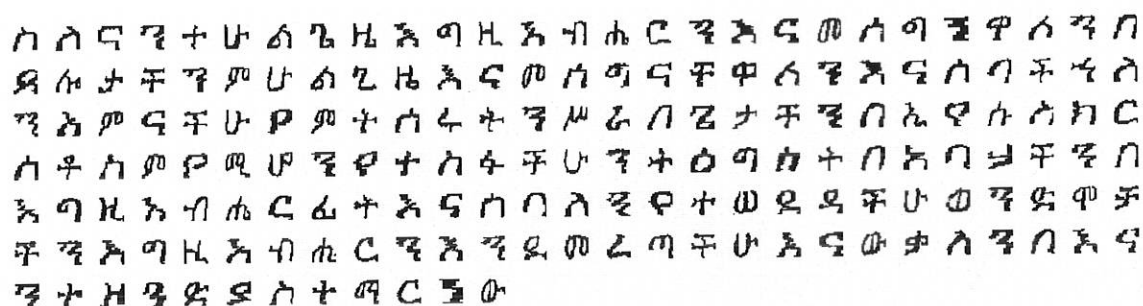


Figure 17: Sample Normalized Characters

As can be seen from the sample output, characters with small character width are stretched and will have a deformed morphology. As in the case for characters with smaller width, characters with larger width are also deformed morphologically. Sample output for character with smaller width is presented below where the first character is morphologically deformed after transformed to a 20x20 pixel size; the effect is also observed to be the same for a 16x16 character representation.



Figure 18: Morphologically Deformed Character after Normalization

Like wise, disconnected character strokes are also normalized by the module taken as one unique character. Sample normalization output for such patterns is presented below.



Figure 19: A Deformed Character and a Noise Due to Detached Stroke

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

4.7 *Creating Character Representation*

After the characters are normalized 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 top left coordinate of each character is identified and stored during character size normalization for further processing. For each character, starting from the top left coordinate of the image, the bitmap area will be traced horizontally for the standard character width, which is either 20 pixels or 16 pixels for the normalized image. During tracing, if a pixel is with color white a zero will be written on the text file intended to store the binary representation of the characters. The value to be written on the file will be one if the pixel color is found to be black.

This process will continue for each row of a character creating the same pattern on the text file holding the representation of the characters found in the scanned and processed image. The text file is organized in such a way that it will have twenty (or sixteen for the 16x16 case) elements in each line and the number of rows or lines will be the number of characters in the document multiplied by the character height (which is 20 pixels or 16 pixels).

The module that is developed in the Visual C++ programming environment to do this conversion is presented as follows.

```

FILE *Binary=fopen("Binary.txt","w");

int B=1,W=0;
Temp=new Fedeloch;
Temp=Start;
while(Temp!=NULL)
{
    for(j=Temp->StartPoint.y;j<=Temp->EndPoint.y;j++)
    {
        for(i=Temp->StartPoint.x;i<=Temp->EndPoint.x;i++)
        {
            color=dcimage.GetPixel(i,j);
            if(color==m_Black)
                fprintf( Binary, " %d", B );
            if(color==m_White)
                fprintf( Binary, " %d", W );
        }
        fprintf( Binary, "\n", "" );
    }
    Temp=Temp->NextFedel;
}
fclose(Binary);

```

Figure 20: C++ Code for Creating Binary Representation of Bitmap Character Image

Sample binary representation for the first and third characters of the test document written by one of the scribe and processed by the above procedures is presented as follows.

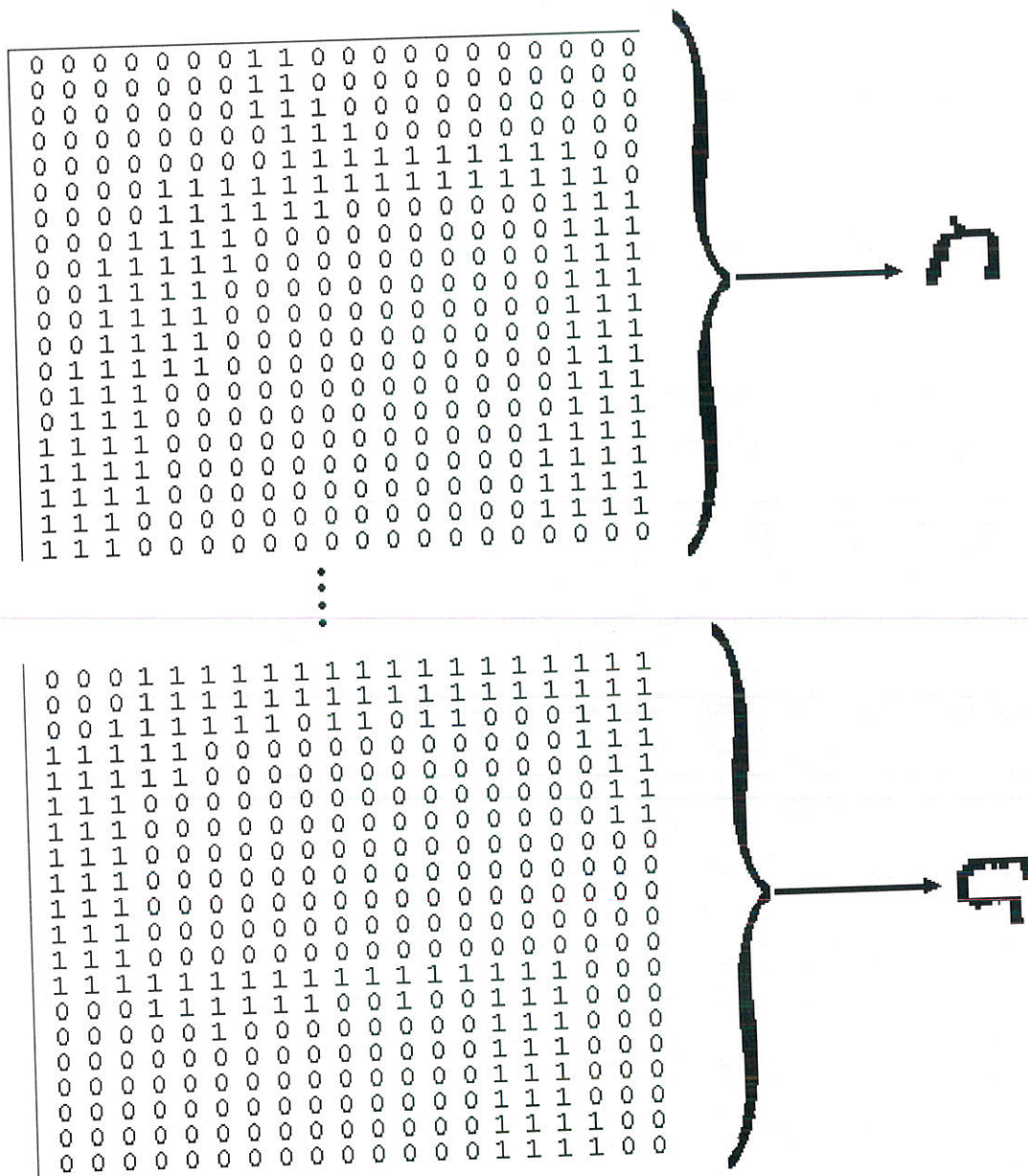


Figure 21: Sample Binary Representation for two Characters

4.8 Character Recognition

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, create, train and test the network will be in the domain of the tool and its corresponding functions.

4.8.1 Artificial Neural Network Architecture

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 two normalization sizes used for further consideration are the 20x20 and 16x16 matrix of zeros and ones representing the characters in the test and training documents.

The following graphical presentation will give 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.

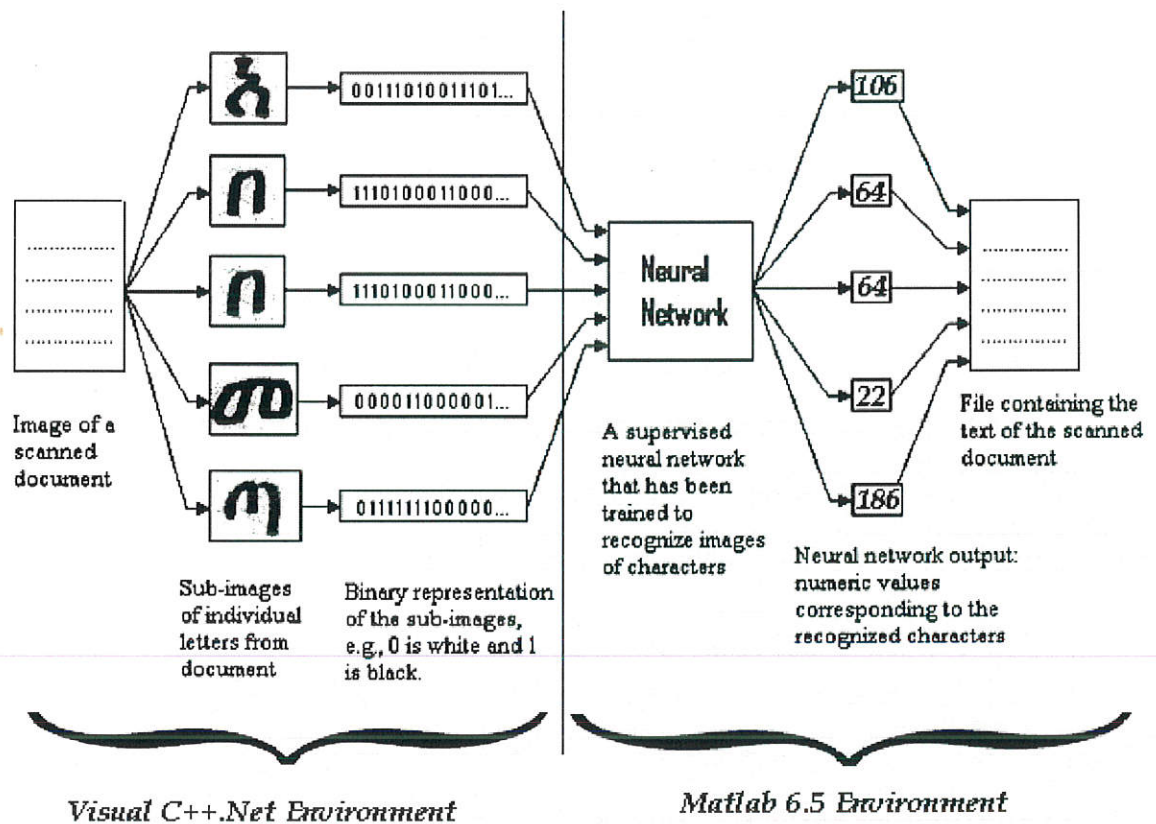


Figure 22: General Architecture of the Recognition Network

(Original image from: <http://www.nd.com/neurosolutions/products/ns/whatisNN.html>)

It is described in section 4.7 that each character identified in the raster image will be converted to its equivalent binary representation after applying some preprocessing on the image. This representation will be used for training and testing the network after the designing and development of the neural network, which will be discussed next.

4.8.2 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

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 stuck 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.

```
YekumTsifetNetwork = newff(input_range, [40 8], {'logsig','logsig'},'traingdx');
```

Where:

input_range; is the minimum and maximum values at the input nodes

[40 8]; is size of neurons at hidden and output layer respectively

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

'traingdx'; is the learning function of the network

4.8.5 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. As finding many scribes to collect large training sample as well as testing data set is claimed to be difficult due to the set of reasons mentioned in section 4.2, another approach is used while training the network in addition to text produced by two scribes

The approach used is that, one can generate as many random noise patterns to be imposed on the original binary representation of each character as required. This method will open an opportunity to the network to face different patterns of same character. This helps to enhance the generalization power of the network when the system faces with new patterns. Sample of randomly generated noisy character representations for the first character from the character set at different level of variance is displayed below. The result reveals that the character won't lose its major morphology but will have some noise introduced to it.

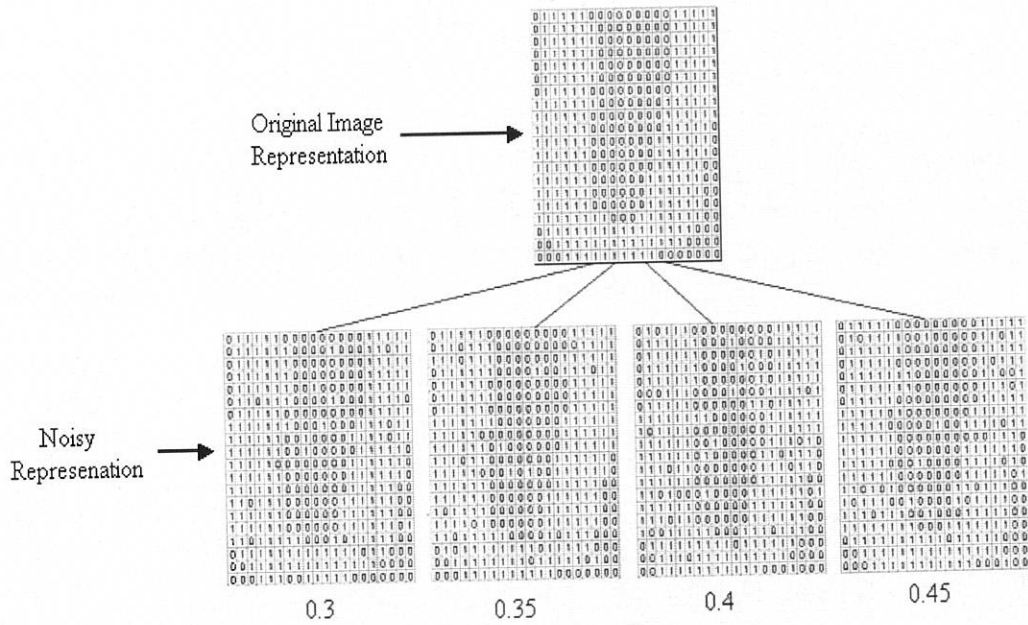


Figure 23: Sample Character Representations with its Various Noisy Level

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:

[YekumTsifetNetwork,tr] = train (YekumTsifetNetwork,Fedel,Fedel_Target);

Where:

YekumTsifetNetwork; is the network created using the instruction in section 4.7

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

Fedel_Target; is pattern of target output for each corresponding input

As described in the previous sections, finding large training data set for the case of “Yekum Tsifet” is very difficult. The real data is found on animal skin that is very noisy by its very nature and beyond the scope of this work as character tracing

intended to be done on white papers. Thus, as mentioned above, the systematic approach used in this work is the introduction of random Gaussian noise to each character pattern during training. To have different levels of noise, the intensity of the noise imposed on each character is varied to some level.

The line of command written to train the network with randomly generated noisy character pattern is presented below.

```

T = [Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target
Fedel_Target Fedel_Target];

for pass = 1:100
    P = [Train1, Train2, Train3, ...
        (OZMatrix(Train1 + randn(N,M)*0.2)),...
        (OZMatrix(Train1 + randn(N,M)*0.3)),...
        (OZMatrix(Train2 + randn(N,M)*0.2)), ...
        (OZMatrix(Train2 + randn(N,M)*0.3)), ...
        (OZMatrix(Train3 + randn(N,M)*0.2)), ...
        (OZMatrix(Train3 + randn(N,M)*0.3))];
    [YekumTsifetNetworkn,tr] = train(YekumTsifetNetworkn,P,T);

```

Figure 24: Neural Network Training with Noise

where:

$\text{randn}(N,M)*0.x$ will create a random noise of size $N \times M$ with $0.x$ intensity.

Here $N \times M$ is the size of the input matrix. Where N is the number of row bits representing each character and M is the number of characters (231)

For training the network with different patterns of the Amharic character set, the whole set is made to be written by two located scribes. Actually, one scribe is made to produce another copy of the character set that makes up the number of pages containing the whole character set to be three. This is done due to the fact that writers will put somewhat different layout for same character when painting it more than once. This will help the network to encounter even more different patterns.

For training the network, one copy of each of the three original character patterns with two different copies of the pattern with different level of noise is fed to the network. To increase the efficiency of the network, providing large sample is important. To attain this goal the above training module is repeatedly executed many times which will generate different random noisy patterns and train the network with it. For the problem at hand, the network is trained with the above input for 100 iterations, which will make the total number of patterns of each character provided to the network to be 900. For the whole character set, a total of 207,900 (231×900) patterns are passed to the network for training.

As pointed out in section 4.8.2, the experimentation is done on two sets of categories. The first is for characters sized to a 20×20 and the other for 16×16 character representation. Thus, two neural networks are created for the two cases and the training as well as test characters sets are sized to both 20×20 and 16×16 representations.

After creating the two networks and training each with the training samples available and as well with the generated noisy data, the performance is tested on

two page Amharic text written by the two scribes. The text is extracted from the New Testament and provided to the scribes with a computer printout.

4.8.6 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 will help 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 will help to see how powerful the network is in generalization when faced with new character patterns. Thus, test with patterns not included in the training session can be used to measure how the network will act for new patterns with and without noise.

To create a match-up between the output and the target values, the 231 characters are assigned a number 1 up to 231 starting from the first character. The value of each cell in the eight bit target matrix is an 8-bit binary equivalent of the decimal number assigned for the character. Thus, for all the test cases what the network does is; it will provide an eight bit sequence of zeros and ones. This eight bit value will be converted to its decimal equivalent value and the character with that decimal number representation will be taken as an output of the network for that specific input pattern. Thus the testing of the network created and trained at sections 4.8.4 and 4.8.5 is tested using the following code.

$$Y = \text{sim}(\text{YekumTsifetNetwork}, \text{Test});$$

Where:

YekumTsifetNetwork; is the network created and trained for different patterns, and

Test; is the testing input pattern

Y; is the output of the network

The two artificial neural networks created for two different numbers of input nodes are tested for accuracy of classifying new character pattern to one of the 231 characters of the Amharic language. The output of the network could range between 0 and 256 since it is an 8 bit (2^8) pattern. For the case where the network output is between 232-256 (which is the range that does not belong to the target values of the 231 characters), the output will be declared as unknown, otherwise the best approximation is taken as its value.

Test using the training set

To see whether the network can recognize the character set used for training, all the three pages that contain the whole character set is fed to the network. The output of the two networks is presented in the following table; the total number of characters for the testing is then 231 for each case since each contains all the characters under consideration.

Network	Performance /Rate of Recognition		
	Training Set 1	Training Set 2	Training Set 3
With 256 Input nodes	98.70%	96.54%	96.10%
With 400 Input nodes	96.97%	97.84%	95.24%

Table 3: Network Performance for the Training Set

Test using the training set with noise

The three training set with different level of noise are also fed to the network to see its performance on noisy data set. The test is done for both networks to see the efficiency of both networks. The results of the three test sets with three different levels of noise are presented below.

Test Set	Network With 256 Input nodes			Network With 400 Input nodes		
	0.2 noise	0.25 noise	0.3 noise	0.2 noise	0.25 noise	0.3 noise
Training Set 1	92.21%	58.87%	32.03%	93.51%	70.13%	46.75%
Training Set 2	87.88%	56.28%	35.06%	94.81%	68.40%	38.53%
Training Set 3	86.15%	54.55%	27.27%	93.07%	64.07%	35.50%

Table 4: Network Performance for Training Set with Noise

Test using new test case

After the network is tested for the set of characters it is trained with, (with and without noise), the network is also tested with new text written by the scribes. This test will help to see the response of the network to new character input patterns that the network never faced during the training phase. The set of characters for the two test pages are 717 and 739 respectively. Thus both networks are tested for a total of 1,456 new character patterns. The recognition rate of the network is presented in the following table.

Network	Performance /Rate of Recognition	
	Test Set 1	Test Set 2
With 256 Input nodes	16.88%	21.92%
With 400 Input nodes	16.18%	31.53%

Table 5: Network Performance for New Test Set

As can be observed from the three set of testing scenarios, the network behaves remarkably when provided with character set it is trained with. For the case of characters with random noisy patterns imposed on the training set, the network performance decreases with the increase of noise level forced on the original character. This is expected since adding more noise means distorting the morphology of the character at great level. For the case of the new test patterns the network accuracy is not bad considering the fact that it is handwritten document and the training is done on small samples.

While analyzing the result of the network output, it is found that the characters $\mathfrak{R}$ and $\mathfrak{E}$, characters $\mathfrak{h}$ and $\mathfrak{X}$, and characters $\mathfrak{N}$ and $\mathfrak{A}$ to be recognized as the other and the families of the Φ character are found to be more similar for network as counter families of the Ψ character.

Comparing the two networks for the performance on different test cases, the network with 256 input nodes is found to be more effective in recognizing the set of character patterns used for training. For the other two cases, noisy training patterns and new test samples, the recognition rate of the network with 400 input nodes is found to be more efficient.

And finally, the performance tables (Table 3, Table 4 and Table 5) reveal that, the network with 400 input nodes perform in a better manner than the one with lesser input layer nodes for new patterns. Thus resizing the characters to a dimension of 16x16 cannot be considered as a loss less transformation since the recognition rate is found to be affected with the size.

4.9 Conclusion

In this section the major activities performed during experimentation were discussed. The portion also discussed the implementation of different algorithms with the coding and sample graphical representation of the attained result with the problems faced at each stage.

Generally, the whole process from data collection and compilation to character recognition is presented. The character segmentation phase showed that the line segmenter works with no error and the character isolator perform with 95.96% efficiency. Moreover the size normalization functions available in Visual C++.Net works well for size scaling. And the performance of the network for each training and test set reveals that the network is capable of recognizing a good many of the character patterns used for training and noisy training samples. The two networks are also able to recognize 16.1% up to 31% of characters in the test sets produced by different scribes.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Introduction

In this study an attempt was made to recognize a handwritten Amharic text written using a special kind of writing style, which traditionally is called “Yekum Tsifet”. The writing system was used as a dominant knowledge preserving and correspondence method for long in Ethiopia. For the course of attaining a good recognition rate, various preprocessing algorithms were reviewed, modified and used. The major tasks performed will be discussed next and this section will be winded up by forwarding set of recommendations for future research focus.

5.2 Conclusion

Handwriting has been and partly is a dominant way of communication and knowledge preserving method in our day-to-day life. The fallback in this method of communication is that; the content is not in a format to be processed by technologically efficient and powerful text processing software. In general it is not in its computer readable format. A wide variety of research has been done to convert texts and pictures in printed form to a computer readable format. One wing of this research, which is called optical character recognition, is dealing only with characters that are used to represent sounds of a language.

Character recognition implements a wide range of approaches to recognize most if not all of the character images. The problem domain is more difficult when the interest is in recognizing handwritten characters. Most researchers agree that a

good recognition rate will be attained if the domain of the problem is narrow to some extent. For the case of handwritten character recognition, one can narrow the problem by restricting the number of writers and the set of character sets to be recognized. This study used the former method due to the rare number of scribes.

In this study, different preprocessing algorithms like global thresholding, stage-by-stage segmentation, normalization 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 global thresholding method for binarizing the digitized image is found to work very well instead of adaptive and local thresholding methods for the problem of interest.

To isolate each character from the digitized and binarized image, there were methods like recognition-based approach, which integrate the recognition and segmentation tasks side by side. The approach sends some portion of a line to a recognizer and resizes the current image based on the output of the recognizer. This kind of approach is robust for cursive writing. Since the writing style of Amharic language is not cursive in principle and more true for “Yekum Tsifet”, stage-by-stage segmentation algorithm, which isolates text lines, words and the characters iteratively, is implemented. The method is found to segment up to 95.96% of the characters from the document provided for training and testing purpose.

As artificial neural network is used for the recognition task, it would be simpler if input patterns have same size to create a generalized network. For this purpose, the characters need to be resized to some uniform size. In the process of scaling each character image, different size normalization techniques were analyzed. From all those, the one described by Pandya and Macy [1996], Run Length Coded method has been tried out and as pointed out by the author, problem of round-off error affected the images and found to be computationally time consuming. To this end, the image resizing facility available in the Visual C++.Net programming environment was used and found to be very good. In the course of the size normalization process, characters with minimum width (like '፲') stretched horizontally too much creating a deformed morphology and also characters with wider width (like '፳') found to be compressed too much.

For classifying different input patterns as one of the 231 Amharic characters, an artificial neural network approach is implemented. The network developed is with feedforward 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.

The network found to recognize most of the patterns it is trained with and some of new test patterns. A performance rate that ranges between 98.7%-95.24% is attained for the former testing and recognition rate between 16.1% and 31% is observed for new test document. The network is also tested with original training pattern with different level of random noise imposed on it and a recognition rate

ranging between 94.8% and 27.27% is obtained for different noise levels by both networks.

Based on the result of the recognition rate, the network with 400 input nodes is found to work better than that of with 256 nodes. This could be due to the fact that compressing the image to a 16x16 pixel dimension may deform the morphology of the character.

As the study conducted is partly concerned for the academic experience, facilities like user interface and integration between the two programming environments is not done.

The experimentation showed that there is an opportunity to deal with Amharic handwritten noisy data, especially those written on manuscripts. Equipping this experiment with additional preprocessing steps such as thinning, slant correction and skew adjustment could enhance the recognition rate of the engine to a higher level.

5.3 Recommendation

Inclusion of more preprocessing steps to the system will obviously enhance recognition performance. Moreover, training the network with more real training samples will also have a positive effect in the generalization power of the network. Towards this end, the following sets of recommendations, which require further investigation, are forward.

1. More assessment on real manuscripts and text retrieval for such documents must be sorted out to make use of the OCR systems to digitize the manuscripts.
2. Additional preprocessing algorithms like thinning, noise removal, slant correction and the likes should be considered.
3. The network must be exposed to more training samples to increase the knowledge acquired by the network, which will result in better performance.
4. Other special characters not in the 231 basic set and numerals should be included to make the recognizer applicable for any text document written with the writing style.
5. Word and higher level of recognition techniques, which use semantic and cognitive approach, should be considered for better results.

REFERENCES

- Assefa Liban; "***Parchment Preparation***", Addis Ababa, Ethiopia, 1978
- Bender, M.L. et al. (ed); "***Language in Ethiopia***". London, Oxford University Press.1976
- Berhanu Aderaw; "***Amharic Character Recognition using Artificial Neural Networks***", (Masters Thesis). Addis Ababa: Department of Electrical Engineering, Addis Ababa University, 1999.
- Blumenstein, B.V.M. & Kulkarni, S.; "***Recent achievements in off-line handwritten recognition systems***", International Conference on Computational Intelligence and Multimedia Applications, Melbourne, Australia, 27-33., 1998
- Brown E.W.; "***Character Recognition by Feature Extraction***" N.E. University, 1992
Internet: <http://www.ccs.neu.edu/home/feneric/charrecnn.html>.
- Burr D. J.; "***Experiments on neural net recognition of spoken and written text***",
IEEE Trans on ASSP 36, No. 7, 1988.
- Casey, R.G. and Nagy, G; "***Recursive segmentation and classification of composite character patters***", International Conference on Pattern Recognition. V2 ,1982 pp 1023-1026
- Dereje Teferi; "***Optical Character Recognition of Typewritten Amharic Text***".
(Masters Thesis). Addis Ababa, School of Information Studies for Africa, Addis Ababa University, 1999.
- Encarta Encyclopedia 2001; "***Optical Character Recognition***"; Microsoft Encarta Encyclopedia. Microsoft Corporation.
- Ermias Ababe ; "***Recognition of Formatted Amharic Text Using Optical Character Recognition***". (Masters Thesis). Addis Ababa, School of Information Studies for Africa, Addis Ababa University, 1998.
- Gray, P.J. ; "***Optical Scanning, OCR, and MICR***"; Automatic Data Processing Handbook, New York; McGraw-Hill Book Company, pp 2-117 to 2-125, 1977
- Green, W. B.; "***Introduction to Electronic Document Management System***";
Academic Press, Inc., Boston, 1993.
- Haigh, Susan; "***Optical Character Recognition (OCR) as a Digitized Technology***";
Information Technology Services, National Library of Canada, 1996.

- Hammerstrom Dan; "**Neural Networks at Work**"; Journal of the International Neural Network Society, IEEE Transactions on Neural Networks, and the International Journal of Neural Networks, 1993.
- Hopfield J.J.; "**Neural Networks and Physical Systems with Emergent Collective Computational Abilities**", Proc. Natl. Acad. Sci. USA 79, 2554, 1982.
- Jun Fan; "**Off-line OCR for Printed Chinese Characters: A Survey**"; Columbia University, Electrical Engineering. Accessed in October 2003.
- Kahan S., Pavlidis T., and Baird H.S.; "**On the Recognition of Printed Characters of Any Fonts and Size**", IEEE Transactions on pattern analysis and machine intelligence, Vol. 9, No. 2, pp. 274-288, Mar. 1987
- Kefyalew Merahi; "**The Contribution of Ethiopian Orthodox Church to Ethiopian Civilization**", Addis Ababa, 1987
- Mesay Hailemariam; "**Handwritten Amharic Character Recognition: The case of Postal Addresses**", (Masters Thesis). Addis Ababa, Department of Information Science, Addis Ababa University, 2003.
- Million Meshesha; "**A Generalized Approach to Character Recognition of Amharic Texts**", (Masters Thesis). Addis Ababa, School of Information Studies for Africa, Addis Ababa University, 2000.
- Mori, S., Nishida, H. & Yamada, H.; "**Optical Character Recognition**", John Wiley & Sons, Inc. New York, 1999.
- Nigussie Tadesse; "**Handwritten Amharic Text Recognition Applied to the Processing of Bank Checks**", (Masters Thesis). Addis Ababa, School of Information Studies for Africa, Addis Ababa University, 2000.
- Pal, U. and Chaudhuri, B.B.; "**Computer Recognition of Printed Bangla Scripts**"; International Journal of Systems Science. v12, n21 pp36-45, 1995
- Pandya A.S and Macy R.B; "**Pattern Recognition with Neural Networks and C++**", CRC Press LLC, 1996
- Pujari, A. K. et al; "**An Adaptive Character Recognizer for Telugu Scripts using Multiresolution Analysis and Associative Memory**", University of Hyderabad, INDIA, 2002
- Senior W. and Robinson A. J.; "**An Off-Line Cursive Handwriting Recognition System**". IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3):309-321, 1998.

- Seong-Whan Lee, Dong-June Lee and Hee-Seon Park; "***A New Methodology for Gray-Scale Character Segmentation and Recognition***" *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 18, No. 10, October 1996.
- Sergew Hable-Selassie; "***Bookmaking in Ethiopia***", Lei den, Netherlands, 1981
- Sergew Hable-Selassie; "***Ancient and Medieval Ethiopian History to 1270***"; Addis Ababa, 1972
- Srihari, S. N and R. K. Srihari; "***Written Language Recognition***"; NSF-CEC Joint Survey on Speech and Natural Language Processing (Chapter 3), 1996
- Srihari S.N., Srikantan G., Hong T. and Lam S.W.; "***Research in Japanese OCR***" Handbook on Optical Character Recognition and Document Image Analysis eds. P.S.P. Wang and H. Bunke, World Scientific Publishing Company, 1996.
- Simon J. C.; "***Off-Line Cursive Word Recognition***". Proceeding of the IEEE, 80(7):1150-1161, 1992.
- Tom M. Mitchell; "***Machine Learning***", The McGraw-Hill Companies, Inc., 1997
- Wang A.-B., Fan K.-C.; "***Optical recognition of handwritten Chinese character by hierarchical radical matching method***", *Pattern Recognition*, 34, pp 15-35, 2001.
- Worku Alemu; "***The Application of OCR Techniques to the Amharic Script***", (Masters Thesis). Addis Ababa, School of Information Studies for Africa, Addis Ababa University, 1997.
- Yaregal Assabie; "***Development of Versatile Character Recognition System for Amharic Text***"; (Masters Thesis) School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2002.
- Zeev, R.; "***Towards Improved Script for Amharic and other Languages***". Addis Ababa, The Imperial Ethiopian Government: Ministry of Education and Fine Arts, 1966.

APPENDIXES

Annex-1: Training set one

[illegible]

Annex-2: Training set two

[illegible]

Annex-3: Training set three

[illegible]

Annex-4: Test Case one

ሰለፍንተ ሁልጌዜ እግዚአብሔርን እናመሰግናለንና በጸሎታችን
 ም ሁልጌዜ እናመሰግናለንና እናሰባቸውንም እምናቸው
 የምትሰሩትን ሥራ በጌታችን በኢየሱስ ክርስቶስም
 የሚሆን የተሰጠታችንን ተፅዕኖት በአባታችን በእግዚአብሔር
 ፊት እናሰባለን የተወደዳችሁ ወንጀላዊነትን እግዚአብሔርን
 እንዲመረጣችሁ እናውቃለን በእናንተ ዘንድ ያስተማርነው
 ተመህርት በነገር ብቻ አይደለምና በመንፈስ ቅዱስ
 በሚደረግ ኃይል በታመነው ልመናም ነው እንጂ
 ሰለፍንተም በእናንተ ዘንድ የሆነውን እናንተ ታውቃላችሁ
 እነሆ እናንተም እኛን መሰላችሁ በመንፈስ ቅዱስ ነገሥት
 ደስታ ጋር በቡዙ መገራ የእግዚአብሔርን ቃል ተቀብላችኋልና
 በመቄደንያና በአካዲያ ሳመቱ ሰዎች ሁሉ ምሳሌ ሆናችሁ
 የእግዚአብሔርን ቃል በእናንተ ዘንድ ዘንድ ተመረው ተቀብለዋል
 ልና በመቄደንያና በአካዲያ ነቢዮችም አይደሉም በእግዚአብሔር
 ማመናችሁ በየሀገሩ ሁሉ ተሰምታለች እንጂ

የናንተንም ነገር እንናገር ዘንድ እኛ መንም ለእውደም
 ወደናንተ እንዲገባን ግዴታ ከማመልከትም እግዚአብሔርን
 ወደ ማመልከት እንዲተመለሱ እነሱ ያውቃሉና ሕይወት
 ለሆነ ለእውነተኛው አመለካከት ተገዢ ዘንድ ከውታን
 የተነሳው ልጄ ኢየሱስ ክርስቶስ ከሰማይ እንዲመጣ ተሰፋ
 እድርጋችሁ ከሚመጣውም ጥፋት አሉ ያድነናል ብላችሁ
 ወንጀላዊነትን እኛ ያደረግንላችሁን እናንተ ታውቃላችሁ
 ወደእናንተ መግባታችን በነገሩ አይደለም በፈሊጽ ለዮስ
 እንዲመታውቁት ተሰድበናል መገራንም ተቀብለናል እንጂ
 ያን ጊዜ በፈጣሪያችን አጋኝነት የክርስቶስን ተመህርት
 በቡዙ ድካም ነገርናችሁ ተመህርታችን ስሕተትና ጌጢክት
 ሸግግላም የሆነ አይደለምና

አለኛት ሁልጊዜ እግዚአብሔርን እናመሰግናለን፤ ዋለህ በ
 ጸሐታችሁም ሁልጊዜ እናመሰግናለን፤ ዋለህ እናስባችን
 ከህ እምናችሁ የመታሰባችን ሥራ በጾታችን በእያሁኑ ክር
 ከቶስም የሚሆነ የተሰፋችሁ ትዕግስት በልባችሁ በእግ
 ዚአብሔር ፊት እናስባለን የተወደዳችሁ ወገኖቻችን እግ
 ዚአብሔር እንጸመረጣችሁ እናውቃለን በእናንተ ጸሐፊዎች
 ተማርነው ሕመህርት በነገር ገብቶ አዲሳልምና በመሃፈክ
 ሕዝብ በሚደረግ ሕይወት በታመነው ልመናም ነው እንጂ ስለ
 ነገሩም በእናንተ ዘንድ የሆነውን እናንተ ታውቃችሁ እ
 ነህ እናንተም እኛን መስጋችሁ በመሃፈክ ትደክ ከሚሆንደ
 ከታ ጋር በብዙ መከራ የእግዚአብሔርን ቃል ተቀብላችኋልና
 በመቁዳና በእካዲያ ማመኑ ስሙ ሁሉ ምሳሌ ሆናችሁ
 የእግዚአብሔርን ቃል ከእናንተ ዘንድ ተመረው ተቀብለዋል
 ነ በመቁዳና በእካዲያ ብቻም አዲሱም በእግዚአብሔር
 ርሃምናችሁ በየሀገሩ ሁሉ ተሰምቷል እንጂ

የኖተንም ነገር እንኖገር ዘንድ እኛ መንም እንወድም ወደ
 ኖተን ኢንዱስትሪ ጣኔት ከመንም ለክም እግዚአብሔርን ወደማ
 ምላሽ እንደተመለሳችሁ እኛው ያውቃለን ሕይወት ለሆነ ሰኢ
 ውስጥኛው እምነት ትገዙ ዘንድ ከመታየት የተገኘው ልጅ ኢ
 የሱስ ክርስቶስ ከሰማይ እንደመጣ ተከፋ እድርጋት ሆኖ ከመጸ
 መጣውም ጥፋት እሱ ያድነናል ብላችሁ ወገድሞቻችንን እኛ
 ያደረግንላችሁን እናየታል ታውቃላችሁ ወደኛት መግባታች
 ን በክንቅ እደጸላለሁ በፈለግኩት እንደመታወቁት ተስፋ
 በናል መከራንም ተቀብለናል እንጂ ያን ጊዜ በፈጣሪዎችን እ
 ኃጽነት የክርስቶስን ትምህርት በብዙ ድካም ነገርናችሁ ት
 ምህርታችን ስሎትና ኃጢአት ሽንገላም የሆነ አደጋ ለመኖ
 እግዚአብሔር በመረመረን ወገኒልን ልማከተማር በታመነ ገንዘብ
 ብ እንደያዝ እናስተምራለን እንጂ ልቡናችንን ለመረመረን ሰኢ
 ግዚአብሔር ነው እንጂ ለውን ለውን ደስ እንደሚያልፍ አደጋ ለመኖ

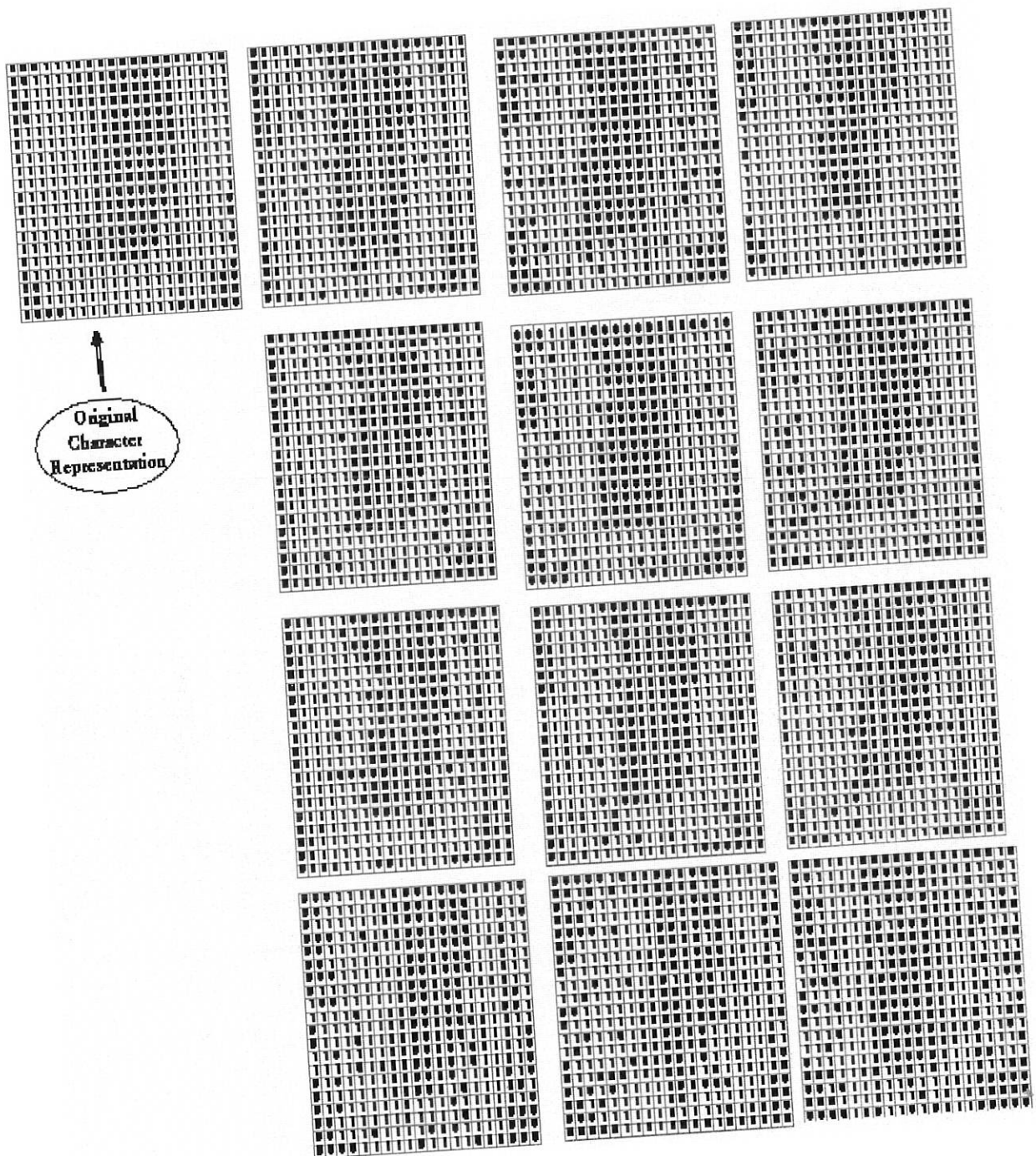
Annex-6: The Amharic Character Set

Order							Labialized				
1 st	2 nd	3 rd	4 th	5 th	6 th	7 th					
ሀ	ሁ	ሂ	ሃ	ሄ	ሀ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ					
ሐ	ሑ	ሒ	ሓ	ሔ	ሐ	ሑ					
መ	ሙ	ሚ	ማ	ሚ	ም	ሞ					
ሠ	ሡ	ሢ	ሣ	ሤ	ሠ	ሡ					
ረ	ሩ	ሪ	ራ	ሪ	ር	ሮ					
ሰ	ሱ	ሲ	ሳ	ሴ	ሰ	ሱ					
ሸ	ሹ	ሺ	ሻ	ሼ	ሸ	ሹ					
ቀ	ቁ	ቂ	ቃ	ቄ	ቀ	ቁ					
በ	ቡ	ቢ	ባ	ቤ	በ	ቡ					
ተ	ቱ	ቲ	ታ	ቲ	ቱ	ቲ					
ገ	ገ	ጊ	ገ	ጊ	ገ	ገ					
ነ	ነ	ነ	ነ	ነ	ነ	ነ					
ኀ	ኁ	ኂ	ኃ	ኄ	ኀ	ኆ					
ወ	ወ	ወ	ወ	ወ	ወ	ወ					
ዐ	ዑ	ዒ	ዓ	ዒ	ዐ	ዑ					
ከ	ከ	ከ	ከ	ከ	ከ	ከ					
ኸ	ኹ	ኺ	ኻ	ኼ	ኸ	ኹ					
ዘ	ዘ	ዘ	ዘ	ዘ	ዘ	ዘ					
ዐ	ዐ	ዐ	ዐ	ዐ	ዐ	ዐ					
የ	የ	የ	የ	የ	የ	የ					
ገ	ገ	ገ	ገ	ገ	ገ	ገ					
ደ	ደ	ደ	ደ	ደ	ደ	ደ					
ጀ	ጀ	ጀ	ጀ	ጀ	ጀ	ጀ					
ጠ	ጠ	ጠ	ጠ	ጠ	ጠ	ጠ					
ጨ	ጨ	ጨ	ጨ	ጨ	ጨ	ጨ					
ጸ	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ					
ሀ	ሀ	ሀ	ሀ	ሀ	ሀ	ሀ					
ጸ	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ					
ፈ	ፈ	ፈ	ፈ	ፈ	ፈ	ፈ					
ፒ	ፒ	ፒ	ፒ	ፒ	ፒ	ፒ					

ሸ ሹ ሺ ሻ ሼ ሽ ሾ

Source: [Bender et al, 1976]

Annex-7: Sample Randomly Generated Noisy Character



The noise is generated for the first character at 0.3 level of Gaussian noise

Annex-8: MATLAB code for creating and training the network

% This function will take the binary representation of the three training character sets with either 400x231 or 256x231 matrix.
 % The code first train the network with the original training sets. Then random noise will be added to each of the characters in %
 % the three% training sets and the network will be trained with it. Training with noise is done in such a way that, each training
 % character set along with it two copies with different level of noise will be fed to the network.

```
function [YekumTsifetNetwork, tr]=YekumTsifetTrain_Three(Train1, Train2, Train3, input_range, Fedel_Target)
```

```
%Network with 40 hidden and 8 output Layer is created with activation function 'logsig'
```

```
YekumTsifetNetwork = newff(input_range, [40 8], {'logsig','logsig'}, 'traingdx');
```

```
Time_Start=clock;
```

```
YekumTsifetNetwork.performFcn = 'mse';
YekumTsifetNetwork.trainParam.lr= 0.2;
YekumTsifetNetwork.trainParam.goal = 0.001;
YekumTsifetNetwork.trainParam.show = 10;
YekumTsifetNetwork.trainParam.epochs = 10000;
YekumTsifetNetwork.trainParam.mc = 0.8;
YekumTsifetNetwork.trainParam.min_grad = 1e-15;
```

```
TrainAll=[ Train1, Train2, Train3];
TargetAll= [Fedel_Target Fedel_Target Fedel_Target];
```

```
%Training the network with the original training charcater sets
[YekumTsifetNetwork,tr] = train(YekumTsifetNetwork,TrainAll,TargetAll);
```

```
YekumTsifetNetworkn = YekumTsifetNetwork;
YekumTsifetNetworkn.trainParam.goal = 0.002;
YekumTsifetNetworkn.trainParam.lr= 0.02;
YekumTsifetNetworkn.trainParam.epochs = 10000;
```

```
[N,M]=size(Train1);
T = [Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target Fedel_Target];
```

```
%Training the network with 100 training batches here each batch is having 9 forms of each character.
```

```
for pass = 1:100
    P = [Train1, Train2, Train3, ...
        (OZMatrix(Train1 + randn(N,M)*0.2)),...
        (OZMatrix(Train1 + randn(N,M)*0.3)),...
        (OZMatrix(Train2 + randn(N,M)*0.2)), ...
        (OZMatrix(Train2 + randn(N,M)*0.3)), ...
        (OZMatrix(Train3 + randn(N,M)*0.2)), ...
        (OZMatrix(Train3 + randn(N,M)*0.3))];
```

```
%Training the network with the three original training character sets and two levels of noisy set for each original set.
[YekumTsifetNetworkn,tr] = train(YekumTsifetNetworkn,P,T);
end
```

```
YekumTsifetNetwork.performFcn = 'mse';
YekumTsifetNetwork.trainParam.lr= 0.2;
YekumTsifetNetwork.trainParam.goal = 0.001;
YekumTsifetNetwork.trainParam.show = 10;
YekumTsifetNetwork.trainParam.epochs = 10000;
YekumTsifetNetwork.trainParam.mc = 0.8;
YekumTsifetNetwork.trainParam.min_grad = 1e-15;
```

```
%Training the network with the original training character sets again
[YekumTsifetNetwork,tr] = train(YekumTsifetNetwork,TrainAll,TargetAll);
```

```
time=etime(clock, Time_Start);
disp(time);
return;
```

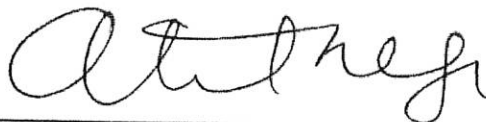

DECLARATION

This thesis is my original work, has not been presented for a degree in any other university and all sources of material used for the thesis have been duly acknowledged.



Wondwossen Mulugeta Gewe

THE THESIS HAS BEEN SUBMITTED FOR EXAMINATION WITH OUR
APPROVAL AS UNIVERSITY ADVISORS

 27th June 2004

Dr. Atul Negi

**Department of Computer and Information Sciences,
University of Hyderabad, India.**



Ato Nigussie Tadesse