Tribhuvan University
Institute of Science and Technology

Off-line Nepali Handwritten Character Recognition Using MLP and RBF Neural Networks

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Masters Degree in Computer Science & Information Technology

By
Ashok Kumar Pant
Date: 05 June, 2012

Tribhuvan University
Institute of Science and Technology

Off-line Nepali Handwritten Character Recognition Using MLP and RBF Neural Networks

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Masters Degree in Computer Science & Information Technology

By
Ashok Kumar Pant
Date: 05 June, 2012

Supervisor
Prof. Dr. Shashidhar Ram Joshi

Co-Supervisor
Dr. Sanjeeb Prasad Panday

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

Students Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

... ...
Ashok Kumar Pant
Date: 05 June, 2012

Supervisor's Recommendation

I hereby recommend that this dissertation prepared under my supervision by **Mr. Ashok Kumar Pant** entitled **Off-line Nepali Handwritten Character Recognition Using MLP and RBF Neural Networks** in partial fulfilment of the requirements for the degree of M.Sc. in Computer Science and Information Technology be processed for the evaluation.

... ...
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics & Computer Engineering,
Institute of Engineering,
Pulchowk, Nepal

Date: 05 June, 2012

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

Evaluation Committee

... ...
Asst. Prof. Nawaraj Paudel
Central Department of Computer Science & Information Technology,
Tribhuvan University, Kathmandu, Nepal
(Act. Head)

... ...
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics & Computer Engineering, Institute of Engineering,
Pulchowk, Kathmandu, Nepal
(Supervisor)

... ...
Assoc. Prof. Mr. Bal Krishna Bal
(External Examinar)

... ...
Mr. Bishnu Gautam
(Internal Examinar)

Date: 22 June, 2012

ACKNOWLEDGEMENTS

With deep sense of gratefulness I express my genuine thanks to my respected and worthy supervisor **Prof. Dr. Shashidhar Ram Joshi**, Head of Electronics & Computer Engineering Department(Kathmandu, Nepal) for his valuable guidance in carrying out this work under his effective supervision and enlightenment.

I am also thankful to all the staff members of the Department of Computer Science, TU (Kathmandu, Nepal) for their full cooperation and help.

ABSTRACT

An off-line Nepali handwriting recognition, based on the neural networks, is described in this research work. For the recognition of off-line handwritings with high classification rate a good set of features as a descriptor of image is required. Two important categories of the features are described, geometric and statistical features for extracting information from character images. Directional features are extracted from geometry of skeletonized character image and statistical features are extracted from the pixel distribution of skeletonized character image. The research primarily concerned with the problem of isolated handwritten character recognition for Nepali language. Multilayer Perceptron (MLP) & Radial Basis Function (RBF) classifiers are used for classification. The principal contributions presented here are preprocessing, feature extraction and MLP & RBF classifiers. The another important contribution is the creation of benchmark dataset for off-line Nepali handwritings. There are three datasets for Nepali handwritten numerals, Nepali handwritten vowels and Nepali handwritten consonants respectively. Nepali handwritten numeral dataset contains total 288 samples for each 10 classes of Nepali numerals, Nepali handwritten vowel dataset contains 221 samples for each 12 classes of Nepali vowels and Nepali handwritten consonant dataset contains 205 samples for each 36 classes of Nepali consonants. The strength of this research is efficient feature extraction and the comprehensive classification schemes due to which, the recognition accuracy of **94.44%** is obtained for Nepali handwritten numeral dataset, **86.04%** is obtained for Nepali handwritten vowel dataset and **80.25%** is obtained for Nepali handwritten consonant dataset.

Keywords:

Off-line handwriting recognition, Image processing, Neural networks, Multilayer perceptron, Radial basis function, Preprocessing, Feature extraction, Nepali handwritten datasets

TABLE OF CONTENTS

Acknowledgement i

Abstract ii

List of Figures vi

List of Tables vii

List of Algorithms viii

Abbreviations ix

1 INTRODUCTION 1
- 1.1 Introduction . . . 1
  - 1.1.1 Motivation . . . 1
  - 1.1.2 On-line versus Off-line Recognition . . . 2
  - 1.1.3 Nepali Natural Handwriting . . . 2
  - 1.1.4 Application of Off-line Nepali Handwriting Recognition System . . . 3
  - 1.1.5 Human performance in recognizing Handwritten Texts . . . 3
  - 1.1.6 Artificial Neural Networks . . . 3
    - 1.1.6.1 History of Artificial Neural Networks . . . 4
    - 1.1.6.2 Real Life Applications of Artificial Neural Networks . . . 5
- 1.2 Challenges . . . 5
- 1.3 Problem Definition . . . 6
- 1.4 Objectives . . . 6
- 1.5 Contribution of this Thesis . . . 6
- 1.6 Outline of the Thesis . . . 7

2 Literature Review 8
- 2.1 Previous Work . . . 8
- 2.2 Individual Character Recognition . . . 9
- 2.3 Word Recognition . . . 9
- 2.4 Preprocessing . . . 9
- 2.5 Feature Extraction . . . 10
- 2.6 Recognition Methods . . . 10
  - 2.6.1 Template Matching . . . 10
  - 2.6.2 Statistical Techniques . . . 10
  - 2.6.3 Structural Techniques . . . 11
  - 2.6.4 Neural Network Techniques . . . 11

3 SYSTEM OVERVIEW **12**
3.1 Image Acquisition . . . 12
3.2 Image Preprocessing . . . 14
3.3 Feature Extraction . . . 14
3.4 Recognition . . . 15

4 RESEARCH METHODOLOGY **16**
4.1 Image Preprocessing . . . 16
4.1.1 RGB to Grayscale Conversion . . . 17
4.1.2 Noise Removal . . . 17
4.1.3 Image Segmentation . . . 18
4.1.4 Image Inversion . . . 18
4.1.5 Universe of Discourse . . . 19
4.1.6 Size Normalization . . . 19
4.1.7 Image Skeletonization . . . 19
4.2 Feature Extraction . . . 21
4.2.1 Directional Features . . . 21
4.2.2 Moment Invariant Features . . . 22
4.2.3 Euler Number . . . 23
4.2.4 Normalized Area of Character Skeleton . . . 24
4.2.5 Centroid of Image . . . 24
4.2.6 Eccentricity . . . 24
4.3 Training and Testing . . . 25
4.3.1 Multilayer Feedforward Backpropagation Network . . . 25
4.3.1.1 Levenberg-Marquardt Learning Algorithm . . . 26
4.3.1.2 Gradient descent with momentum and adaptive learning rate . . . 28
4.3.2 Radial Basis Function Network . . . 29
4.3.2.1 Orthogonal Least Square Training Algorithm . . . 30
4.3.3 Performance Metrics . . . 31

5 IMPLEMENTATION **32**
5.1 MATLAB . . . 32
5.2 Image Processing Toolbox . . . 33
5.3 Neural Network Toolbox . . . 33
5.4 Feature Vector Creation . . . 34

6 NEPALI HANDWRITTEN DATASETS **37**
6.1 Dataset Creation Procedure . . . 37
6.2 Nepali Handwritten Consonant Dataset . . . 38
6.3 Nepali Handwritten Vowel Dataset . . . 38
6.4 Nepali Handwritten Numeral Dataset . . . 39

7 EXPERIMENTATIONS AND RESULTS **40**

8 CONCLUSION **49**
8.1 Conclusion . . . 49
8.2 Future Scope . . . 50

Appendix A Neural Network Parameters **51**

A.1 NguyenWidrow Weights Initialization 51
A.2 Activation Function 51

Appendix B Sample Source Codes **53**
B.1 Image Preprocessing Source Code 53
B.2 Feature Extraction Source Code 55
B.3 Training and Testing Source Code 57

LIST OF FIGURES

1.1 Sample of Nepali Handwritten Consonants. 2
1.2 Sample of Nepali Handwritten Vowels. 3
1.3 Sample of Nepali Handwritten Numerals. 3
1.4 A Physical Neuron. 4
1.5 An Artificial Neuron. 4

3.1 Off-line Handwriting Recognition System. 12
3.2 Image Slicing. 13
3.3 Image Acquisition. 13
3.4 Image Preprocessing. 14
3.5 Feature Extraction. 14
3.6 Training and Testing. 15

4.1 RGB to Grayscale Conversion. 17
4.2 Noise Removal. 17
4.3 Image Binarization. 18
4.4 Image Negatives. 19
4.5 Universe of Discourse. 19
4.6 Size Normalization. 20
4.7 Binary Image Skeletonization. 21
4.8 Feedforward Multilayer Perceptron. 26
4.9 RBF Neural Network. 29

5.1 Nepali consonant letter 'ka'. 34
5.2 Letter 'ka' as binary matrix. 34
5.3 Zoned images 34

6.1 Nepali Handwritten Consonants. 38
6.2 Nepali Handwritten Vowels. 38
6.3 Nepali Handwritten Numerals. 39

7.1 Recognition Accuracy of Off-line Handwriting Recognition Systems. 41
7.2 Training Time of Off-line Handwriting Recognition Systems. 42
7.3 Recognition Accuracy of Each Numeral Classes. 43
7.4 Recognition Accuracy of Each Vowel Classes. 44
7.5 Recognition Accuracy of Each Consonant Classes. 48

A.1 Hyperbolic tangent sigmoid activation function. 52

LIST OF TABLES

7.1 Neural Network Configurations. 40
7.2 Recognition Results. 41
7.3 Recognition Results For Individual Numerals. 42
7.4 Confusion Matrix of Numeral Dataset Testing. 43
7.5 Recognition Results For Individual Vowels. 44
7.6 Confusion Matrix of Vowel Dataset Testing. 44
7.7 Recognition Results For Individual Characters. 45
7.8 Confusion Matrix (I) of Consonant Dataset Testing. 46
7.9 Confusion Matrix (II) of Consonant Dataset Testing. 47

LIST OF ALGORITHMS

4.1 Image Preprocessing 16
4.2 Image Binarization 18
4.3 Image Skeletonization Algorithm 20
4.4 Levenberg-Marquardt Back-propagation Algorithm 28
4.5 OLS Algorithm 31
6.1 Handwritten Dataset Creation 37
A.1 Nguyen-Widrow Weight Initialization Algorithm 51

LIST OF ABBREVIATIONS

ADALINE (ADAptive LInear Element)

ANN Artificial Neural Network

CM Confusion Matrix

CR Character Recognition

GDMA Gradient Descent with Momentum & Adaptive Learning Rate

HWR Handwriting Recognition

IPT Image Processing Toolbox

LM Levenberg-Marquardt

LMS Least Mean Square

LS Least Square

MAT Medial Axis Transformation

MLP Multilayer Perceptron

NNT Neural Network Toolbox

OCR Optical Character Recognition

OLS Orthogonal Least Square

RBF Radial Basis Function

RBFNN Radial Basis Function Neural Network

SVM Support Vector Machine

VLSI Very Large Scale Integration

Chapter 1

INTRODUCTION

1.1 Introduction

Handwriting Recognition is the mechanism for transforming the written text into a symbolic representation. Off-line handwriting recognition is the task of determining what characters or words are present in a digital image of handwritten text. The problem of handwriting recognition can be viewed as a classification problem where we need to identify the most suitable character the given figure matches to. It is a sub-field of Optical Character Recognition (OCR), whose domain can be machine-print or handwriting but is more commonly machine-print. Handwriting Recognition plays an essential role in many human computer interaction applications including cheque verification, mail sorting, office automation, handwritten form verification, etc.

1.1.1 Motivation

Handwriting recognition is a special problem in the domain of pattern recognition and machine intelligence. The field of handwriting recognition can be split into two different categories: on-line recognition and off-line recognition. On-line mode deals with the recognition of handwriting captured by a tablet or similar touch-screen device, and uses the digitized trace of the pen to recognize the symbol. Off-line mode determines the recognition of text images. More about on-line and off-line techniques is given in section 1.1.2. The problem of handwriting increases when we operate it in the off-line mode. Lots of work has been done in this area in the past few years. The motivation behind developing character recognition systems is inspired by its wide range of applications including human-computer interaction, archiving documents, automatic reading of checks, number plate reading of vehicles, etc.

Artificial Neural Network (ANN) in the area of handwriting recognition achieves more efficiency and accuracy than using other statistical recognition tools. ANN is grabbing great attention in the field of pattern recognition due to its simple structure, high accuracy, parallel computing, fault-tolerance and self learning capacity.

Nepali handwriting recognition system can also help in automatic reading of ancient Nepali manuscripts. Automatic processing benefits into availability for their contents. Besides the advantages of automated handwriting system, image degradation, unexpected markings and previously unseen writing styles provide challenges in the recognition procedure.

1.1.2 On-line versus Off-line Recognition

In On-line handwriting recognition, user is directly connected to the system using an electronic pen or a touch-screen and recognition is carried out in real time. All various geometrical attributes, e.g. positions, distances, angles, curvatures and local directions have been evaluated on sample points. We also have temporal information about the character being written, such as the stroke order, pen up and pen down time, sequence of points traced, etc. High accuracy can be achieved in the case of on-line handwriting recognition, since we can extract various important geometric, statistic and temporal features from character being written.

In off-line handwriting recognition, the recognition is carried out on handwritten text that is captured using a scanner or a camera. Thus, the text is treated as an image. Off-line approach is free from stroke order variations. However, both on-line and off-line methods are similar in many aspects except for the fact that off-line methods does not have any temporal information. However, using certain heuristic methods and prior knowledge (for example in the Roman script, writing is always performed from left to right), we can determine the direction of the strokes with respect to time.

In on-line approach, the character is normally represented as a sequence of feature vectors which are extracted along the pen tip trajectories and thus exhibits some dynamic characteristics. In contrast, in off-line approach, the character is modelled by the feature vector describing the holistic shape directly.

1.1.3 Nepali Natural Handwriting

Nepali language belongs to Devanagari script which is invented by Brahmins around 11th century AD. Devanagari script is also adapted in many other languages like Hindi, Marathi, Bangali and many more. In Nepali language, there are 33 pure consonants (vyanjan) and also half forms, 13 vowels (swar), 16 modifiers and 10 numerals [1]. In addition, consonants occur together in clusters, often called conjunct consonants. Altogether, there are more than 500 different characters. According to the Nepal census, conducted by His Majesty's Government of Nepal in 2001 AD, More than 17 million speakers worldwide, including 11 million within Nepal speak Nepali language.

Nepali is written from left to right along a horizontal line. Characters are joined by horizontal bars that create an imaginary line by which Nepalese texts are suspended, often called 'shirorekha. The single or double vertical line at the end of writing represents a completeness of one sentence which is called 'purnabiram'. A few samples of nepali handwritten characters are shown in Figure 1.1, Figure 1.2 and Figure 1.3.

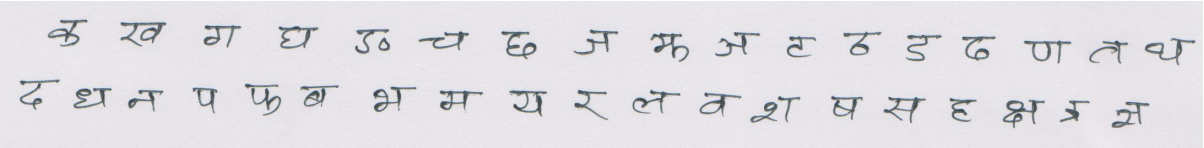

Figure 1.1: Sample of Nepali Handwritten Consonants.

Figure 1.2: Sample of Nepali Handwritten Vowels.

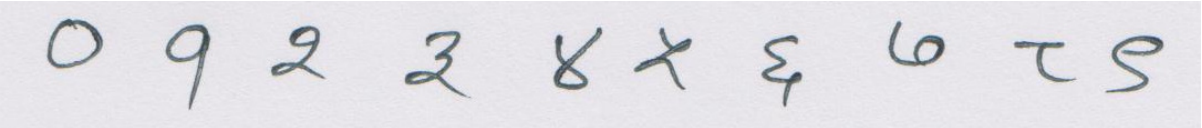

Figure 1.3: Sample of Nepali Handwritten Numerals.

1.1.4 Application of Off-line Nepali Handwriting Recognition System

The main industrial applications for the recognition of off-line handwritten text are currently found in the area of address reading and bank cheque processing, where recognition is based on digitized images of physical envelopes or bank cheques.

Applications for off-line recognition of unconstrained handwritten text may be found in the future to the automatic recognition of personal notes and communications. Recognition rate of current systems in the off-line, writer independent case are still far from being perfect.

The automatic transcription of large handwritten archives seems a more realistic scenario today. Such applications could support automatic indexing for information retrieval systems used in digital libraries which do not require perfect recognition rates.

Making handwritten texts available for searching and browsing is of tremendous value. To digitalize the historic documents written in ancient time, off-line Nepali handwriting recognition plays a good role. Indexing of handwritten documents for searching and sorting is another application area of off-line Nepali Handwriting recognition.

1.1.5 Human performance in recognizing Handwritten Texts

Most people can read handwritings with word recognition rates of 100% if they are familiar with that particular language. Human performance will be often poor, if a letter by letter recognition is required. The knowledge of the possible vocabulary and frequently used word sequences (word n-grams) can help to improve the word recognition rate. However, optimal transcriptions of handwritten texts can be expected if the reader is familiar with both the language and the topic of the text, i.e. if the text is actually understood.

It can generally be observed that the amount of task specific knowledge has a significant impact on the resulting system performance.

1.1.6 Artificial Neural Networks

Artificial neural network is non-linear, parallel, distributed, highly connected network having capability of adaptivity, self-organization, fault tolerance, evidential response and Very Large Scale Integration (VLSI) implementation, which closely resembles with physical nervous system. Physical nervous system is highly parallel, distributed information processing system having high degree of connectivity with capability of self learning. Human nervous system

contains about 10 billion neurons with 60 trillions of interconnections. These connections are modified based on experience. Typical physical neuron is given in figure 1.4 and artificial neuron is given in figure 1.5.

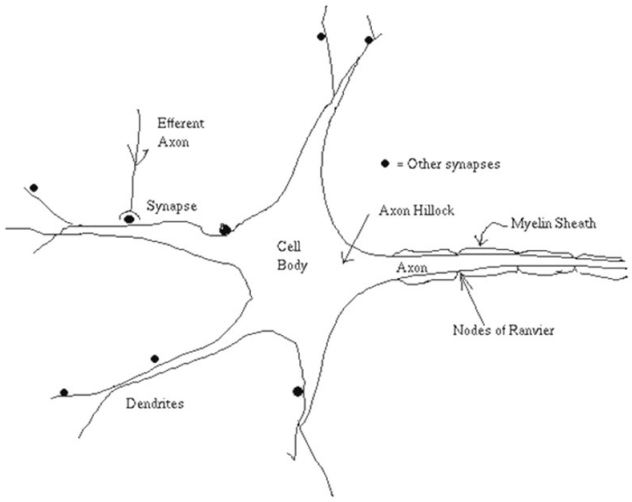

Figure 1.4: A Physical Neuron.

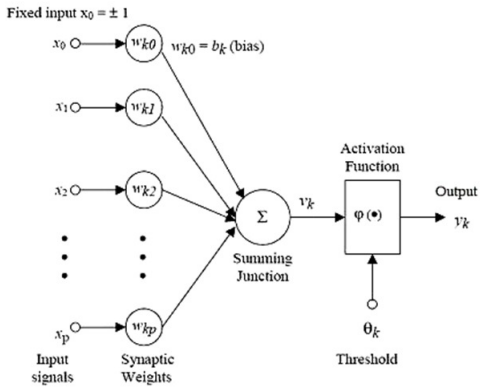

Figure 1.5: An Artificial Neuron.

Artificial neural networks are composed of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real, biological nervous system is highly complex. Artificial neural network algorithms attempt to abstract this complexity and focus on what may hypothetically matter most from an information processing point of view. Good performance (e.g. as measured by good predictive ability, low generalization error), or performance mimicking animal or human error patterns, can then be used as one source of evidence towards supporting the hypothesis that the abstraction really captured something important from the point of view of information processing in the brain. Another incentive for these abstractions, is to reduce the amount of computation required to simulate artificial neural networks, so as to allow one to experiment with larger networks and train them on larger data sets.

1.1.6.1 History of Artificial Neural Networks

The modern view of neural network began in the 1940s with the work of Warren McCulloch and Walter Pitts, who showed that networks of artificial neurons could, in principle, compute any arithmetic or logical function.

McCulloch and Pitts were followed by Donald Hebb, who proposed that classical conditioning (as described by Pavlov in the experiment of food presenting to the dog) is present because of the properties of individual neurons. Hebb proposed a mechanism for learning artificial neurons in the way of biological neurons.

The first practical application of the artificial neural networks came in the late 1950s, with the invention of the perceptron network and associative learning rule by Frank Rosenblatt [2] .

Around 1956, Bernard Widrow and Ted Hoff introduced a new learning algorithm and used to train adaptive linear neural networks, which is similar in structure and capability to Rosan-

blatts perceptron [3] . Another system besides perceptron was the (ADAptive LInear Element) (ADALINE) which was developed in 1960 by Widrow and Hoff (of Stanford University).

Rosenblatts and Widrows network both suffer from limitation of applicability of networks only to linearly seperable classes of problems. Limitations of these networks were publicized in the book by Marvin Minsky and Seymour Papert with the fact that there were no powerful digital computers to do experiments. So, for a decade, neural network research was largely suspended.

Development of neural network dramatically increased from 1980, as there were powerful personal computers to do experiments and the development of multilayer backpropagation perceptron network. The most influenced publication of the backpropagation algorithm was by Devid Rumelhart and James Mclelland. This algorithm was the answer to the criticisms Minsky and Papert had made in the 1960s.

Significant progress has been made in the field of neural networks-enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology.

1.1.6.2 Real Life Applications of Artificial Neural Networks

Some real life applications of Artificial Neural Network are given below.

- Signal processing (e.g. adaptive echo cancelling).
- Control (e.g. manufacturing plants for controlling automated machines).
- Robotics (e.g. vision recognition).
- Pattern recognition (e.g. recognizing handwritten characters).
- Medicine (e.g. storing medical records based on case information).
- Speech production (e.g.reading text aloud).
- Speech recognition.
- Vision (e.g. face recognition).
- Business (e.g. rules for mortgage decisions).
- Financial Applications (e.g. stock market prediction).
- Data Compression (e.g. images).
- Game Playing (e.g. chess).

1.2 Challenges

Unconstrained off-line handwriting recognition has significantly different writing styles and shapes. Shapes of the same character glyph vary across writers and even for the same writer. General handwritten text document can have large variations on writings. Same Character, word or sentence can have different writing styles, alignment styles, skewness and slantness for the same document. It makes handwriting recognition more complicated.

Off-line handwritten text recognition can be seen as the most general case of handwriting recognition. Complexity of handwriting recognition system vary according to its problem domain (e.g., Number plate recognition, Cheque verification, Digit recognition, etc...). Writer Recognition is much more difficult than Writer Independent Handwriting Recognition. So, unconstrained off-line handwriting recognition is a very challenging problem in the domain of pattern recognition and artificial intelligent.

1.3 Problem Definition

The high-level task of off-line handwriting recognition is to classify the ordered sequence of images of off-line characters. In this research work, problem of Nepali handwritten character recognition is addressed. This corresponds to the ability of human beings to recognize such characters, which they are able to do with little or no difficulty. The recognition task is carried out with Artificial Neural Network. Many geometric and statistical features are extracted from images so that the performance and accuracy of recognition system is achieved in the range of human ability of recognition. The system performs character recognition by quantification of the character into a mathematical vector entity using the geometrical and statistical properties of the character image. The sub problems in the domain of off-line handwriting recognition such as, noise removal, image binarization, object skeletonization, size normalization, etc. have great impact on recognition procedure. These sub-problems are also addressed with the most suitable solutions in the literature for this type of research work.

1.4 Objectives

The objective of this research is to investigate various feature extraction techniques and to compare Neural Network based pattern recognition techniques namely Multilayer Feed-forward Network and Radial Basis Function Network. Comparative Performance matrices are analysed. The sub-problem field of off-line handwriting recognition is also addressed. Main objectives are given below.

- To compare performance and efficiency of MLP and RBF Neural Networks on Off-line Nepali Handwriting Recognition Problem.
- To investigate Geometric and Statistical feature extraction techniques for off-line Nepali handwritten text.
- To investigate preprocessing techniques (segmentation, skeletonization, normalization, etc.) for handwritten documents.
- To create benchmark databases for Nepali handwritten characters.

1.5 Contribution of this Thesis

The main contribution of this thesis to the field of Off-line Nepali Handwriting Recognition can be seen in its extensive experimental work. A more detailed list of the various contributions is provided below,

- Use of ANN to analysis the off-line handwriting recognition problem.
- Investigation of feature extraction techniques for off-line handwritten documents.
- Creation of Nepali Handwritten character database for the experimentation with the recognition system.

1.6 Outline of the Thesis

The remaining part of the document is organized as follows,

Chapter 2 describes the state of the art of the handwriting recognition. It includes the methods and techniques used in the area of handwriting recognition till now.

Chapter 3 describes the Off-line Handwriting Recognition System architecture. The Top level system overview along with sub-system engines are given with data flow directions.

Chapter 4 describes research methodologies used in the research. All the preprocessing, feature extraction and recognition algorithms are described in this part of the document.

Chapter 5 describes the implementation of the system. It includes the techniques used for the realizations of the algorithms given in chapter 4.

Chapter 6 describes the corpus used in the evaluation of the purposed system. Off-line handwriting recognition system is experimented with three self created Nepali handwritten character datasets for consonants, vowels and numerals respectively.

Chapter 7 describes the experimentation results of the recognition systems. Performance and efficiency of the proposed systems evaluated in Nepali handwritten character datasets are given in this section.

Chapter 8 contains the summary and future scope of the research work.

Chapter 2

Literature Review

2.1 Previous Work

It is an ancient dream to make machines able to perform tasks like humans. The origin of character recognition can actually be found back in 1870 as C.R. Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells. Two decades later the Polish P. Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. During the first decades of the 19^{th} century several attempts were made to develop devices to aid the blind through experiments with OCR. However, the modern version of OCR did not appear until the middle of the 1940s with the development of the digital computer.

Before the age of digital computers, there was no such researches in the field of handwriting recognition. The early researches after the digital age were concentrated either upon machine-printed text or upon a small set of well-separated handwritten text or symbols. Machine-printed Character Recognition (CR) generally used template matching and for handwritten text, low-level image processing techniques were used on the binary image to extract feature vectors, which were then fed to statistical classifiers [4].

CR research is somewhat limited until 1980 due to the lack of powerful computer hardware and data perception devices. The period from 1980-1990 witnessed a growth in CR system development [5] due to rapid growth in information technology [6]. However, the CR research was focused on basically the shape recognition techniques without using any semantic information. This led to an upper limit in the recognition rate, which was not sufficient in many practical applications. Research progress on the off-line and on-line recognition during 1980-1990 can be found in [7] and [8] respectively.

After 1990, image processing techniques and pattern recognition techniques were combined using artificial intelligence. Along with powerful computers and more accurate electronic equipments such as scanners, cameras and electronic tablets, there came in efficient, modern use of methodologies such as artificial neural networks (ANNs), hidden Markov models (HMMs), fuzzy set reasoning and natural language processing.

Character segmentation from cursive handwritten documents is a difficult task. So in literature most of the researches were conducted on separated characters. Segmentation based approach for isolated off-line Devanagari word recognition is described in [9]. Isolated Devanagari character recognition with Regular Expressions & Minimum Edit Distance Method is described in

[10]. Research work [1] describes the template based Nepali alphanumeric handwriting recognition. Recent work on off-line devanagari character recognition carried out by Sharma et.al. (2006) uses quadratic classifiers for recognition and achieved 98.86% recognition accuracy for devanagari numerals and 80.36% recognition accuracy for devanagari characters [11]. Research work [12] describes multiple classifier combination for Off-line Handwritten Devnagari Character Recognition and achieves 92.16% recognition accuracy for devanagari characters. Paper [13] compares Support Vector Machine (SVM) and ANN for off-line devanagari character recognition problem. Handwriting Recognition system based on cloud computing in given in paper [14].

On Devnagari, a few techniques have been tested but no comparison of various recognition techniques available in literature is made. Also, there is lack of benchmark databases for Handwritten Devnagari Script to test the recognition systems. Only small lab experiments have been found in the literature.

Although research on recognizing isolated handwritten characters has been quite successful, recognizing off-line cursive handwriting has been found to be a challenging problem. There is a large corpus of research on the application of character recognition in different domains, but no system to date has achieved the goal of system acceptability.

2.2 Individual Character Recognition

Recognition of individual characters have greater recognition accuracy than the recognition of whole words. But the segmentation of individual characters from handwritten documents is a error prone task due to the unconstrained domain of handwritings. Among all individual character recognition domains, individual digit recognition is much more researched. Due to connectivity and smooth drawing properties, digit recognition has higher accuracy than other handwriting recognition domains. There are many researches in the domain individual character recognition [5, 15, 16, 11, 1, 10, 17, 12, 13, 18, 19, 14, 20, 21]. Recognition algorithms used for the recognition of individual characters includes the probability based approaches, statistical approaches, neural network based approaches, etc.

2.3 Word Recognition

Word-based recognition system determine the entire word without any attempt to segment or locate individual characters. The methods used in word-based recognition are usually similar to those of character recognition.These approaches avoid the individual letter segmentation problem. However, the whole word have higher writing variability than in the case of single character. Researches in the domain of word recognition are included in [6, 22, 9]

2.4 Preprocessing

For off-line handwriting recognition, most of the data is extracted from the scans of pages of handwritten text. The quality of these scanned pages is often poor due to scanning artifacts, noise or low resolution. From these text pages, text lines or single words have to be extracted for recognition, non-text background like figures and other markings need to be ignored.

Depending on the captured image quality, the extracted text lines or words have to be preprocessed for further usage, for example, slant correction, skew correction, curve smoothing, size normalization, etc. Other preprocessing techniques include colour normalization, noise removal, image binarization, image skeletonization and components analysis for estimation of partial words and characters.

2.5 Feature Extraction

Feature extraction is one of the important stage of the handwriting recognition. Feature extraction is essential for efficient data representation and high recognition performance. A good feature set should represent the syntactic and semantic characteristic of a class that helps distinguish it from other classes. We can have many different features that can be extracted from preprocessed character images [23, 24, 25, 20].

There are three major categories of feature extraction techniques:

- **Geometrical and Topological Features:** Extracting and Counting Topological Structures, Geometrical Properties, Coding, Graphs, Trees, Strokes, Chain Codes etc.
- **Statistical Features:** Zoning, Crossing and Distances, Projections, Distribution measures, etc.
- **Global Transformation and Series Expansion Features:** Fourier Transform, Cosine Transform, wavelets, Moments, Karhuen-Loeve Expansion, etc.

2.6 Recognition Methods

There are large number of recognition techniques for the handwriting recognition problem [24]. The major category of recognition strategies are: template matching, statistical methods, structural methods, and neural networks.

2.6.1 Template Matching

Template matching techniques determine the degree of similarity between two vectors (pixel distributions, shapes, curvatures, etc) in the feature space. Matching techniques can be grouped into three classes: direct matching, deformable templates and elastic matching, and relaxation matching. One of the most widely used technique for template matching is Dynamic Programming.

2.6.2 Statistical Techniques

Statistical methods are concerned with statistical decision functions and a set of optimal criteria, which determine the probability of the observed pattern belonging to a certain class. In statistical representation, the input pattern is described by a feature vector. Statistical techniques use concepts from statistical decision theory to establish decision boundaries between pattern classes. There are several statistical techniques for handwriting recognition, such as, k-Nearest Neighbour, Hidden Markov Model, Fuzzy set reasoning, Support Vector Machine, etc.

2.6.3 Structural Techniques

In structural methods the handwritten characters are represented as unions of structural primitives. In the structural methods the character primitives extracted from handwriting are quantifiable, and one can find the relationship among them. Basically, structural methods can be categorized into two classes: grammatical methods and graphical methods.

2.6.4 Neural Network Techniques

A Neural Network is a computing structure consisting of a massively parallel interconnection of artificial adaptive neurons. The main advantages of neural networks is its ability to be trained automatically from examples, good performance with noisy data, possible parallel implementation, and efficient tools for learning large databases. There are many neural network based recognition techniques like multilayer perceprton, radial basis function, recurrent networks, self organizing maps, etc.

Chapter 3

SYSTEM OVERVIEW

The Top level handwriting recognition system is divided into four sub-systems, image acquisition, preprocessing, feature extraction and recognition. The top level model of proposed system is given in Figure 3.1.

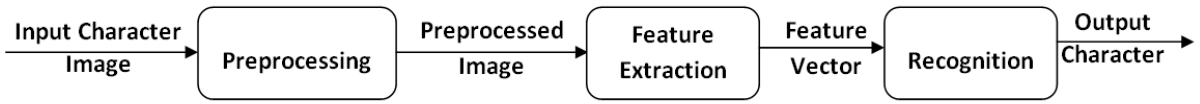

Figure 3.1: Off-line Handwriting Recognition System.

The basic system architecture of the image acquisition, preprocessing, feature extraction and recognition is briefly described in section 3.1, section 3.2, section 3.3 and section 3.4, respectively.

3.1 Image Acquisition

Images are acquired by scanning the handwritten documents using digital scanners or by taking photograph of the handwriting document using digital camera. CANON LIDE scanner is used to scan the handwritten samples from different writers. Each images are scanned at 300dpi in RGB color mode. Detail of the handwritten corpus used for the experimentation is given in chapter 6. Figure 3.2 shows the slicing of off-line character images from handwritten samples and Figure 3.3 shows the final stage of the image acquisition.

Figure 3.2: Image Slicing.

Figure 3.3: Image Acquisition.

3.2 Image Preprocessing

Pre-processing is done prior to feature extraction algorithms. The raw images are subjected to a number of preliminary processing steps to make it usable in the descriptive stages of character analysis. Pre-processing aims to produce clean document images that are easy for the Recognition systems to operate accurately. The block diagram of preprocessing system is given in Figure 3.4. Details of the each preprocessing steps is described in the section 4.1.

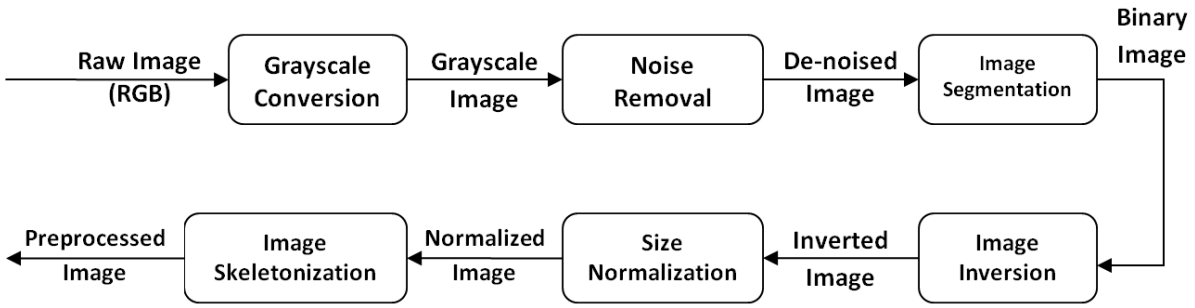

Figure 3.4: Image Preprocessing.

3.3 Feature Extraction

After pre-processing of the character images, feature vectors are extracted, which is used in the training and recognition stage. Feature sets play one of the most important roles in a recognition system. A good feature set should represent characteristic of a class that helps distinguish it from other classes, while remaining invariant to characteristic differences within the class. The high level block diagram of the feature extraction system is given in the Figure 3.5. Detail description of each feature extraction techniques is given in section 4.2.

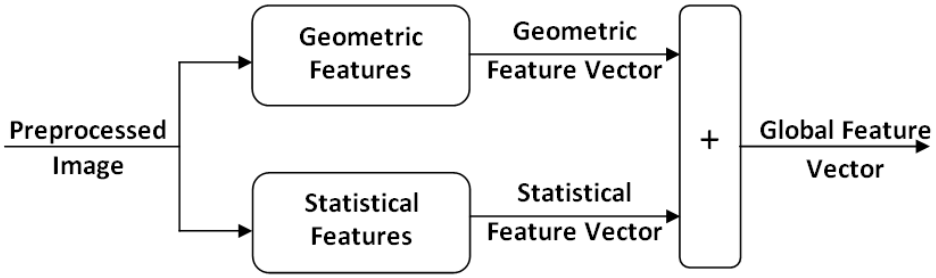

Figure 3.5: Feature Extraction.

3.4 Recognition

Recognition engine of the system consists of two neural network based algorithms implemented in it. There are two stages of the recognition, training and testing. In the training stage system learns how to behave in the new environment of similar inputs and in the testing stage accuracy of the classification is determined. The top level recognition system is given in figure 3.6. Detail of the each algorithms used in the recognition system is given in the section 4.3.

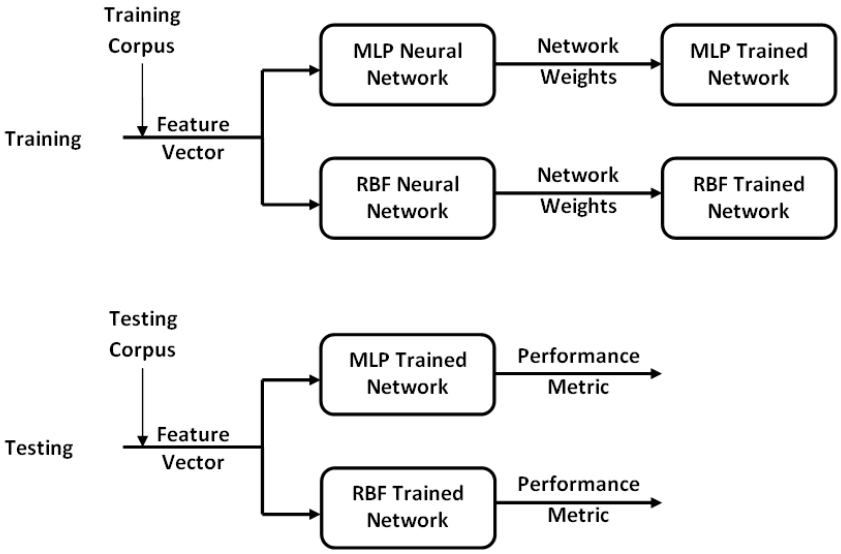

Figure 3.6: Training and Testing.

Chapter 4

RESEARCH METHODOLOGY

This chapter describes the theoretical concept behind the methods used in this research. Architecture of the proposed handwriting recognition system is given in chapter 3. Subsections of this chapter are organized as follows.

Section 4.1 describes the image preprocessing techniques. Feature extraction techniques are described in the section 4.2. System training and testing methods are described in the section 4.3.

4.1 Image Preprocessing

Image preprocessing is the preliminary step of the recognition procedure. Preprocessing describes how images are normalized before extracting features from it. Region of interest is selected from given character image and make it fine conditioned. This section provides the detail description about steps used in preprocessing of given character images. The main steps of image preprocessing are given in algorithm 4.1.

Algorithm 4.1 Image Preprocessing

1: Read image.
2: Convert RGB images to gray scale image.
3: Remove noise using median filter.
4: Convert the gray scale images to binary image.
5: Image Inversion (making background black & foreground white).
6: Determine the universe of discourse of image.
7: Normalize the image to a predefined size of 36x36 pixels.
8: Convert the normalized image to single pixel thick skeleton image.

Following subsections describe each of these steps in detail. The organization of subsections are as follows: Color normalization method is described in section 4.1.1, noise removal is described in section 4.1.2, image segmentation procedure is described in the section 4.1.3, image inversion is given in section 4.1.4, universe of discourse selection technique is presented in the section 4.1.5, size normalization method is given in section 4.1.6 and image skeletonization process is described in the section 4.1.7.

4.1.1 RGB to Grayscale Conversion

Given 24 bit true colour RGB image is converted into 8 bit grayscale image. Grayscale component is calculated by taking weighted summation of R, G and B components of RGB image. Weights are selected same as the NTSC color space selects for the luminance i.e. the grayscale signal used to display pictures on monochrome televisions. For the RGB image $f(x, y)$, corresponding graysclae image is given by,

$$g(x, y) = 0.2989 * R(f(x, y)) + 0.5870 * G(f(x, y)) + 0.1140 * B(f(x, y)) \tag{4.1}$$

where, $R(f(x, y))$ is red component of the RGB image $f(x, y)$, and so on.

Figure 4.1 shows the RGB to grayscale conversion of input image.

Figure 4.1: RGB to Grayscale Conversion.

4.1.2 Noise Removal

Noise removal is one of the important step of image preprocessing. Any unnecessary pixels are removed with the help of filtering. Non-linear median filtering technique is used for noise removal. Median filter is an effective method of noise removal which can suppress isolated noise without blurring sharp edges. Median filter replaces a pixel with the median value of its neighbourhood. For the digital image $f(x, y)$, median filtered image is obtained as,

$$g(x, y) = median\{f(i, j) \,|\, (i, j) \in \ w\} \tag{4.2}$$

where, w is the neighbourhood centered around location (x, y) in the image. Figure 4.2 shows the denoised image corresponding to given grayscaled image.

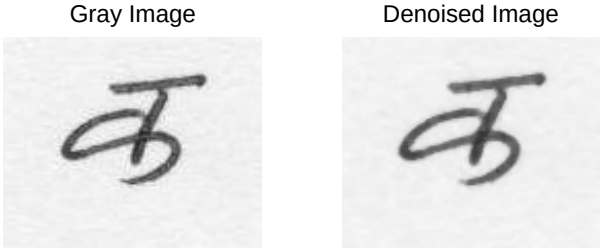

Figure 4.2: Noise Removal.

4.1.3 Image Segmentation

Segmentation is the central problem of distinguishing object from the background. There are many algorithms for image segmentation. This section describes the thresholding method with Otsu's threshold selection technique [26] for grayscale image segmentation.

Otsu's graylevel threshold selection method for image binarization is a nonparametric and unsupervised method of automatic threshold selection. An optimal threshold is selected by the discriminant criteria i.e., by maximizing the interclass variance between white and black pixels. Image segmentation algorithm for grayscale image is given in Algorithm 4.2. Figure 4.3 shows binarized image of given grayscale image.

Figure 4.3: Image Binarization.

Algorithm 4.2 Image Binarization

1: Compute histogram and probability of each intensity level i , $i = 1 \cdots L$.
2: Compute between class variance for black and white pixels, by taking threshold $t = 1 \cdots L$,
3: $\sigma_b^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$
where,
4: $\omega_1(t) = \sum_{1}^{t} p(i)$ is the class probability for black pixels,
5: $\omega_2(t) = \sum_{t+1}^{L} p(i)$ is the class probability for white pixels,
6: $\mu_1(t) = \sum_{1}^{t} \frac{ip(i)}{\omega_1(t)}$ is the class mean for black pixels,
7: $\mu_2(t) = \sum_{t+1}^{L} \frac{ip(i)}{\omega_2(t)}$ is the class mean for white pixels,
8: and, $p(i)$ is the probability of intensity level i.
9: Select t for which $\sigma_b^2(t)$ is maximum.
10: Finally, binarize the grayscale image $f(x, y)$ using the threshold t as,
11: $g(x, y) = \begin{cases} 1 & if\ f(x, y) \geq t \\ 0 & if\ f(x, y) < t \end{cases}$

4.1.4 Image Inversion

Handwritten documents are normally written in white paper with black pen. For the recognition system, we assume black pixels as a background and white pixels as the foreground. So,

captured images are inverted before passing into the recognition engine. The inverted image of binary image $f(x, y)$ can be obtained by the negative transformation as,

$$g(x, y) = 1 - f(x, y) \tag{4.3}$$

Figure 4.4 shows the negative image of given binary image.

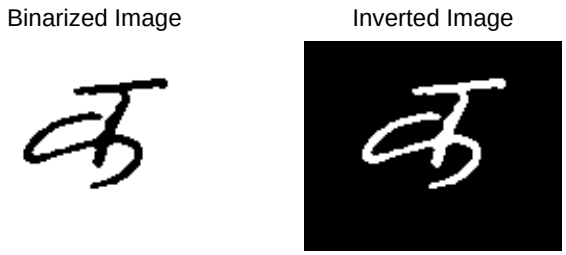

Figure 4.4: Image Negatives.

4.1.5 Universe of Discourse

Determining universe of discourse of the character image is finding smallest rectangle that encloses the character object. It removes extra pixels outside the convex hull of the character image. Figure 4.5 shows the discoursed image of given binary image.

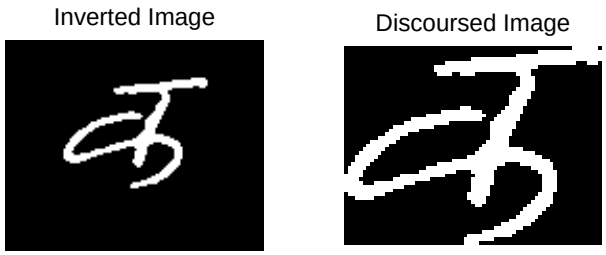

Figure 4.5: Universe of Discourse.

4.1.6 Size Normalization

Size normalization is the technique of converting all the variable size input images to fixed size images. Size normalization is done so that we do not require paddings of pixels at the time of feature extraction. All the input images are normalized to the predefined size of 36x36 pixels. Figure 4.6 shows the size normalized image.

4.1.7 Image Skeletonization

Skeletonization is a process for reducing object regions in a binary image to a skeletal remainder that largely preserves the extent and connectivity of the original object while throwing away

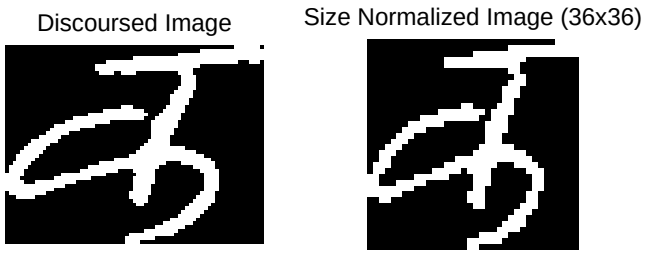

Figure 4.6: Size Normalization.

most of the original object pixels. It plays an important role in the preprocessing phase of handwriting recognition. Skeletonization creates single pixel wide connected object boundary, that preserves euler number of the original object. There are many reasons behind skeletonization of the binary image such as, to reduce the amount of data and time for processing the image, to extract critical features like end-points, intersection-points, connections etc., to help in shape analysis algorithms, and so on. Skeletonization calculates a medial axis skeleton so that points of this skeleton are at the same distance of its nearby borders. Image thinning algorithm is first given by Harry Blum (1967) [27]. Binary image skeletonization is carried out by the object thinning procedure that iteratively delete boundary points of a region object to the constraints that deletion of these points; (1) does not remove end points, (2) does not break connectivity and (3) does not cause excessive erosion of the region. Medial Axis Transformation (MAT) algorithm [28] for binary image thinning is given in algorithm 4.3. The MAT of a object region has an intuitive definition based on the so-called "prairie fire concept". Consider an image region, as a prairie of uniform dry grass, and suppose a fire it lit along its border. All fire fronts will advance into the region at the unique speed. The MAT of the object region is then the set of points reached by more than one fire front at the same time. Figure 4.7 shows the skeletonized image of given binary image.

Algorithm 4.3 Image Skeletonization Algorithm

1: Repeat following steps until no contour points.
2: (a) Delete all contour points according to Definition 1.
3: (b) Delete all contour points according to Definition 2.
4: Definition 1: For contour point $P1$ (see table right for the 8- connected relationship) that satisfies following conditions,
(a) $2 \leq N(P1) \leq 6$
(b) $T(P1) = 1$
(c) $P2.P4.P6 = 0$
(d) $P4.P6.P8 = 0$

P9	P2	P3
P8	P1	P4
P7	P6	P5

Table: The 8- connected relationship

where $N(P1)$ is the number of non-zero neighbours of $P1$: $N(P1) = \sum_{i=2}^{9} P_i$, and $T(P1)$ is the number of 0-1 transitions in the ordered sequence $P2, P3, \cdots, P9, P2$ (clockwise).
5: Definition 2: For contour point $P1$ that satisfies the following conditions
(a) same as above
(b) same as above
(c′) $P2.P4.P8 = 0$
(d′) $P2.P6.P8 = 0$

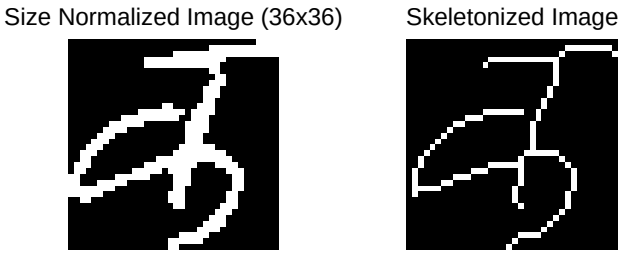

Figure 4.7: Binary Image Skeletonization.

4.2 Feature Extraction

After Pre-processing of the character images the next stage of character recognition is feature extraction. Feature extraction step plays one of the most important roles in the recognition. It is the heart of recognition system. A good feature set should represent characteristic of the class that helps distinguish it from other classes, while remaining invariant to characteristic differences within the class. Hundreds of features are available in the literature [23, 24, 25].

This section describes features extraction techniques. Combination of statistical features (Moment Invariants, Centroid and Area) and geometric features (Directional Features, Euler Number and Eccentricity) are used to describe the character property. These features are extracted from individual preprocessed handwritten character images.

4.2.1 Directional Features

Directional features are extracted from skeletonized character image based on the basic line types that form the skeleton of the character. These topological features may represent global and local properties of the object along with some knowledge about the contour of the object or some knowledge about the components of the object. This technique implements the idea given in [29]. Each pixel in the image is traversed. Individual line segments, there directions and intersection points are identified from segmented character image.

For this, given pre-processed character image is zoned into 3x3 windows of equal size. Then number, length, type of lines and intersection points are identified in each zone. The line segment that would be determined in each zone is categorized into four types: vertical lines, horizontal lines, right diagonal lines and left diagonal lines. To extract direction features, the following steps are required,

1. Starting points and intersection point identification.
2. Individual line segment identification.
3. Labelling line segment.
4. Line type normalization.

From each zone of the character image following properties are extracted [21].

1. Number of horizontal lines.
2. Number of vertical lines.

3. Number of Right diagonal lines.
4. Number of Left diagonal lines.
5. Normalized Length of all horizontal lines.
6. Normalized Length of all vertical lines.
7. Normalized Length of all right diagonal lines.
8. Normalized Length of all left diagonal lines.
9. Number of intersection points.

It results feature vector of dimensional 9 for each zone. A total of 81 (9x9) features are obtained.

4.2.2 Moment Invariant Features

Moment invariants are important tools in object recognition problem. These techniques grab the property of image intensity function. Moment invariants were first introduced to the pattern recognition community in 1962 by Hu [30], who employed the results of the theory of algebraic invariants and derived his seven famous invariants to rotation of 2-D objects. Moment invariants used in this research for extracting statistical patterns of character images are given in [28]. Moment invariants are pure statistical measures of the pixel distribution around the centre of gravity of the character and allow capturing the global character shape information.

The standard moments m_{pq} of order $(p+q)$ of an image intensity function $f(x,y)$ is given by,

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \quad p,q = 0,1,2,... \tag{4.4}$$

A uniqueness theorem states that if $f(x,y)$ is piecewise continues and has non-zero values only in a finite part of the x_{vis}, y_{vis} plane, moments of all order exist and the moment sequence (m_{pq}) is uniquely determined by $f(x,y)$. Conversely, (m_{pq}) is uniquely determines $f(x,y)$.

For discrete domain, the 2-D moment of order $(p+q)$ for a digital image $f(x,y)$ of size MxN is given by,

$$m_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^p y^q f(x,y) \quad p,q = 0,1,2,... \tag{4.5}$$

The corresponding central moment of order $(p+q)$ is defined as,

$$\mu_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(x-\bar{x})^p(y-\bar{y})^q f(x,y) \quad p,q = 0,1,2,... \tag{4.6}$$

where,

$$\bar{x} = \frac{m_{10}}{m_{00}} \; and \; \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.7}$$

The normalized central moments, denoted by η_{pq}, are defined as,

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{4.8}$$

where,

$$\gamma = \frac{p+q}{2} + 1 \quad for \quad p+q = 2, 3, ... \tag{4.9}$$

A set of seven invariant moments can be derived from the second and third moments [30] which are invariant to translation, scale change, mirroring, and rotation, are given as follows.

$$\phi_1 = \eta_{20} + \eta_{02} \tag{4.10a}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{4.10b}$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{12} - \eta_{03})^2 \tag{4.10c}$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{4.10d}$$
$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} - 3\eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{4.10e}$$
$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned} \tag{4.10f}$$
$$\begin{aligned}\phi_7 = {} & 3(\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + 3(\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{4.10g}$$

4.2.3 Euler Number

Euler number is the difference of number of objects and the number of holes in the image. This property is affine transformation invariant. Euler number is computed by considering patterns of convexity and concavity in local 2-by-2 neighbourhoods [31]. Euler number can be calculated by using local information and does not require connectivity information. Calculating the Euler number of a binary image can be done by counting the occurrences of three types of 2x2 binary patterns.

$$P1 = \left(\begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix} , \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} , \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} , \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \right)$$

$$P2 = \left(\begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix} , \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix} , \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix} , \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \right)$$

$$P3 = \left(\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} , \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right)$$

Let $C1$, $C2$ and $C3$ be the number of occurrences of patterns $P1$, $P2$ and $P3$ respectively, then the Euler Number for the image with 4- and 8-connectivity is simply given as [31],

$$E_4 = \frac{1}{4}(C1 - C2 + 2C3) \tag{4.11}$$
$$E_8 = \frac{1}{4}(C1 - C2 - 2C3) \tag{4.12}$$

4.2.4 Normalized Area of Character Skeleton

The area of the object within the binary image is simply the count of the number of pixels in the object for which $f(x, y) = 1$. Normalized regional area is the ratio of the number of the pixels in the skeleton of binary image to the total number of pixel in the image.

4.2.5 Centroid of Image

Centroid specifies the center of mass of the object region in given image. Horizontal centroid coordinate is calculated by dividing the sum of all horizontal positions of the object which are non zero with area of the object. Similarly, vertical centroid coordinate is calculated by dividing the sum of all vertical positions of the object which are non zero with area of the object. Centroid coordinates of the binary image $f(x, y)$ of size MxN are given in equation 4.13,

$$centroid_x = \frac{m_{10}}{m_{00}} = \frac{\frac{1}{N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} x_j f(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f(i,j)} \tag{4.13a}$$

$$centroid_y = \frac{m_{01}}{m_{00}} = \frac{\frac{1}{M}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} y_j f(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f(i,j)} \tag{4.13b}$$

4.2.6 Eccentricity

The eccentricity is the ratio of the distance between the foci of the ellipse that best fit the character object and its major axis length. Eccentricity describes the rectangularity of the region of the object. The measure of eccentricity can be obtained by using the minor and major axes of such an ellipse [32]. This method of calculating eccentricity uses the second order moments. The eccentricity of the ellipse is given by,

$$e = \sqrt{\frac{\alpha^2 - \beta^2}{\alpha^2}} \tag{4.14}$$

where α and β are semi-major axis and semi-minor axis of the ellipse respectively, which are given by,

$$\alpha = \sqrt{\frac{2[\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}]}{\mu_{00}}} \tag{4.15}$$

$$\beta = \sqrt{\frac{2[\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}]}{\mu_{00}}} \tag{4.16}$$

where μ_{pq} is as described in equation 4.6.

4.3 Training and Testing

After the feature extraction phase, process of training and testing begins. In the training phase, recognition system learns patterns of different classes from input feature vectors. The learning is done in supervised manner. After the training phase, system is ready to test in unknown environment. Recognition system is then tested against testing feature vectors and accuracy and efficiency of the system is calculated. In this research, recognition is carried out using two neural network algorithms, multilayer feedforward neural network with Levenberg-Marquardt (LM) and Gradient Descent with Momentum & Adaptive Learning Rate (GDMA) learning and radial basis function networks with Orthogonal Least Square (OLS) learning. Section 4.3.1 describes the MLP algorithm and section 4.3.2 describes the RBF algorithm.

4.3.1 Multilayer Feedforward Backpropagation Network

A multilayer feedforward neural network consists of a layer of input units, one or more layers of hidden units, and one layer of output units. A neural network that has no hidden units is called a Perceptron. However, a perceptron can only represent linear functions, so it isn't powerful enough for the kinds of applications we want to solve. On the other hand, a multilayer feedforward neural network can represent a very broad set of non-linear functions. So, it is very useful in practice. This multilayer architecture of Network determines how the input is processed. The network is called feedforward because the output from one layer of neurons feeds forward into the next layer of neurons. There are never any backward connections, and connections never skip a layer. Typically, the layers are fully connected, meaning that all units at one layer are connected with all units at the next layer. So, this means that all input units are connected to all the units in the layer of hidden units, and all the units in the hidden layer are connected to all the output units.

Usually, determining the number of input units and output units is clear from the application. However, determining the number of hidden units is a bit of an art form, and requires experimentation to determine the best number of hidden units. Too few hidden units will prevent the network from being able to learn the required function, because it will have too few degrees of freedom. Too many hidden units may cause the network to tend to over-fit the training data, thus reducing generalization accuracy.

Consider a multilayer feed-forward network as shown in Figure 4.8. The net input to unit I in layer $k+1$ is given by,

$$n_i^{k+1} = \sum_{i=1}^{Sk} w_{ij}^{k+1} a_j^k + b_i^{k+1} \tag{4.17}$$

where, Sk is the number of neurons in the k^{th} layer, w_{ij}^{k+1} is the weight from j^{th} neuron of layer k to neuron i of the layer $k+1$, a_j^k is the output of neuron j from the layer k and b_i^{k+1} is the bias connected to i^{th} neuron in the layer $k+1$.

The output of unit I will be,

$$a_i^{k+1} = f^{k+1}(n_i^{k+1}) \tag{4.18}$$

where, $f^{k+1}(\cdot)$ is the activation function (see appendix A.2) used in layer $k+1$.

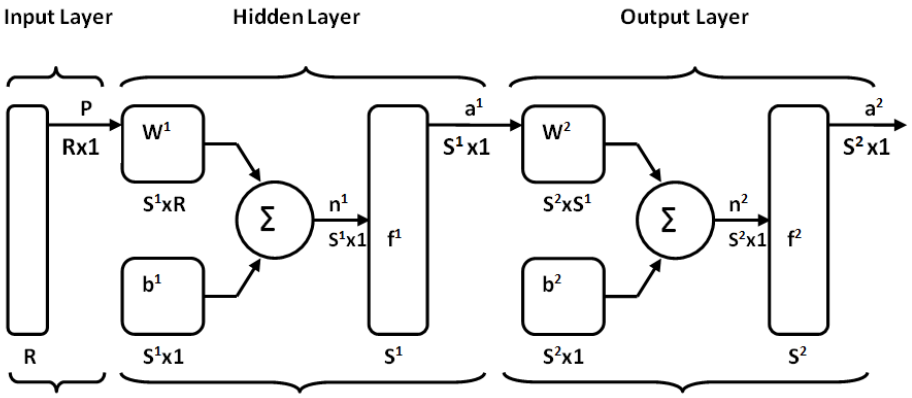

Figure 4.8: Feedforward Multilayer Perceptron.

For a M layer network the system equations in the matrix form are given by

$$a^0 = p \tag{4.19}$$

$$a^{k+1} = f^{k+1}(W^{k+1}a^k + b^{k+1}), k = 0, 1, 2, ..., M-1 \tag{4.20}$$

where, p is the input layer of the network.

The network learns association between a given set of input-output pairs $(p_i, t_i), i = 1, 2, ..., Q$ for Q number of training samples.

index for the network is given as,

Now, next step of calculating feed forward quantities for all layers in the network is the back propagation training. In this phase, we decide how neural network learn weights and biases with minimizing the generalization error. The quantity of weight and bias to adjust in the network parameters is send back to network from output layer to first hidden layer. In this research work, Levenberg-Marquardt Backpropagation Learning and Gradient Descent with Momentum & Adaptive Learning Rate are applied for learning the network parameters. Levenberg-Marquardt Backpropagation Learning algorithm is described in the section 4.3.1.1 and Gradient Descent with Momentum & Adaptive Learning rate algorithm is described in the section 4.3.1.2.

4.3.1.1 Levenberg-Marquardt Learning Algorithm

Many algorithms focus on standard numerical optimization, i.e, using alternative methods to heuristic based optimizations for computing the weights associated with network connections. The most popular algorithms for this optimization are the conjugate gradient and Newtons methods. Newtons method is considered to be more efficient in the speed of convergence, but its storage and computational requirements go up as the square of the size of the network. The LM algorithm is efficient in terms of high speed of convergence and reduced memory requirements compared to the two previous methods. In general, with networks that contain up to several hundred weights, the LM algorithm has the fastest convergence [33].

For LM , the performance index to be minimized is defined as,

$$F(w) = \sum_{p=1}^{Q}\sum_{k=1}^{K}(t_{kp} - a_{kp})^2 \tag{4.21}$$

where, $w = [w_1\ w_2\ w_3\ ...\ w_N]^T$ consists of all the weights of the network (including bias), t_{kp} is the target value of the k^{th} output and p^{th} input pattern, a_{kp} is the actual value of the k^{th} output and p^{th} input pattern, N is the number of weights, and K is the number of the network outputs.

Equation (4.21) can be written as,

$$F(w) = E^T E \tag{4.22}$$

where $E = [e_{11}\ ...\ e_{K1}\ e_{12}\ ...\ e_{K2}\ ...\ e_{1P}\ ...\ e_{KQ}]^T$,

$e_{kp} = t_{kp} - a_{kp}$, $k = 1, 2, ..., K$, $p = 1, 2, ..., Q$ where E is the cumulative error vector (for all input patterns). From equation (4.22) the Jacobian matrix is defined as,

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \cdots & \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} & \cdots & \frac{\partial e_{21}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{K1}}{\partial w_1} & \frac{\partial e_{K1}}{\partial w_2} & \cdots & \frac{\partial e_{K1}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1Q}}{\partial w_1} & \frac{\partial e_{1Q}}{\partial w_2} & \cdots & \frac{\partial e_{1Q}}{\partial w_N} \\ \frac{\partial e_{2Q}}{\partial w_1} & \frac{\partial e_{2Q}}{\partial w_2} & \cdots & \frac{\partial e_{2Q}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{KQ}}{\partial w_1} & \frac{\partial e_{KQ}}{\partial w_2} & \cdots & \frac{\partial e_{KQ}}{\partial w_N} \end{bmatrix} \tag{4.23}$$

The increment of weights Δw at iteration t can be obtained as follows:

$$\Delta w = -(J^T J + \mu I)^{-1} J^T E \tag{4.24}$$

where, I is identity matrix, μ is a learning parameter and J is Jacobian of K output errors with respect to N weights of the neural network.

Now, network weights are calculated using the following equation,

$$w_{t+1} = w_t + \Delta w \tag{4.25}$$

For $\mu = 0$, it becomes the Gauss-Newton method. For very large value of μ the LM algorithm becomes the steepest decent algorithm. The learning parameter μ is automatically adjusted at each iteration to meet the convergence criteria. The LM algorithm requires computation of the Jacobian matrix J at each iteration and inverse of $J^T J$ matrix of dimension NxN. This large dimensionality of LM algorithm is sometimes unpleasant to large size neural network.

The learning parameter μ is updated in each iteration as follows. Whenever the $F(w)$ value is decreases, μ is multiplied by decay rate β_{inc}, whenever, $F(w)$ is increases, μ is divided by decay rate β_{dec} in new step.

The LM Back-propagation training is illustrated in the Algorithm 4.4.

Algorithm 4.4 Levenberg-Marquardt Back-propagation Algorithm

1: Initialize the weights using Nguyen-Widrow weight initialization algorithm (see Appendix A.1) and parameter μ.
2: Compute the sum of the squared errors over all inputs $F(w)$.
3: Solve Eq.(4.24) to obtain the increment of weights.
4: Recompute the sum of squared errors $F(w)$ using $w + \Delta w$ as trial w, and adjust
5: **if** trial $F(w) \leq F(w)$ in Step 2 **then**
6: $w = w + \Delta w$
7: $\mu = \mu.\beta_{inc}$
8: Goto Step 2.
9: **else**
10: $\mu = \frac{\mu}{\beta_{dec}}$
11: Goto Step 4.
12: **end if**

4.3.1.2 Gradient descent with momentum and adaptive learning rate

Gradient descent with momentum and variable learning rate is heuristic based optimization technique, developed from an analysis of the performance of the standard steepest descent algorithm, used for learning the feed-forward back-propagation neural network. It has a good convergence capacity than other heuristic based technique.

The term momentum controls feedback loop and help to overcome at the situation of noisy gradient, so that local minima can be passed without stoking. Momentum Adds a percentage of the last movement to the current movement. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface.

The term adaptive learning rate plays important role for fast convergence. The learning rate automatically changes while learning the patterns. For each iteration of learning, if the performance decreases toward the goal, then the learning rate is increased by the factor η_{inc}. If the performance increases by more than the factor η_{max_inc}, the learning rate is adjusted by the factor η_{dec} and the change that increased the performance is not made.

To learn the association between a given set of input-output pairs $(p_i, t_i), i = 1, 2, ..., Q$ for Q number of training samples, the performance index (given in Eq.4.21) should be minimized.

The gradient of error function is the partial derivative of error function with respect to network weights and biases. It is given as,

$$\nabla F = \frac{\delta F}{\delta w_{ij}^k} \tag{4.26}$$

Change in weight for iteration $t + 1$ is given by,

$$\Delta w(t + 1) = \mu \Delta w(t) + (1 - \mu)\eta \nabla F \tag{4.27}$$

where, μ is the momentum constant and η is the learning rate.

Now, Weight is adjusted by,

$$w(t + 1) = w(t) + \Delta w(t + 1) \tag{4.28}$$

4.3.2 Radial Basis Function Network

Radial-basis function network is used to perform a complex pattern classification task, the problem basically solved by transforming it into a high dimensional space in a non-linear manner. Non linear separability of complex patterns is justified by Cover's theorem,which stated as, "A complex pattern-classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space".

Radial basis functions are embedded into a two layer feed-forward network. Such a network is characterized by a set of inputs and set of outputs. In between the inputs and outputs, there is a layer of processing units called hidden units. Each of them implements a radial basis function. Generally, there is a single hidden layer. In pattern classification problem, inputs represent feature values, while each output corresponds to a class. A general RBF Neural Network Architecture is given in Figure 4.9.

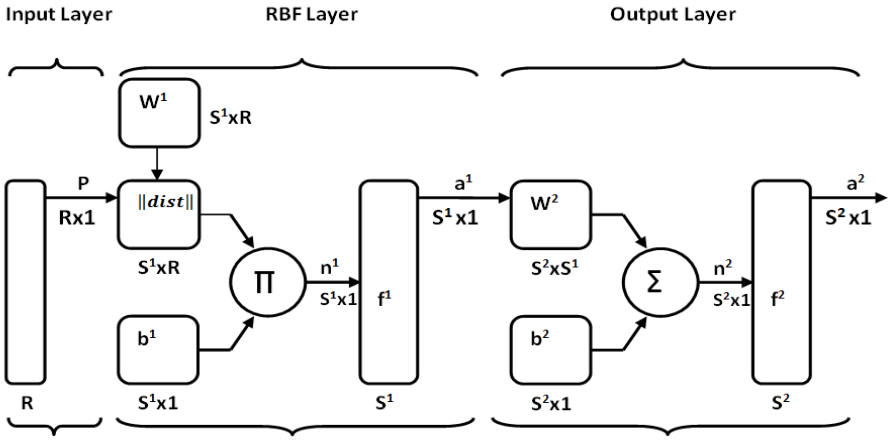

Figure 4.9: RBF Neural Network.

Let the input vector $x = [x_1, x_2, ..., x_R]^T$ is feed to RBF Network. The network output can be obtained by,

$$t_k = \sum_{j=1}^{M} w_{kj}\phi_j(x) + b_k \tag{4.29}$$

where $\phi_j(\cdot)$ denotes the radial basis function of the j^{th} hidden neuron, w_{kj} denotes the hidden-to-output weight from j^{th} hidden neuron to k^{th} output neuron, b_k denotes the bias connected to k^{th} output neuron, and M denotes the total hidden neurons.

A radial basis function is a multidimensional function that describes the distance between a given input vector and a pre-defined centre vector. There are different types of radial basis function. A normalized Gaussian function usually used as a radial basis function, which is given as,

$$\phi_j(x) = \exp(-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}) \tag{4.30}$$

where, μ_j and σ_j denote the centre and spread width of the j^{th} neuron, respectively.

Generally, the Radial Basis Function Neural Network (RBFNN) training can be divided into two stages:

1. Determine the parameters of radial basis functions, i.e., Gaussian centre and spread width.
2. Determine the output weight w with supervised learning method. Usually Least Mean Square (LMS).

The first stage is very crucial, since the number location of centres in the hidden layer will influence the performance of the RBF neural network directly. In the next section, the principle and procedure of self-structure RBF algorithm will be described.

4.3.2.1 Orthogonal Least Square Training Algorithm

Orthogonal Least Squares is the most widely used supervised algorithm for radial basis neural network training. OLS is a forward stepwise regression procedure. Starting from a large pool of candidate centres, i.e., from training examples, OLS sequentially selects the centre that results in the largest reduction of sum-square-error at the output.

For M basis vectors $\Phi = (\phi_1, \phi_2, ..., \phi_M)$, orthogonal set $Q = (q_1, q_2, ..., q_M)$ can be derived with the help of Grahm-Schmidt orthogonalization procedure as follows:

$$q_1 = \phi_1 \tag{4.31}$$

$$\alpha_{ik} = \frac{q_i^T \phi_k}{q_i^T q_i} \; 1 \leq i \leq k \tag{4.32}$$

$$q_k = \phi_k - \sum_{i=1}^{k-1} \alpha_{ik} q_i \tag{4.33}$$

$$for \; k = 2, 3, ..., M$$

The orthogonal set of basis vectors Q is linearly related to the original set Φ by the relationship $\Phi = QA$ [34], where A is upper triangular matrix given by,

$$A = \begin{bmatrix} 1 & \alpha_{12} & ... & \alpha_{1M} \\ 0 & 1 & & ... \\ ... & & & \alpha_{M-1,M} \\ 0 & ... & 0 & 1 \end{bmatrix} \tag{4.34}$$

Using this orthogonal representation, the RBF solution is expressed as,

$$T = \Phi W = QG \tag{4.35}$$

and the Least Square (LS) solution for the weight vector G in the orthogonal space is,

$$G = (Q^T Q)^{-1} Q^T T \tag{4.36}$$

Since Q is orthogonal, $Q^T Q$ is then diagonal and each component of G can be extracted independently without ever having to compute a pseudo-inverse matrix as,

$$g_i = \frac{q_i^T T}{q_i^T q_i} \tag{4.37}$$

The sum of squares of the target vector T is given as,

$$T^T T = \sum_{i=1}^{M} g_i^2 q_i^T q_i + E^T E \tag{4.38}$$

OLS select a subset of regressors in stepwise forward manner by selecting the regressor q_i,which contributes to reduction of the error (relative to the total $T^T T$) by,

$$[err]_i = \frac{g_i^2 q_i^T q_i}{T^T T} \tag{4.39}$$

The summary of Orthogonal Least Square RBF training [34] is given in Algorithm 4.5.

Algorithm 4.5 OLS Algorithm

1: At the first step,for $1 \leq i \leq M$, compute
2: Orthogonalize vector: $q_1^{(i)} = \phi_i$
3: Compute LS Solution: $g_1^{(i)} = \frac{{q_1^{(i)}}^T T}{{q_1^{(i)}}^T q_1^{(i)}}$
4: Compute error reduction: $[err]_i^{(1)} = \frac{{g_1^{(i)}}^2 {q_1^{(i)}}^T q_1^{(i)}}{T^T T}$
5: select regressor that yields highest reduction in error $q_1 = \arg_{q_1^{(i)}} \max [err]_1^{(i)} = \phi_{i_1}$
6: At the k^{th} step, for $1 \leq i \leq M$, and i not already selected
7: Orthogonalize vector: $\alpha_{jk}^{(i)} = \frac{q_j^T \phi_i}{q_j^T q_j}$, $q_k^{(i)} = \phi_i - \sum_{i=1}^{k-1} \alpha_{jk}^{(i)} q_j$, $for 1 \leq j \leq k$
8: Compute LS Solution: $g_k^{(i)} = \frac{{q_k^{(i)}}^T T}{{q_k^{(i)}}^T q_k^{(i)}}$
9: Compute error reduction: $[err]_k^{(i)} = \frac{{g_k^{(i)}}^2 {q_k^{(i)}}^T q_k^{(i)}}{T^T T}$
10: select regressor $q_k = \arg_{q_k^{(i)}} \max [err]_k^{(i)} = \phi_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} q_j$
11: Stop at iteration M if residual error falls below pre-specified threshold ϵ
12: $1 - \sum_{j=1}^{M} [err]_j \leq \epsilon$
13: The regressors $\{\phi_{i_1}, \phi_{i_2}, \cdots, \phi_{i_M}\}$ define the final subset of RBF centers.

4.3.3 Performance Metrics

Performance of the recognition systems is evaluated on the basis of the metrics described below:

1. **Recognition Rate:** Number of instances that were classified correctly out of the total instances.
2. **Misclassification Rate:** Number of instances that were classified incorrectly out of the total instances.
3. **Model Build Time:** Time taken to train a classifier on a given data set.

Chapter 5

IMPLEMENTATION

All the algorithms of purposed handwriting recognition system are implemented in MATLAB ®version 7.13.0.564 (R2011b). MATLAB is installed on a Intel(R) Core(TM)2 Duo CPU T6500 @ 2.10GHz processor. The Computer has total main memory of 2 Gigabyte and 32-bit Microsoft Windows 7 Ultimate operating system installed in it.

5.1 MATLAB

The name MATLAB stands for MATrix LABoratory. MATLAB is high-level technical computing language having high-performance computation. MATLAB started life in the 1970s as a user-friendly interface. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB has very wide range of applications and can now be used as a very simple and transparent programming language where each line of code looks very much like the mathematical statement it is designed to implement. Typical uses of the MATLAB are given below.

- Mathematical and scientific computation.
- Algorithm development.
- Application development, including Graphical User Interface building.
- Data analysis, exploration and visualization.
- Modelling, simulation and prototyping.
- Graphics, in 2D and 3D, including colour, lighting and animation.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows solving many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering and science. In industry, MATLAB is the tool of choice for high-productivity research, development and analysis.

The reason behind using MATLAB for my research work experimentation is its wide variety and flexible toolboxes. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. In this research work, mainly image processing toolbox and neural network toolbox are used along with other basis functionality of the MATLAB.

5.2 Image Processing Toolbox

Image Processing Toolbox (IPT) is very rich MATLAB toolbox for working with images. It includes many powerful and very efficient image processing functions. It provides a comprehensive set of standard algorithms and graphical tools for image processing, analysis, visualization and algorithm development. We can use IPT to perform complex image processing tasks like image enhancement, image deblurring, feature detection, noise reduction, image segmentation, geometric transformations, image registration, etc. Many toolbox functions are multithreaded to take advantage of multicore and multiprocessor computers.

Here, IPT is used for image preprocessing techniques like noise removal, segmentation, skeletonization, etc. and for some feature extraction techniques (eg. euler number). It is also used for visualizing and analysing the results of image preprocessing and feature extraction steps.

5.3 Neural Network Toolbox

Neural Network Toolbox (NNT) provides tools for designing, implementing, visualizing, and simulating neural networks. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections between elements. Neural networks are used for applications where formal analysis would be difficult or impossible, such as pattern recognition, identification, classification, speech, vision and control systems. Neural Network Toolbox supports feedforward networks, radial basis networks, dynamic networks, self-organizing maps, and other proven network paradigms. NNT provides very customizable neural network functions which we can adjust according to our need. A typical neural network training contains following steps:

1. Loading data source.
2. Selecting attributes required.
3. Decide training, validation, and testing data.
4. Data manipulations and Target generation.
5. Neural Network creation and initialisation.
6. Network Training and Testing.
7. Performance evaluation.

In this research, NNT is used for multilayer feedforward backpropagation network and radial basis function network training algorithms.

5.4 Feature Vector Creation

Global feature vector is created according to the algorithms described in the section 4.2. Let us consider image shown in figure 5.1. The binary matrix of this skeletonized image is given in figure 5.2.

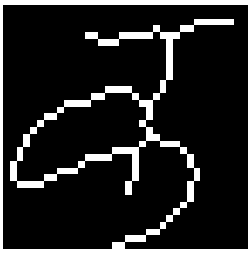

Figure 5.1: Nepali consonant letter 'ka'.

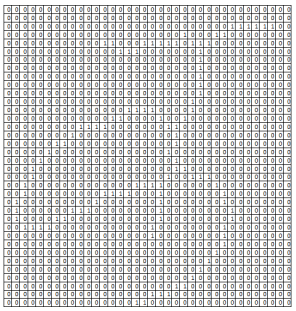

Figure 5.2: Letter 'ka' as binary matrix.

5.4.1 Directional Feature Vector

For the directional feature extraction, image is zoned into 3x3 sub-images. Figure 5.3 shows zoned images of image given in the figure 5.1.

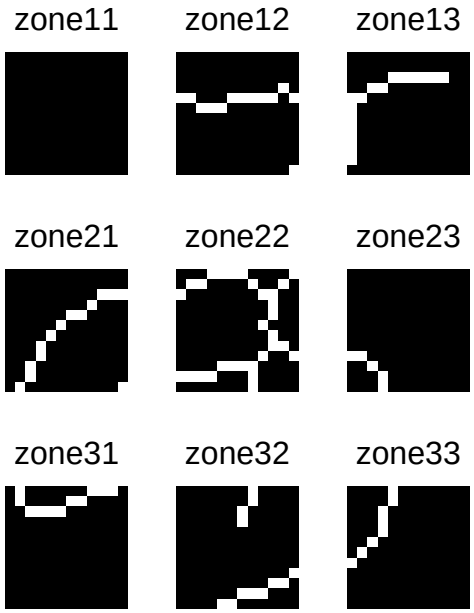

Figure 5.3: Zoned images

Normalized feature vectors corresponding to each zone are given in Table 5.1.

zone11 feature vector	[-1,-1,-1,-1,-1,-1,-1,-1,-1]
zone12 feature vector	[0.80,1,1,1,0.85,0,0,0,1]
zone13 feature vector	[0.80,1,1,1,0.85,0,0,0,1]
zone21 feature vector	[1,1,0.80,1,0,0,0.86,0,1]
zone22 feature vector	[0.60,0.80,0.80,0.80,0.63,0.03,0.06,0.03,0.40]
zone23 feature vector	[1,1,1,0.80,0,0,0,0.80,1]
zone31 feature vector	[0.80,1,1,1,0.90,0,0,0,1]
zone32 feature vector	[0.80,0.80,1,1,0.58,0.25,0,0,1]
zone33 feature vector	[1,1,0.80,1,0,0,0.87,0,1]

Table 5.1: Directional feature vectors corresponding to figure 5.3.

5.4.2 Moment Invariant Feature Vector

Moment invariant features are extracted from whole character image. It implements the algorithm given in section 4.2.2. Moment invariant features corresponding to figure 5.1 are given below,

[1.3937, 0.1858, 0.8792, 0.0024, 0.0001, 0.0009, -0.0001]

5.4.3 Euler Number Feature Vector

There is a single entry for Euler number in global feature vector. An Euler number corresponding to figure 5.1 is 0.

5.4.4 Normalized Area of Character Skeleton Feature Vector

There is a single entry for normalized area of character image in global feature vector. Normalized area corresponding to figure 5.1 is 0.0841. Normalized area is calculated by following formula.

$$normalized\_area = 1 - \frac{area}{image\_size}$$

5.4.5 Centroid Feature Vector

Centroid is also extracted from whole character image. There are two entries for centroid in global feature vector. Normalized centroid corresponds to figure 5.1 is given below,

$$[0.9857, 0.9860]$$

5.4.6 Eccentricity Feature Vector

There is single entry in global feature vector for eccentricity. Eccentricity corresponds to figure 5.1 is 0.6870.

5.4.7 Global Feature Vector

Global feature vector is created by appending all above feature vectors to a single feature vector. There is 93 dimensional global feature vector for each character images. The global feature corresponds to figure 5.1 is given below,

input_vector=[-1, -1, -1, -1, -1, -1, -1, -1, -1, 0.80, 1, 1, 1, 0.85, 0, 0, 0, 1, 0.80, 1, 1, 1, 0.87, 0, 0, 0, 1, 1, 1, 0.80, 1, 0, 0, 0.86, 0, 1, 0.60, 0.80, 0.80, 0.80, 0.63, 0.03, 0.07, 0.03, 0.40, 1, 1, 1, 0.80, 0, 0, 0, 0.80, 1, 0.80, 1, 1, 1, 0.90, 0, 0, 0 1, 0.80, 0.80, 1, 1, 0.58, 0.25, 0, 0, 1, 1, 1, 0.80, 1, 0, 0, 0.87, 0, 1, 0.999, 0.999, 0.999, 0.999, 0.999, 0.999, 1.00, 0, 0.08, 0.98, 0.98, 0.69].

To create target vector corresponding to above global feature vector, we create a vector with dimensionality equal to output classes. For figure 5.1, since it belongs to Nepali consonant class, we have 36 dimension target vector with first entry equal to 1 and all other entries equal to 0. So, the target vector corresponds to figure 5.1 is given by,

target_vector=[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Now, above feature vector along with corresponding target vector is ready to feed into neural network for training and testing. Training is done in supervised manner with two neural network algorithms, MLP and RBF networks.

Chapter 6

NEPALI HANDWRITTEN DATASETS

To do experimentation with this research work, benchmark datasets for Off-line Nepali Handwritten Characters are created. There are three separate datasets for Nepali handwritten consonants, vowels and numerals. The overall dataset creation procedure is discussed in section 6.1. More detail about Nepali consonant dataset, vowel dataset and numeral dataset is given in the section 6.2, section 6.3 and section 6.4, respectively. Training and testing samples for recognition system are selected from each dataset in certain ratio.

6.1 Dataset Creation Procedure

Datasets are created by taking handwriting samples from different writers from different fields. The overall procedure for creating off-line Nepali handwritten character datasets is described in the Algorithm 6.1.

Algorithm 6.1 Handwritten Dataset Creation

1: Take handwritten samples from different writers in A4 paper.
2: Scan the sample documents using digital scanner.
3: Slice and crop individual characters from each documents.
4: Make individual directory for each category of datasets.
5: Make individual class directory in each dataset directory.
6: Drag individual characters in the corresponding class directory.
7: Datasets are ready.

6.2 Nepali Handwritten Consonant Dataset

Nepali language contains 36 different Devanagari alphabets for Consonants. Figure 6.1 shows Nepali handwritten consonants. To create training and testing dataset of Nepali handwritten consonants, handwritten consonant images are scanned with digital scanner and cropped for individual characters.

Nepali handwritten consonant dataset contains total **7380** image samples. Samples are taken from 45 different writers from different fields. Each class of consonant dataset contains 205 images with varying styles and shapes.

Figure 6.1: Nepali Handwritten Consonants.

6.3 Nepali Handwritten Vowel Dataset

Nepali language contains 12 different Devanagari alphabets for vowels. Figure 6.2 shows Nepali handwritten vowels. To create training and testing dataset of Nepali handwritten vowels, handwritten vowel samples are scanned with digital scanner and cropped for individual characters.

Nepali handwritten vowel dataset contains total **2652** image samples. Samples are taken from 44 different writers from different fields. Each class of vowel dataset contains 221 images with varying styles and shapes.

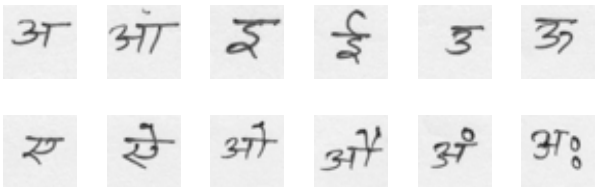

Figure 6.2: Nepali Handwritten Vowels.

6.4 Nepali Handwritten Numeral Dataset

Nepali language contains 10 different Devanagari numerals. Figure 6.3 shows Nepali handwritten numerals. To create training and testing dataset of Nepali handwritten numerals, handwritten numeral images are scanned with digital scanner and cropped for individual characters.

Nepali handwritten numeral dataset contains total **2880** image samples. Samples are taken from 45 different writers from different fields. Each class of numeral dataset contains 288 images with varying styles and shapes.

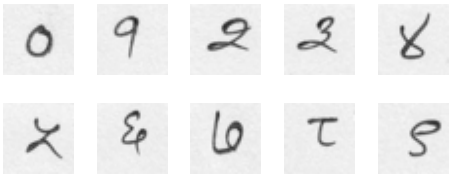

Figure 6.3: Nepali Handwritten Numerals.

Chapter 7

EXPERIMENTATIONS AND RESULTS

The aim of this research work was to evaluate the complete MLP and RBF neural network handwriting recognition systems. Experiment is done with off-line isolated Nepali handwritten alphabets and numerals. There are three separate datasets for Nepali consonants, vowels and numerals. First of all, system is trained in each dataset using different random samples from each dataset in supervised manner. And then system is tested against new samples and accuracy is measured. In every datasets the data was partitioned into three parts, for training, validation and testing. Every set has completely different samples, which are selected randomly in uniformly manner from the pool of given dataset. Experimentation with each recognition systems is carried out five times in each datasets. The accuracy and efficiency for each datasets is the average of all the experimentation results for that dataset. Feature vectors used for training and testing is as described in section 5.4. The network configurations and recognition results for MLP and RBF handwriting recognition systems for each datasets are described in this chapter.

Table 7.1 shows neural network configurations for all three datasets on MLP and RBF recognition systems. Training and testing samples are uniformly distributed random samples. The remaining samples in each datasets other than training and testing samples are used as validation samples. Validation is used for generalization and early stopping the recognition system while training.

Dataset Name	**Dataset Size (Class * Samples)**	**Recognition Algorithm**	**Train/Test Samples**	**Hidden Layer Neurons**	**No. of Epochs**
Numeral Dataset	10 * 288 = 2880	MLP (LM)	2016/432	30	9
		RBF	2304/576	580	580
Vowel Dataset	12 * 221 = 2652	MLP (LM)	1857/397	30	12
		RBF	2122/530	840	840
Consonant Dataset	36 * 205 = 7380	MLP (GDMA)	5411/891	100	1000
		RBF	5166/891	1025	1025

Table 7.1: Neural Network Configurations.

Table 7.2 shows performance matrices for MLP and RBF based off-line handwriting recognition systems on each datasets. The best recognition result is obtained with RBF based recognition system on off-line Nepali numeral dataset. In each datasets the RBF based recognition system outperforms MLP based recognition system. Training time, recognition accuracy and miss-classification rates for each dataset experimentation are given in Table 7.2.

Dataset Name	**Recognition Algorithm**	**Training Time (min.)**	**Recognition Accuracy (%)**	**Miss-classification Rate (%)**
Numeral Dataset	MLP	16.66	87.50	12.50
	RBF	69.91	**94.44**	5.55
Vowel Dataset	MLP	24.36	79.15	20.85
	RBF	113.23	**86.04**	13.96
Consonant Dataset	MLP	13.92	71.72	28.28
	RBF	308.4	**80.25**	19.75

Table 7.2: Recognition Results.

Figure 7.1 shows the graph of recognition system accuracy on all three datasets.

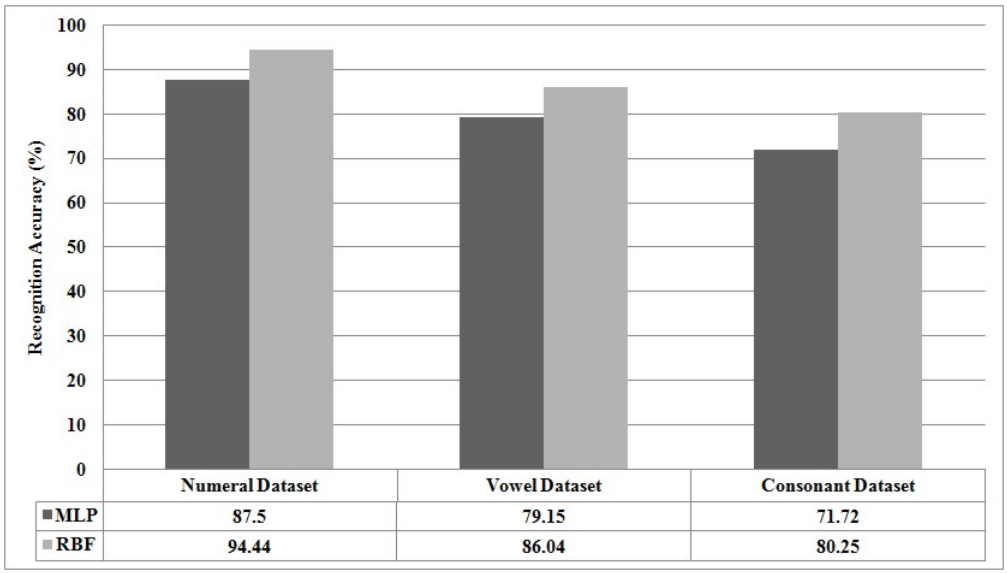

Figure 7.1: Recognition Accuracy of Off-line Handwriting Recognition Systems.

Figure 7.2 shows the graph of training time for each recognition systems in each datasets.

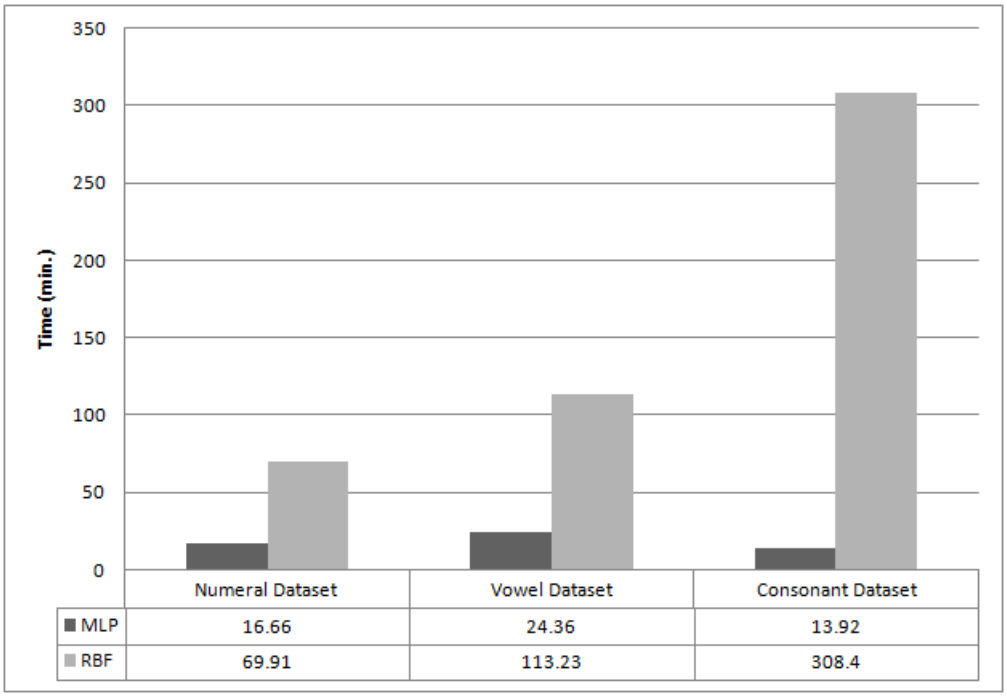

Figure 7.2: Training Time of Off-line Handwriting Recognition Systems.

Table 7.3 shows individual recognition results for Nepali handwritten numeral dataset for each classes. The corresponding Confusion Matrix (CM) for test set of numeral dataset is given in table 7.4. The numeral test set contains 576 random samples for testing. Here, CM is an 10x10 matrix with $CM(i, j)$ is the number of target samples of the i^{th} class classified by the outputs into class j.

Class	Recognition Algorithms	Recognition Accuracy (%)	Class	Recognition Algorithms	Recognition Accuracy (%)
०	MLP	97.7	९	MLP	94.1
	RBF	98.6		RBF	98.4
२	MLP	85.1	३	MLP	82.1
	RBF	93.0		RBF	90.4
४	MLP	82.5	५	MLP	82.5
	RBF	98.0		RBF	81.1
६	MLP	82.9	७	MLP	86.7
	RBF	95.0		RBF	96.2
८	MLP	97.8	९	MLP	80.4
	RBF	100		RBF	90.9

Table 7.3: Recognition Results For Individual Numerals.

Figure 7.3 shows the graph of recognition accuracy for each classes of numeral dataset.

Class	०	१	२	३	४	५	६	७	८	९
०	69	0	0	0	0	0	0	0	0	1
१	0	61	0	0	0	0	1	0	0	0
२	0	0	53	4	0	0	0	0	0	0
३	0	0	3	47	0	0	2	0	0	0
४	0	0	0	0	49	1	0	0	0	0
५	0	0	5	3	1	43	0	0	0	1
६	0	1	0	0	0	0	57	1	0	0
७	0	0	0	0	1	0	1	51	0	0
८	0	0	0	0	0	0	0	0	64	0
९	0	1	1	1	0	1	1	0	0	50

Table 7.4: Confusion Matrix of Numeral Dataset Testing.

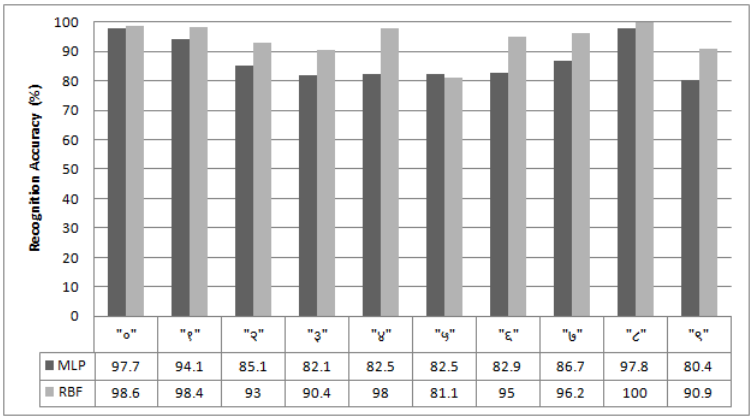

Figure 7.3: Recognition Accuracy of Each Numeral Classes.

Table 7.5 shows individual recognition results for Nepali handwritten vowel dataset for each classes. The corresponding CM for test set of vowel dataset is given in table 7.6. The vowel test set contains 530 random samples for testing. Here, CM is an 12x12 matrix with $CM(i, j)$ is the number of target samples of the i^{th} class classified by the outputs into class j.

Figure 7.4 shows the graph of recognition accuracy for each classes of vowel dataset.

Table 7.7 shows individual recognition results for Nepali handwritten consonant dataset for each classes. The corresponding CM for test set of consonant dataset is given in table 7.8 and table 7.9 . The consonant test set contains 891 random samples for testing. Here, CM is an 36x36 matrix with $CM(i, j)$ is the number of target samples of the j^{th} class classified by the outputs into class i.

Figure 7.5 shows the graph of recognition accuracy for each classes of consonant dataset.

Class	Recognition Algorithms	Recognition Accuracy (%)	Class	Recognition Algorithms	Recognition Accuracy (%)
अ	MLP	77.8	आ	MLP	93.5
	RBF	78.9		RBF	87.8
इ	MLP	84.2	ई	MLP	88.5
	RBF	91.1		RBF	90.5
उ	MLP	75.8	ऊ	MLP	85.2
	RBF	92.1		RBF	87.8
ए	MLP	78.4	ऐ	MLP	82.9
	RBF	97.6		RBF	85.7
ओ	MLP	71.4	औ	MLP	64.7
	RBF	76.5		RBF	80.0
अं	MLP	64.1	अः	MLP	86.5
	RBF	76.1		RBF	88.5

Table 7.5: Recognition Results For Individual Vowels.

Class	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	अं	अः
अ	45	6	0	0	0	4	0	0	1	0	1	0
आ	3	36	0	0	0	0	0	0	0	1	0	1
इ	0	0	51	0	1	0	0	0	1	1	2	0
ई	0	0	2	38	0	0	0	1	0	0	1	0
उ	1	0	2	0	35	0	0	0	0	0	0	0
ऊ	2	0	0	0	1	36	0	0	0	0	0	2
ए	1	0	0	0	0	0	40	0	0	0	0	0
ऐ	1	0	0	0	0	0	2	36	2	0	1	0
ओ	0	0	0	0	0	0	0	1	26	5	2	0
औ	0	0	0	0	0	0	0	0	6	32	2	0
अं	1	0	0	2	0	0	0	1	5	2	35	0
अः	1	3	0	0	0	0	0	0	0	2	0	46

Table 7.6: Confusion Matrix of Vowel Dataset Testing.

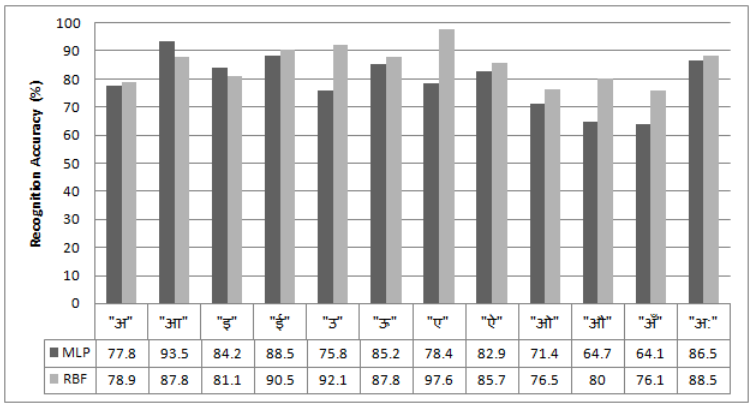

Figure 7.4: Recognition Accuracy of Each Vowel Classes.

Class	Recognition Algorithms	Recognition Accuracy (%)	Class	Recognition Algorithms	Recognition Accuracy (%)
क	MLP	94.1	ख	MLP	78.3
	RBF	94.1		RBF	91.3
ग	MLP	80.8	घ	MLP	51.9
	RBF	92.3		RBF	77.8
ङ	MLP	60.0	च	MLP	84.6
	RBF	64.0		RBF	84.6
छ	MLP	66.7	ज	MLP	73.1
	RBF	77.8		RBF	80.8
झ	MLP	65.4	ञ	MLP	59.4
	RBF	73.1		RBF	68.8
ट	MLP	96.3	ठ	MLP	63.6
	RBF	92.6		RBF	72.7
ड	MLP	76.9	ढ	MLP	91.7
	RBF	88.5		RBF	91.7
ण	MLP	85.0	त	MLP	92.3
	RBF	90.0		RBF	96.2
थ	MLP	50.0	द	MLP	64.0
	RBF	72.7		RBF	76.0
ध	MLP	75.0	न	MLP	36.8
	RBF	79.2		RBF	38.8
प	MLP	87.5	फ	MLP	72.4
	RBF	100		RBF	86.2
ब	MLP	75.9	भ	MLP	63.6
	RBF	75.9		RBF	90.9
म	MLP	66.7	य	MLP	44.0
	RBF	73.3		RBF	56.0
र	MLP	89.3	ल	MLP	66.7
	RBF	82.1		RBF	77.8
व	MLP	73.9	श	MLP	80.0
	RBF	78.3		RBF	91.4
ष	MLP	76.0	स	MLP	55.2
	RBF	80.0		RBF	79.3
ह	MLP	88.2	क्ष	MLP	60.0
	RBF	82.4		RBF	72.0
त्र	MLP	73.1	ज्ञ	MLP	64.0
	RBF	76.9		RBF	80.0

Table 7.7: Recognition Results For Individual Characters.

Class	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट	ठ	ड	ढ	ण	त	थ	द
क	16	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
ख	0	21	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
ग	0	0	24	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
घ	0	0	0	21	0	1	0	0	0	0	0	0	0	0	0	0	1	0
ङ	0	0	0	0	16	0	0	0	0	0	0	0	1	0	0	0	0	0
च	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0
छ	0	0	0	1	0	0	14	0	0	0	1	0	0	0	0	0	0	2
ज	0	0	0	0	1	0	0	21	0	1	0	0	0	0	0	0	0	0
झ	0	0	0	0	0	0	1	0	19	0	0	0	0	0	0	0	0	0
ञ	0	0	0	0	0	0	0	3	0	22	0	0	0	0	0	0	0	0
ट	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0
ठ	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	1
ड	0	0	0	0	2	0	0	0	0	0	0	1	23	1	0	0	0	0
ढ	0	0	0	0	1	0	1	0	0	0	0	1	0	22	0	0	0	1
ण	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0
त	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0
थ	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	16	0
द	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	19
ध	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	2	0
न	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
प	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0
फ	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ब	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0
भ	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
म	0	0	1	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0
य	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
र	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
ल	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
व	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0
श	0	0	1	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0
ष	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
स	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
ह	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
क्ष	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
त्र	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0
ज्ञ	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Table 7.8: Confusion Matrix (I) of Consonant Dataset Testing.

Class	ध	न	प	फ	ब	भ	म	य	र	ल	व	श	ष	स	ह	क्ष	त्र	ज्ञ
क	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
ख	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ग	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2	0
घ	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ङ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
च	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0
छ	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
ज	0	1	0	0	0	0	2	1	0	0	0	0	0	0	0	0	2	0
झ	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ञ	0	2	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0
ट	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
ठ	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
ड	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ढ	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
ण	0	0	0	1	0	0	0	2	0	1	1	0	1	0	0	0	0	0
त	0	2	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
थ	1	0	0	0	1	0	0	0	1	0	0	1	2	0	0	1	0	1
द	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0
ध	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
न	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
प	0	0	24	1	0	1	2	3	0	0	0	0	0	0	1	0	0	0
फ	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ब	0	0	0	0	22	0	0	0	0	0	1	1	0	0	0	1	0	1
भ	1	0	0	0	0	20	0	1	0	0	0	0	0	1	0	1	0	0
म	0	2	0	0	1	0	22	0	0	0	0	0	0	2	0	0	0	0
य	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0
र	0	1	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0
ल	0	1	0	0	0	0	0	1	0	14	0	0	0	0	0	0	0	2
व	0	1	0	0	1	0	0	0	0	0	18	1	0	0	0	0	0	0
श	1	0	0	0	0	0	0	0	0	0	0	32	0	2	0	0	0	0
ष	0	0	0	0	0	0	0	0	0	0	0	0	20	0	1	0	0	0
स	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	1	0	0
ह	0	0	0	0	0	0	0	0	0	0	1	0	0	0	14	0	0	0
क्ष	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	18	0	0
त्र	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	20	1
ज्ञ	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	20

Table 7.9: Confusion Matrix (II) of Consonant Dataset Testing.

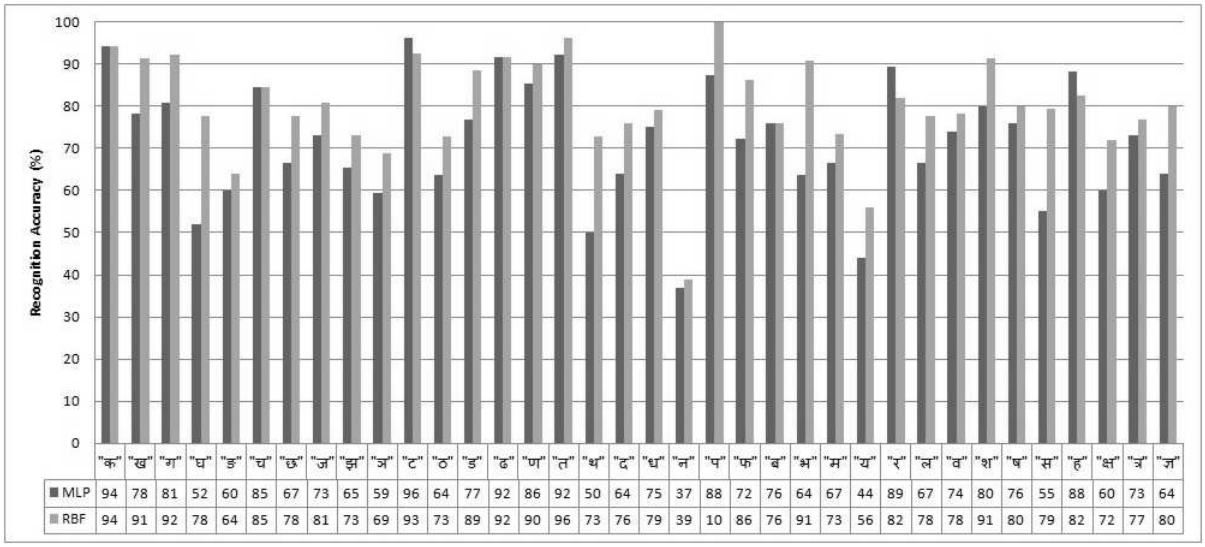

Figure 7.5: Recognition Accuracy of Each Consonant Classes.

Chapter 8

CONCLUSION

8.1 Conclusion

Off-line handwriting recognition with neural network is presented and evaluated on Nepali handwritten alphabet and numeral datasets. Two neural network based supervised classification methods (MLP and RBF) are experimented. The important preprocessing steps and feature extraction techniques for segmented characters are also described in full detail.

Off-line Nepali Handwritten Character recognition is a difficult problem, not only because of the great amount of variations in human handwriting, but also because of the overlapped and joined characters. Recognition accuracy of the system greatly depends upon the nature of the data to be recognized.

For the experimentation with the purposed off-line Nepali handwriting recognition system, self created off-line Nepali handwritten character datasets are used. Three are three well populated handwritten datasets for Nepali numerals, vowels and consonants respectively. All datasets are so created that, it can also be used as writer identification and document verification research. Nepali handwritten numeral dataset contains total **2880** image samples with 288 samples for each 10 classes of Nepali numerals, Nepali handwritten vowel dataset contains total **2652** image samples with 221 samples for each 12 classes of Nepali vowels and Nepali handwritten consonant dataset contains **7340** image samples with 205 samples for each 36 classes of Nepali consonants.

Off-line Nepali handwriting recognition system evaluated on Off-line Nepali numeral dataset gives best accuracy of **94.44%** over all other datasets. In other datasets, due to high variations on writing styles, shapes and cursive nature of characters, the recognition accuracy is decreases. But, very encouraging results are obtained while experimentation on all datasets. For Off-line Nepali vowel dataset, recognition accuracy of **86.04%** is obtained and for off-line Nepali consonant dataset, recognition accuracy of **80.25%** is obtained. In all cases, RBF based recognition system outperforms MLP based recognition system. RBF based recognition engine takes little more time for training but gives a very encouraging results while testing.

Recognition accuracy is directly proportional to the set of good features. So, feature extraction engine is the most important part of the recognition system. Good features describe the property of character image very well, which ultimately helps in learning well the recognition engine the particular pattern. For extracting good features from character image, it should be well pre-processed. So, pre-processing is also very important step of recognition. This research describes two very important category of feature extraction techniques: geometry based

features and pixel distribution based features. Geometric features are very good descriptors of images for capturing physical shape of objects and statistical features are also very important and necessary descriptors for capturing the pixel distribution of the object.

To further improve the quality of recognition system the combination of the proposed methods with additional preprocessing and feature extraction techniques are recommended. Especially, features extracting methods for shape informations which contain the semantic properties of the object have great impact on the efficiency of the recognition system.

8.2 Future Scope

Handwriting recognition is the research topic over past four decades, different methods have been explored by a large number of researchers to recognize characters. Various approaches have been proposed and tested in different parts of the world, including statistical methods, structural and syntactic methods and neural networks. No handwriting recognition system is cent percent accurate till now. The recognition accuracy of the neural networks proposed in this research work can be further improved. Large training corpus with varying writing styles can help in good generalization of the system. The future scope of this research is given below:

- Proposed system can be tested with known handwritten datasets like Germana database, IAM database, MNIST database, CEDAR database etc.
- Experimentation can be enhanced by taking other spatial domain features along with frequency domain features.
- Purposed system can be extended for the recognition of words, sentences and documents.
- Purposed system can be extended for multilingual character recognition.

Appendix A

Neural Network Parameters

A.1 NguyenWidrow Weights Initialization

ANN uses learning algorithms to train the weights and biases of the network. We normally choose random values as initial weights for the network. The random choice of initial weights and biases will affect the performance and efficiency of the learning algorithms. If the average performance of the algorithm is required, a test using several different sets of initial weights and biases will be carried out. Good choice of initial weights and biases in the training phase of the experiments greatly impact the performance of the network training. The chosen initial weights determine the starting point in the error landscape, which controls whether the learning process will end up in a local minimum or the global minimum. The easiest method is to select the weights randomly from suitable range such as between [-1.0 1.0] or minmax(input range). More sophisticated approach for selecting the initial weights for the network training is the Nguyen-Widrow weight initialization technique.

Nguyen-Widrow weight initialization technique calculates the interval from which the weights are taken in accordance with the number of input neurons and the number of hidden neurons [35]. Nguyen-Widrow weight initialization algorithm is given in algorithm A.1.

Algorithm A.1 Nguyen-Widrow Weight Initialization Algorithm

1: Calculate parameter β using,
2: $\beta = 0.7 * p^{\frac{1}{n}}$
3: where $p =$ number of hidden units and
4: $n =$ number of input units
5: Calculate weight w_{ij} as,
6: $$w_{ij} = \beta * \frac{w_{ij}(random)}{\|w_j(random)\|}$$

A.2 Activation Function

A activation function (also known as transfer function) determines the relation between input and output. Activation function normalizes the output value to certain range. There are number

of activation functions in use with neural networks. Activation functions may be linear or non-linear. Some of the widely used activation functions are McCulloch-Pitts thresholding function, piecewise-linear function and sigmoid function.

Sigmoid function has s-shaped graph and is most common form of activation function for defining the output of a neuron. Sigmoid function is strictly increasing, continuous and differential function. It exhibits a graceful balance between linear and nonlinear behavior. An example of the sigmoid function is the logistic function, defined by,

$$\phi(v) = \frac{1}{1 + \exp(-av)} \tag{A.1}$$

where, a is the slope parameter of the sigmoid function and v is the local output of the neuron.

Hyperbolic tangent sigmoid activation function is symmetric bipolar activation function, defined by,

$$\phi(v) = \frac{2}{(1 + \exp(-2v))} - 1 \tag{A.2}$$

The graph of hyperbolic tangent sigmoid activation function is given in figure A.1.

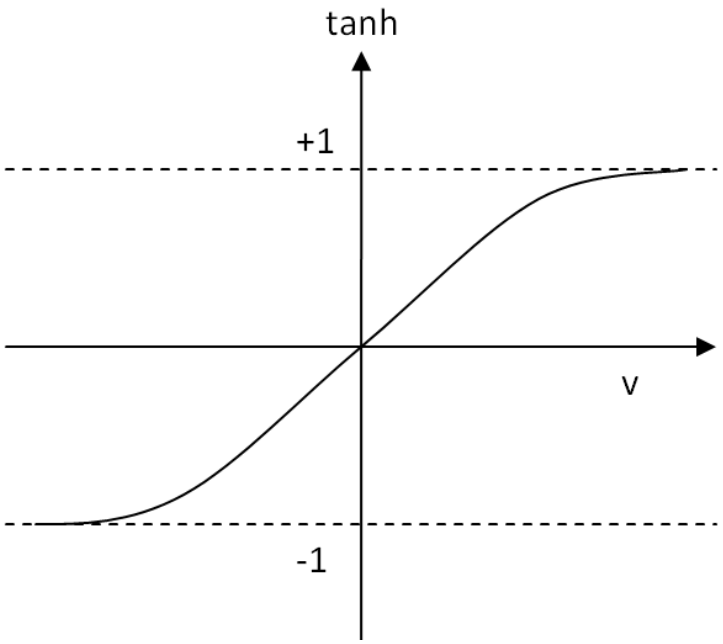

Figure A.1: Hyperbolic tangent sigmoid activation function.

Appendix B

Sample Source Codes

B.1 Image Preprocessing Source Code

Read Image

```
image=imread('image.jpg');
figure('units','normalized','outerposition',[0 0 1 1]);
imshow(image);
```

Colour Normalization

```
figure('units','normalized','outerposition',[0 0 0.5 0.5]);
subplot(1,2,1),imshow(image_rgb);
title('RGB Image','FontSize',14);
image_gray=rgb2gray(image_rgb);
subplot(1,2,2),imshow(image_gray);
title('Gray Image','FontSize',14);
```

Noise Removal

```
image_denoised=medfilt2(image_gray);
image_denoised=image_denoised(2:end-1,2:end-1); % remove corner pixels
subplot(1,2,1),imshow(image_gray);
title('Gray Image','FontSize',14);
subplot(1,2,2),imshow(image_denoised);
title('Denoised Image','FontSize',14);
```

Image Segmentation

```
threshold=graythresh(image); % Otsu's method for finding global threshold
image_binarized=im2bw(image_denoised,threshold);
subplot(1,2,1),imshow(image_denoised);
title('Denoised Image','FontSize',14);
subplot(1,2,2),imshow(image_binarized);
title('Binarized Image','FontSize',14);
```

Image Segmentation

```
image_inverted=¬image_binarized;
subplot(1,2,1),imshow(image_binarized);
title('Binarized Image','FontSize',14);
subplot(1,2,2),imshow(image_inverted);
title('Inverted Image','FontSize',14);
```

Universe of Discourse Selection

```
image_discoursed=universe_of_discourse(image_inverted);
subplot(1,2,1),imshow(image_inverted);
title('Inverted Image','FontSize',14);
subplot(1,2,2),imshow(image_discoursed);
title('Discoursed Image','FontSize',14);
```

Image Size Normalization

```
image_normalized=imresize(image_discoursed,[36 36]);
subplot(1,2,1),imshow(image_discoursed);
title('Discoursed Image','FontSize',14);
subplot(1,2,2),imshow(image_normalized);
title('Size Normalized Image (36x36)','FontSize',14);
```

Image Skeletonization

```
image_skeletonized=bwmorph(image_normalized,'skel','inf');
subplot(1,2,1),imshow(image_normalized);
title('Size Normalized Image (36x36)','FontSize',14);
subplot(1,2,2),imshow(image_skeletonized);
title('Skeletonized Image','FontSize',14);
```

B.2 Feature Extraction Source Code

Directional Features

```
features=[]; % initializing local feature vector
% dividing given image into 3x3 zones
[row,col]=size(image);
zone_height=row/3; %12 pixels (because image size is normalized to 36x36)
zone_width=col/3; %12 pixels (because image size is normalized to 36x36)

zone11=image(1:zone_height,1:zone_width);
zone12=image(1:zone_height,(zone_width+1):2*zone_width);
zone13=image(1:zone_height,(2*zone_width+1):end);

zone21=image((zone_height+1):2*zone_height,1:zone_width);
zone22=image((zone_height+1):2*zone_height,(zone_width+1):2*zone_width);
zone23=image((zone_height+1):2*zone_height,(2*zone_width+1):end);

zone31=image((2*zone_height+1):end,1:zone_width);
zone32=image((2*zone_height+1):end,(zone_width+1):2*zone_width);
zone33=image((2*zone_height+1):end,(2*zone_width+1):end);

% directional features
zone11_features=lineclassifier(zone11);
zone12_features=lineclassifier(zone12);
zone13_features=lineclassifier(zone13);

zone21_features=lineclassifier(zone21);
zone22_features=lineclassifier(zone22);
zone23_features=lineclassifier(zone23);

zone31_features=lineclassifier(zone31);
zone32_features=lineclassifier(zone32);
zone33_features=lineclassifier(zone33);

% Combining features from all nine zones of given image.
%Each row of feature vector contains nine geometric feature elements.
features=[zone11_features;zone12_features;zone13_features;
zone21_features;zone22_features;zone23_features;zone31_features;
zone32_features;zone33_features];
% 'feature' vector is 81 dimensional row vector.
% As there are 9 zones with 9 feature elements each.

% Display zonned images with corresponding feature vectors figure;
subplot(3,3,1),imshow(zone11);title('zone11','FontSize',12);
subplot(3,3,2),imshow(zone12);title('zone12','FontSize',12);
subplot(3,3,3),imshow(zone13);title('zone13','FontSize',12);
subplot(3,3,4),imshow(zone21);title('zone21','FontSize',12);
subplot(3,3,5),imshow(zone22);title('zone22','FontSize',12);
subplot(3,3,6),imshow(zone23);title('zone23','FontSize',12);
subplot(3,3,7),imshow(zone31);title('zone31','FontSize',12);
subplot(3,3,8),imshow(zone32);title('zone32','FontSize',12);
subplot(3,3,9),imshow(zone33);title('zone33','FontSize',12);
```

Moment Invariant Features

```
% Normalized central moments are calculated from each image. These regional
% descriptors form seven elememt feature vector as there are seven moments
% invarients. Normalized Central Moments are independ of transalation,
% rotation, scalling and can be used in pattern identification.
image_size=numel(image); % Number of elemets in given image (36x36=1296)
moments=invmoments(image); % 7 dimensional column vector
```

Euler Number Feature

```
euler_number_feature=bweuler(image);
```

Area of Character Skeleton Feature

```
area=sum(sum(image));
```

Centroid Features

```
[rows,cols] = size(image);
x = ones(rows,1)*[1:cols]; %Matrix with each pixel set to its x coordinate
y = [1:rows]'*ones(1,cols); % "     "     "    "    "  "   "  y    "

area = sum(sum(image));
meanx = sum(sum(double(image).*x))/area;
meany = sum(sum(double(image).*y))/area;

centroid.x=meanx;
centroid.y=meany;

centroid_features=[centroid.x centroid.y]';
```

Eccentricity Features

```
pixellist=regionprops(image,'pixellist') ;

total_list=[];
for i=1:numel(pixellist)
total_list=[total_list;pixellist(i).PixelList];
end
list=total_list;

```

```
if (isempty(list))
MajorAxisLength = 0;
MinorAxisLength = 0;
Eccentricity = 0;
else
  % Assign X and Y variables so that we're measuring orientation
  % counterclockwise from the horizontal axis.

  xbar = centroid.x;
  ybar = centroid.y;

  x = list(:,1) - xbar;
  y = -(list(:,2) - ybar); % This is negative for the
  % orientation calculation (measured in the
  % counter-clockwise direction).

  N = length(x);

  % Calculate normalized second central moments for the region. 1/12 is
  % the normalized second central moment of a pixel with unit length.
  uxx = sum(x.^2)/N + 1/12;
  uyy = sum(y.^2)/N + 1/12;
  uxy = sum(x.*y)/N;

  % Calculate major axis length, minor axis length, and eccentricity.
  common = sqrt((uxx - uyy)^2 + 4*uxy^2);
  MajorAxisLength = 2*sqrt(2)*sqrt(uxx + uyy + common);
  MinorAxisLength = 2*sqrt(2)*sqrt(uxx + uyy - common);
  Eccentricity = 2*sqrt((MajorAxisLength/2)^2 -
      (MinorAxisLength/2)^2) / MajorAxisLength;

e=Eccentricity;
end
```

B.3 Training and Testing Source Code

MLP Training and Testing

```
% Initialize network parameters and proceeds training and testing
inputs = input_feature_vector;
targets = target_feature_vector;

% Mapping all the input vectors to the range [-1 1]
% [inputs,input_maping_setting]=mapminmax(inputs);
% [targets,target_maping_setting]=mapminmax(targets);


% Create a Feedforward Pattern Recognition Network
% Feedforward network with one input layer, one hidden layer and
% one output layer.

hiddenSizes=25; % Number of neurons in hidden layer

```

```
net = feedforwardnet(hiddenSizes);

% Setup Division of Data for Training, Validation, Testing
Q=size(input_feature_vector,2); % Number of image samples
trainRatio = 70/100;     % training samples
valRatio =   15/100;     % validation samples
testRatio = 15/100;      % testing samples

[trainInd,valInd,testInd] = dividerand(Q,trainRatio,valRatio,testRatio);

% Select training and testing samples
% training samples
train_feature_vector=[];
train_target_vector=[];
idx=1;
for i=1:size(trainInd,2)
    train_feature_vector(:,idx)=input_feature_vector(:,trainInd(i));
    train_target_vector(:,idx)=target_feature_vector(:,trainInd(i));
    idx=idx+1;
end

% testing samples
test_feature_vector=[];
test_target_vector=[];
idx=1;
for i=1:size(testInd,2)
    test_feature_vector(:,idx)=input_feature_vector(:,testInd(i));
    test_target_vector(:,idx)=target_feature_vector(:,testInd(i));
    idx=idx+1;
end


% Choose a Network Training Function
   net.trainFcn = 'trainlm';  % Levenberg-Marquardt Learning Algorithm
%  Function Parameters for 'trainlm'
%
%     Show Training Window Feedback   showWindow: true
%     Show Command Line Feedback showCommandLine: false
%     Command Line Frequency                show: 20
%     Maximum Epochs                      epochs: 5000
%     Maximum Training Time                 time: Inf
%     Performance Goal                      goal: 0.1
%     Minimum Gradient                  min_grad: 1e-005
%     Maximum Validation Checks         max_fail: 6
%     Mu                                      mu: 0.001
%     Mu Decrease Ratio                   mu_dec: 0.1
%     Mu Increase Ratio                   mu_inc: 10
%     Maximum mu                          mu_max: 10000000000

% Choose Network Weight Initialization  Function
net.layers{1}.initFcn= 'initnw'; % Nguyen Widrow weight initialization
net.layers{2}.initFcn= 'initnw';

% Choose Network Transfer Function
net.layers{1}.transferFcn='tansig'; % a = tansig(n) = 2/(1+exp(-2*n))-1
net.layers{2}.transferFcn='tansig';

% Choose a Performance Function
```

```
net.performFcn = 'mse';  % Mean squared error

% Choose a network goal (minimum mean square error)
net.trainParam.goal = 1e-003;
% Choose  time imterval to show the network status
net.trainParam.show = 20;
% Choose number of Epoches (iterations)
net.trainParam.epochs = 1000;
% Choose value of mu
net.trainParam.mu=0.001;
% memory reduction parameter
net.efficiency.memoryReduction=1;
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit','plotconfusion'};

% Train the Network
[net,tr] = train(net,train_feature_vector,train_target_vector);

% Test the Network
outputs = sim(net,test_feature_vector);
errors = gsubtract(test_target_vector,outputs);
performance = perform(net,test_target_vector,outputs);

% Overall Percentage of correct and incorrect classification
[c,cm,ind,per] = confusion(test_target_vector,outputs);
% c     Confusion value = fraction of samples misclassified
% cm    S-by-S confusion matrix, where cm(i,j) is the number
% of samples whose target is the ith class that was classified as j

fprintf('Percentage Correct Classification   : %f%%\n', 100*(1-c))
fprintf('Percentage Incorrect Classification : %f%%\n', 100*c)
fprintf('Time to Simulate the Network : %0.2f sec \n', tr.time(end))

save mlp_trained_network;
```

RBF Training and Testing

```
% Initialize network parameters and proceeds training and testing
inputs = input_feature_vector;
targets = target_feature_vector;


% Mapping all the input vectors to the range [-1 1]
% [inputs,input_maping_setting]=mapminmax(inputs);
% [targets,target_maping_setting]=mapminmax(targets);

% Setup Division of Data for Training, Validation, Testing
Q=size(input_feature_vector,2); % Number of image samples
trainRatio = 70/100;    % training samples
valRatio =   15/100;    % validation samples
testRatio = 15/100;     % testing samples

```

```
[trainInd,valInd,testInd] = dividerand(Q,trainRatio,valRatio,testRatio);

% Select training and testing samples
% training samples
train_feature_vector=[];
train_target_vector=[];
idx=1;
for i=1:size(trainInd,2)
    train_feature_vector(:,idx)=input_feature_vector(:,trainInd(i));
    train_target_vector(:,idx)=target_feature_vector(:,trainInd(i));
    idx=idx+1;
end

% testing samples
test_feature_vector=[];
test_target_vector=[];
idx=1;
for i=1:size(testInd,2)
    test_feature_vector(:,idx)=input_feature_vector(:,testInd(i));
    test_target_vector(:,idx)=target_feature_vector(:,testInd(i));
    idx=idx+1;
end

% Initialize the Radial Basis Function Parameters

goal=0.01;  % default=0.0
spread=2.0; % default =1.0
max_neurons=size(train_target_vector,2);
neuron_increment=25;

% Creating Radial Basis Network
tic;
net=newrb(train_feature_vector,train_target_vector,
          goal,spread,max_neurons,neuron_increment);
time=toc;

% -----------Testing the network---------------

outputs=sim(net,test_feature_vector);
errors = gsubtract(test_target_vector,outputs);
% Overall Percentage of correct and incorrect classification
[c,cm,ind,per] = confusion(test_target_vector,outputs);
% c     Confusion value = fraction of samples misclassified
% cm    S-by-S confusion matrix, where cm(i,j) is the number
% of samples whose target is the ith class that was classified as j

fprintf('Percentage Correct Classification   : %f%%\n', 100*(1-c));
fprintf('Percentage Incorrect Classification : %f%%\n', 100*c);
fprintf('Time to Simulate the Network : %0.2f sec \n', time);
% ------------------------------------------------------

save rbf_trained_network;
```

References

[1] K. C. Santosh and C. Nattee, "Template-based nepali handwritten alphanumeric character recognition," *Thammasat International Journal of Science and Technology (TIJSAT), Thammasat University, Thailand*, vol. 12, no. 1, 2007.

[2] S. Haykin, *Neural Networks: A Comprehensive Introduction*. Prentice Hall, 1999.

[3] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. pub-PWS:adr: PWS-Kent Publishing Company, Division of Wadsworth, Inc., 1996.

[4] C. Y. Suen, M. Berthod, and S. Mori, "Automatic recognition of handprinted characters: The state of the art," *Proceedings of IEEE*, vol. 68, pp. 469–487, Apr. 1980.

[5] V. K. Govindan and A. P. Shivaprasad, "Character recognition: A review," *Pattern Recognition*, vol. 23, no. 7, pp. 671–683, 1990.

[6] R. M. Bozinovic and S. N. Srihari, "Off-line cursive script word recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 68–83, Jan. 1989.

[7] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proceedings of IEEE*, vol. 80, pp. 1029–1058, July 1992.

[8] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, pp. 787–808, Aug. 1990.

[9] B. Shaw, S. K. Parui, and M. Shridhar, "Offline handwritten devanagari word recognition: A segmentation based approach," in *ICPR*, pp. 1–4, 2008.

[10] P. S. Deshpande, L. Malik, and S. Arora, "Fine classification & recognition of hand written devnagari characters with regular expressions & minimum edit distance method," *JCP*, vol. 3, no. 5, pp. 11–17, 2008.

[11] N. Sharma, U. Pal, F. Kimura, and S. Pal, "Recognition of off-line handwritten devnagari characters using quadratic classifier," in *Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 805–816, 2006.

[12] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, and M. Kundu, "Multiple classifier combination for off-line handwritten devnagari character recognition," June 30 2010.

[13] S. Arora, D. Bhattacharjee, M. Nasipuri, L. Malik, M. Kundu, and D. K. Basu, "Performance comparison of SVM and ANN for handwritten devnagari character recognition," June 30 2010.

[14] Y. Gao, L. Jin, C. He, and G. Zhou, "Handwriting character recognition as a service: A new handwriting recognition system based on cloud computing," in *ICDAR*, pp. 885–889, IEEE, 2011.

[15] C. Kan and M. D. Srinath, "Invariant character recognition with zernike and orthogonal fourier-mellin moments," *Pattern Recognition*, vol. 35, pp. 143–154, Jan. 2002.

[16] H. Bunke, "Recognition of cursive roman handwriting - past, present and future," *In 7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, vol. 01, pp. 7448–459, 2003.

[17] I. A. Lawal, R. E. Abdel-Aal, and S. A. Mahmoud, "Recognition of handwritten arabic (indian) numerals using freeman's chain codes and abductive network classifiers," in *ICPR*, pp. 1884–1887, IEEE, 2010.

[18] J. Ashok and E. G. Rajan, "Off-line handwritteen character recognition suing radial basis function," *int. J. Advanced Networking and Applications*, vol. 02, pp. 792–795, 2011.

[19] V. J. Dongre and V. H. Mankar, "A review of research on devnagari character recognition," *CoRR*, vol. abs/1101.2491, 2011.

[20] N. VenkateswaraRao, A. Shrikhna, B. RaveendraBabu, and G. R. M. Babu, "An efficient feature extraction and classification of handwritten digits using neural networks," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 1, Oct. 2011.

[21] D. D. Gaurav and R. Ramesh, "A feature extraction technique based on character geometry for character recognition," *CoRR*, vol. abs/1202.3884, 2012.

[22] A. Vinciarelli, "A survey on off-line cursive word recognition," *Pattern Recognition*, vol. 35, pp. 1433–1446, July 2002.

[23] O. D. Trier, A. K. Jain, and T. Taxt, "Feature-extraction methods for character-recognition: A survey," *Pattern Recognition*, vol. 29, pp. 641–662, Apr. 1996.

[24] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, Jan. 2000.

[25] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[26] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Trans. Systems, Man and Cybernetics*, vol. 9, pp. 62–66, Mar. 1979. minimize intra and inter class variance.

[27] H. Blum, "A Transformation for Extracting new Descriptors of Shape," in *Models for the Perception of Speech and Visual Form* (Wathen-Dunn, ed.), pp. 362–380, MIT-Press, 1967.

[28] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Third Edition*. Prentice Hall, 2008.

[29] M. Blumenstein, B. Verma, and H. Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters," in *ICDAR*, pp. 137–141, 2003.

[30] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. IT-8, pp. 179–187, 1962.

[31] W. K. Pratt, *Digital Image Processing (2nd Edition)*. New York, New York: John Wiley & Sons, Inc., 1991.

[32] P. L. Rosin, "Measuring rectangularity," *Mach. Vis. Appl*, vol. 11, no. 4, pp. 191–196, 1999.

[33] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989–993, Nov. 1994.

[34] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 2, pp. 302–309, 1991.

[35] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *IEEE International Joint Conference on Neural Networks (4th IJCNN'90)*, vol. III, (San Diego), pp. III–21–III–26, IEEE, 1990. Stanford.