B.Sc. in Computer Science and Engineering Thesis

Bangla Character Recognition using Artificial Neural Network Step:Classifier Design

Submitted by

Kazi Lutful Kabir
201014025

Md. Rayhan Kabir
201014029

Md. Aminul Islam
201014041

Supervised by

Dr. Hasan Sarwar
Professor and Head of the Department, CSE
United International University

Department of Computer Science and Engineering
Military Institute of Science and Technology

CERTIFICATION

This thesis paper titled **“Bangla Character Recognition using Artificial Neural Network Step:Classifier Design”**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering on December 2013.

Group Members:

Kazi Lutful Kabir
Md. Rayhan Kabir
Md. Aminul Islam

Supervisor:

————————————-

Dr. Hasan Sarwar
Professor and Head of the Department, CSE
United International University
House no. 80, Road 8/A
Sat Masjid Road, Dhanmondi
Dhaka, Bangladesh.

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis paper is the outcome of the investigation and research carried out by the following students under the supervision of Dr. Hasan Sarwar, Professor and Head of the Department, CSE, United International University, House no. 80, Road 8/A, Sat Masjid Road, Dhanmondi, Dhaka, Bangladesh.

It is also declared that neither this thesis paper nor any part there of has been submitted anywhere else for the award of any degree, diploma or other qualifications.

————————————-

Kazi Lutful Kabir
201014025

————————————-

Md. Rayhan Kabir
201014029

————————————-

Md. Aminul Islam
201014041

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor Dr. Hasan Sarwar, Professor and Head of the Department, CSE, United International University, House no. 80, Road 8/A, Sat Masjid Road, Dhanmondi, Dhaka, Bangladesh, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Dhaka	Kazi Lutful Kabir
December 2013	Md. Rayhan Kabir
	Md. Aminul Islam

ABSTRACT

Character recognition is a very popular research field since 1950's. A great deal of research work has been done for various languages specifically in case of English. The recognition of optical characters is known to be one of the earliest applications of Artificial Neural Networks. The use of artificial neural network simplifies development of an optical character recognition application, while achieving highest quality of recognition and good performance. Although Bangla is one of the most widely spoken languages (over 200 million people use Bangla as their medium of Communication) of the world, research is acute in recognition of Bangla characters. Under this context, an effort has been taken globally to computerize the Bangla language. Compared to English and other language scripts, one of the major stumbling blocks in Optical Character Recognition (OCR) of Bangla script is the large number of complex shaped character classes of Bangla alphabet. In addition to 50 basic character classes, there are nearly 160 complex shaped compound character classes in Bangla alphabet. Dealing with such a large variety of characters with a suitably designed feature set is a challenging problem. Uncertainty and imprecision is inherent in handwritten script. Moreover, such a large variety of complex shaped characters, some of which have close resemblance, makes the problem of OCR of Bangla characters more difficult. Considering the complexity of the problem, this research makes an attempt to develop a method for the recognition of Bangla characters using the artificial neural network. Pre-processing steps involves segmentation and binarization. Features are taken using different feature extraction procedures. Multilayered neural network is used in the spirit of back-propagation algorithm for classification as well as recognition of characters. To deal with immense variation and magnificent diversity of Bangla characters, this effort have widened the area for many research works to come to light and to bid fair to be accomplished.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

OCR	: Optical Character Recognition
ANN	: Artificial Neural Network
EDI	: Electronic Data Interchange
AIIM	: Association for Information and Image Management
PDF	: Probability Density Function
MLP	: Multi Layer Perceptron
SVM	: Support Vector Machine
RNN	: Recurrent Neural Network

CHAPTER 1

INTRODUCTION

1.1 Overview of Character Recognition

Humans recognize characters easily and they repeat the character recognition process thousands of times every day as they read papers or books. However, after many years of intensive investigation and research, the ultimate goal of developing an optical character recognition (OCR) system with the same reading capabilities as humans still remains unachieved. Optical Character Recognition deals with the problem of recognizing optically processed characters. Optical recognition is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are drawn. Both hand printed and printed characters may be recognized, but the performance is directly dependent upon the quality of the input documents. The more constrained the input is, the better will the performance of the OCR system be. However, when it comes to totally unconstrained handwriting, OCR machines are still a long way from reading as well as humans. In fact, the computer reads fast and technical advances are continually bringing the technology closer to its ideal. Basically, OCRs are divided into two major categories: typewritten and handwritten. Typewritten OCR systems recognize a document that has been previously typed and scanned prior to recognition. On the other side, handwritten OCR systems recognize a text that has been written by a human. Comparing to handwritten OCR systems, Typewritten OCR systems are usually easier to design and the recognition rate achieved for typewritten recognition systems is more than the handwritten. OCRs are further categorized to offline and online recognition systems. In offline OCR systems, the image of the typewritten or the handwritten text is acquired through scanning. The image then is read by the OCR system and is analyzed for recognition. In online OCR systems, input of the OCR system is an image of a handwritten text which is usually acquired using cell phone or a portable personal computer.

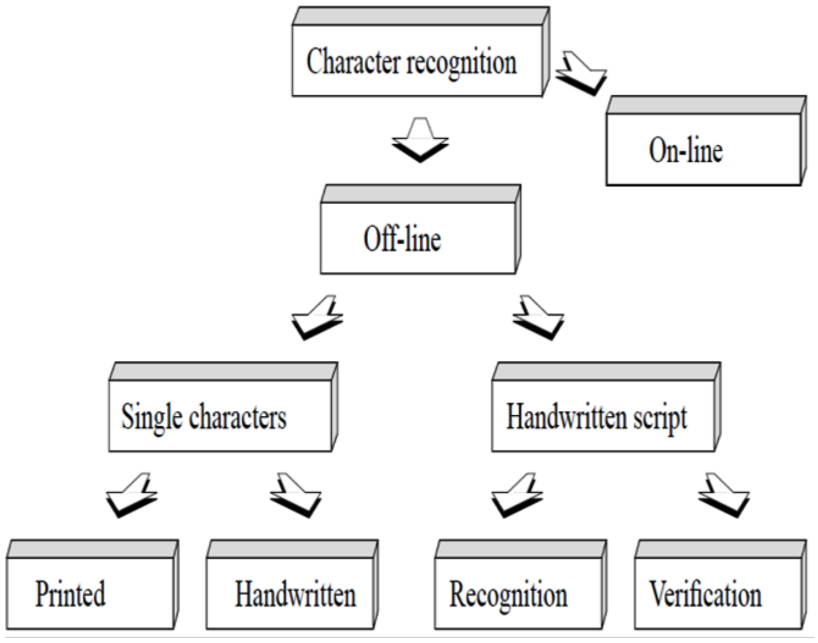


Figure 1.1: Areas of Character Recognition

1.2 Brief History of Character Recognition

In the middle of the 1940s, the first character recognizers appeared and mainly focused on machine-printed text, and some of them dealt with handwritten text or symbols. In 1950s, commercial character recognizers were available for Latin languages. In 1980s, many structural and statistical methods were used in character recognition; some of those recognizers broke the character image into a set of lines and curves and basically focused on the shape recognition techniques without using any semantic information. After 1990, complex character recognition algorithms were developed; many recognizers used sophisticated methodologies such as neural networks, hidden Markov models and natural language processing techniques. Many applications such as reading postal address off envelopes, reading customer filled forms, archiving and retrieving text and digitizing libraries benefit from OCR systems.

1.3 What is OCR?

OCR (optical character recognition) is the recognition of printed or written text characters by a computer. This involves photo-scanning of the text character-by-character, analysis of the scanned-in image, and then translation of the character image into character codes, such as ASCII, commonly used in data processing. In OCR processing, the scanned-in image or bitmap is analyzed for light and dark areas in order to identify each alphabetic letter or numeric digit. When a character is recognized, it is converted into an ASCII code. Special circuit boards and computer chips designed expressively for OCR are used to speed up the recognition process.OCR is being used by libraries to digitize and preserve their holdings. OCR is also used to process checks and credit card slips and sort the mail. Billions of magazines and letters are sorted every day by OCR machines, considerably speeding up mail delivery.

1.4 Current Status of OCR Technology

Optical Character Recognition has come a long way from its beginnings in 1929, when Gustav Tauschek patented a mechanical device that used templates and a photo-detector to read words. Now used on an everyday basis by individuals and companies across the world, the technology is considered one of the greatest technological advancements in word processing. Recent studies have sought to examine the current level of accuracy of OCR software. A recent study found that accuracy still ranges from 71% - 98% depending on the quality of the print, the font of the characters, and the quality of the OCR software. Accuracy rates are examined based on different measures. The way it is measured can alter the accuracy determination. For example, if word context is not used to identify non-existent words and correct them, it could keep what would be an error rate of 1% to a higher error rate of 5%. Cursive text is the most challenging type of written text for OCR to read, and continues to be the highest in error rates. Research is ongoing to develop better ways to read cursive text. One theory is that it would be easier for OCR to recognize a dictionary of words than to try to decipher individual cursive letters. But the development of such software would be very tedious and time consuming. OCR technology is considered a basic technology, but is used in advanced scanning applications. The advanced scanning solution used in any particular software can be unique and patented, even though each is based on a similar OCR technology. As these competitive applications continue to improve, OCR accuracy and capabilities will continue to increase in accuracy and usability.

1.5 Future Scopes of OCR Usage

Today's uses of OCR are still somewhat limited to the scanning of the written word into useable computer text. The uses include word processing, mail delivery system scanning, ticket reading, and other such tasks. In the future, the uses of OCR have been speculated to be far more advanced. Some of these advancements are already in the workings. Most believe that the use for reading the written text will diminish as electronic data interchange (EDI) is used more and more, while paper documents are phased out. Many actually debate whether or not paper will even exist in the next several decades. However, a recent AIIM study found that imaging increased paper consumption because of increased printing. While the debate carries on, it is clear that advancements will lead OCR to greater heights than before. OCR is already used to detect viruses, or unfortunately create them, and to stop spammers. Anti-spamming applications continue to be improved, as the need for increased technology is a constant. OCR is used to prevent viruses by detecting codes hidden in images. While the difficulty of transferring information from legacy systems to modern operating systems can be cumbersome, OCR is able to read screenshots. This can facilitate the transferring of information between incompatible technologies. One of the current hopes for OCR is the chance of developing OCR software that can read compressed files. Text that is compressed into an image and saved as ASCII or Hexadecimal data could be read by OCR and transferred back into readable text. Finally, it is anticipated that OCR will be used in the development of more advanced robotics. The eyes of a robot are essentially a camera meant to input information. If OCR is used to help the robot comprehend text, the uses could be almost endless. All of these advancements and more are expected to keep the technology of OCR in use and continue to expand its current capabilities.

CHAPTER 2

CHARACTER RECOGNITION PROCESS

2.1 Overview

Character recognition has been one of the most fascinating and challenging research areas in the fields of image processing and pattern recognition in the recent years. The goal of Optical Character Recognition (OCR) is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters. OCR is a complex technology that converts images with text into editable formats. OCR allows processing of scanned books, screenshots and photos with text to obtain editable documents like txt, doc or pdf files. This technology is widely used in many areas and the most advanced OCR systems can handle almost all types of images, even such complex as scanned magazine pages with images and columns or photos from a mobile phone. The process of OCR involves several steps including segmentation, feature extraction, and classification each of which is a field unto itself.

2.2 Steps of Character Recognition Process

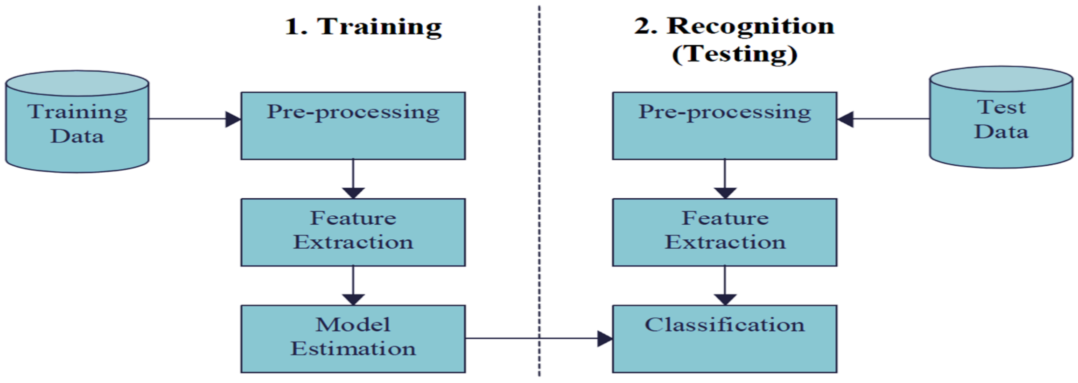


Figure 2.1: Block Diagram of Character Recognition Process

Several steps are involved in character recognition process. Figure 2.1 provides a bird's eye view of the whole process involved [1].

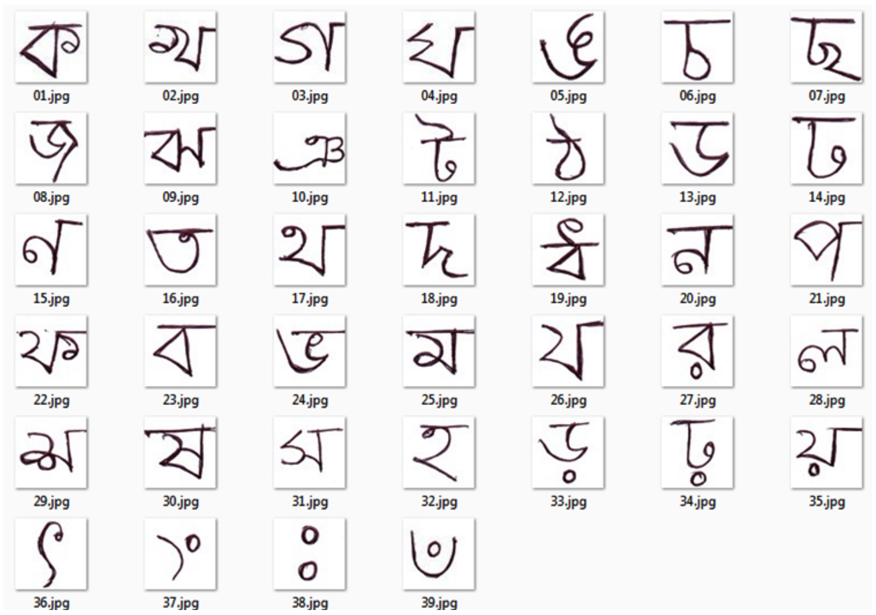


Figure 2.2: Hand-Written Bangla Consonants

অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ

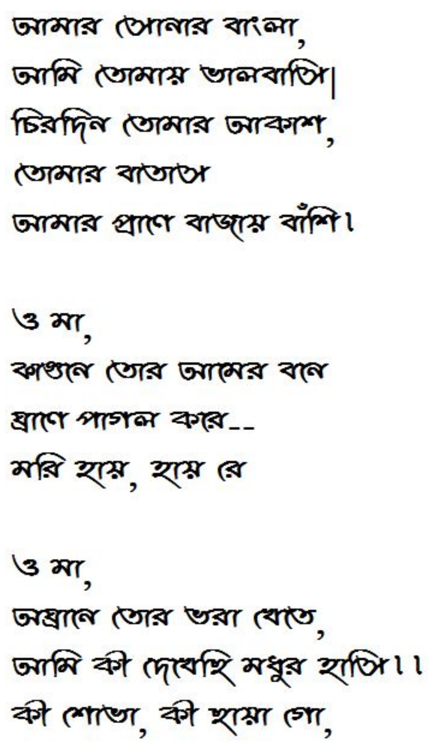
ক খ গ ঘ ঙ চ ছ জ ঝ ঞ

ট ঠ ড ঢ ণ ত থ দ ধ ন

প ফ ব ভ ম য র ল শ ষ

স হ ক্ষ

Figure 2.3: Printed Bangla Alphabets

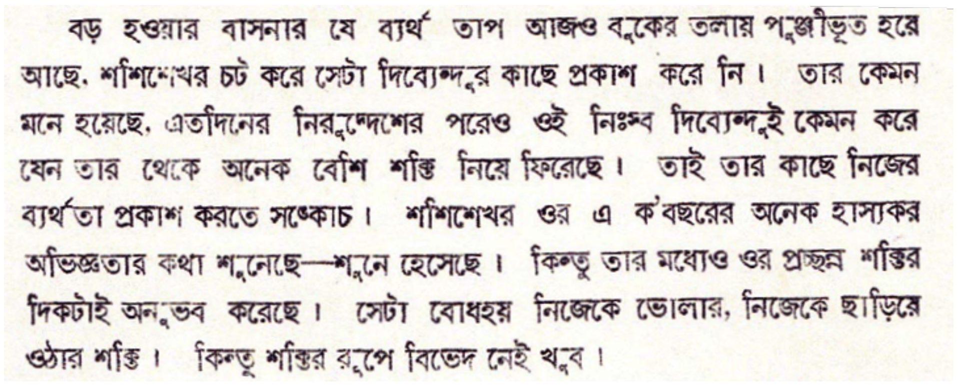


আমার সোনার বাংলা,
আমি তোমায় ভালবাসি।
চিরদিন তোমার আকাশ,
তোমার বাতাস
আমার প্রাণে বাজায় বাঁশি।

ও মা,
ফাগুনে তোর আমের বনে
ঘ্রাণে পাগল করে--
মরি হায়, হায় রে

ও মা,
অঘ্রানে তোর ভরা খেতে,
আমি কী দেখেছি মধুর হাসি।।
কী শোভা, কী ছায়া গো,

Figure 2.4: Document with Stylish Bangla Fonts



বড় হওয়ার বাসনার যে ব্যর্থ তাপ আজও বুকের তলায় পুঞ্জীভূত হয়ে আছে, শশিশেখর চট করে সেটা দিব্যেন্দুর কাছে প্রকাশ করে নি। তার কেমন মনে হয়েছে, এতদিনের নিরুদ্দেশের পরেও ওই নিঃস্ব দিব্যেন্দুই কেমন করে যেন তার থেকে অনেক বেশি শক্তি নিয়ে ফিরেছে। তাই তার কাছে নিজের ব্যর্থতা প্রকাশ করতে সঙ্কোচ। শশিশেখর ওর এ ক'বছরের অনেক হাস্যকর অভিজ্ঞতার কথা শুনেছে—শুনে হেসেছে। কিন্তু তার মধ্যেও ওর প্রচ্ছন্ন শক্তির দিকটাই অনুভব করেছে। সেটা বোধহয় নিজেকে ভোলার, নিজেকে ছাড়িয়ে ওঠার শক্তি। কিন্তু শক্তির রূপে বিভেদ নেই খুব।

Figure 2.5: Old Bangla Novel

2.3 Pre-processing

Data preprocessing describes any type of processing performed on raw data to prepare it for another procedure. Hence, preprocessing is the preliminary step which transforms the data into a format that will be more easily and effectively processed. Therefore, the main task in preprocessing the captured data is to decrease the variation that causes a reduction in the recognition rate and increases the complexities, as for example, preprocessing of the input raw stroke of characters is crucial for the success of efficient character recognition systems. Thus, preprocessing is an essential stage prior to feature extraction since it controls the suitability of the results for the successive stages. The stages in a character recognition system are in a pipeline fashion meaning that each stage depends on the success of the previous stage in order to produce optimal/valid results. Several pre-processing steps are performed in OCR out of which Binarization and Segmentation are most important ones.

2.3.1 Binarization

A binary image is a digital image that has only two possible values for each pixel. Typically the two colors used for a binary image are black and white though any two colors can be used. The color used for the objects in the image is the foreground color while the rest of the image is the background color. In the document scanning industry this is often referred to as bi-tonal. A binary image can be stored in memory as a bitmap, a packed array of bits. A 640x480 image requires 37.5 KiB of storage. Binary images can be interpreted as subsets of the two-dimensional integer lattice. Because of the small size of the image files, fax machine and document management solutions usually use this format. Most binary images also compress well with simple run-length compression schemes. In an OCR, one of the main pre-processing steps is binarization of document images, i.e. separation of foreground from background. Binarization of a text image provides in an ideal case, the foreground text in black and noisy background in white. Different thresholding methods don't give perfect results for all types of documents. Some algorithms might work better for one type of documents where there are marks of strain while they might give poor results for other types having extremely low intensity variations. Different binarization methods are available for different types of documents such as : Niblack's method, Sauvola's algorithm, Wolf's work, Feng's method, Bernsen Algo and so on. Some of them are discussed in brief,

1. Bernsen Algorithm : Bernsen algorithm is proposed for uneven illumination, particularly for shadow removal. Let f(x,y) denotes a gray value of point (x,y). Consider a block whose center is a point (x, y) and size is (2w +1)×(2w +1). The threshold T(x, y) of f(x,y) is computed by,

$$T_1\text{(x,y)} = \frac{max f(x+l,y+k)+min f(x+l,y+k)}{2}\text{; -w} \leq \text{k, l} \leq \text{w}$$

Bernsen algorithm is sensitive to noise, which disturbs the characters extraction. If f(x, y) denotes the gray value obtained with the Gaussian flter, σ is the scale of the Gaussian flter, and k and l are the parameters of the window. An improved Bernsen algorithm can be depicted as,

1. Compute the threshold T1(x, y) of f(x, y)

   $T_1\text{(x,y)} = \frac{max f(x+l,y+k)+min f(x+l,y+k)}{2}\text{; -w} \leq \text{k, l} \leq \text{w}$

2. Create the Gaussian filter for the window, s =(2w + 1)×(2w + 1) of f(x, y).

3. Compute the threshold T_2(x,y) of f′(x, y) as,

   $T_2\text{(x,y)} = \frac{max f'(x+l,y+k)+min f'(x+l,y+k)}{2}\text{; -w} \leq \text{k, l} \leq \text{w.}$

4. To remove noise applying median Filter [2].

2. Niblack Algorithm : Niblack's algorithm calculates a pixel-wise threshold by sliding a rectangular window over the gray level image. The computation of threshold is based on the local mean, m and the standard deviation, s of all the pixels in the window and is given by the equation below,

$T_{Niblack}$ = m + k * s
= m + k * $\sqrt{\sum(Pi-m)^2/NP}$
= m + k * $\sqrt{\sum Pi^2/NP-m^2}$
= m + k $\sqrt{B}$

Where, NP is the number of pixels in the gray image, m is the average value of the pixels P_i, and k is fixed to a certain value(usually, -0.2). Advantage of Niblack is that it always identifies the text regions correctly as foreground but on the other hand tends to produce a large amount of binarization noise in non-text regions also [3].

3. Sauvola's Algorithm : Sauvola's algorithm claims to improve Niblack's method by computing the threshold using the dynamic range of image gray-value standard deviation, R,

$$T_{Sauvola} = m * (1 - k * (1 - \frac{s}{R}))$$

Where, k is set to 0.5 and R to 128. This method outperforms Niblack's algorithm in images where the text pixels have near 0 gray-value and the background pixels have near 255 gray-values. However, in images where the gray values of text and non-text pixels are close to each other, the results degrade significantly [3].

4. Wolf's Algorithm : To address the issues in Sauvola's algorithm, Wolf proposed to normalize the contrast and the mean gray value of the image and compute the threshold as,

$$T_{Wolf} = (1-k) * m + k * M + k * (\frac{s}{R} * (m - M))$$

where, k is fixed to 0.5, M is the minimum gray value of the image and R is set to the maximum gray-value standard deviation obtained over all the local neighborhoods (windows) [3].

5. Feng's Algorithm : Instead of calculating dynamic range of gray-value standard deviation form the whole image like, Feng proposed calculating it locally introducing the notion of two local windows, one contained within the other. The values of local mean m, the minimum gray-level M, and standard deviation s are calculated in the primary local window while the dynamic range standard deviation R_s is calculated in the larger window termed as 'secondary local window'. Binarization threshold is then computed as:

$$T_{Feng} = (1- \alpha_1) * m + \alpha_2 * \frac{s}{R_s} * (m-M) + \alpha_3 * M$$

Where, α_2 and α_3 can be expressed in terms of k_1, k_2, s, R_s and γ. Based on the experimental experiences, γ is set to 2 while the values of other parameters, $\alpha 1$, k_1 and k_2 are proposed to be in the ranges from 0.1 to 0.2, 0.15 to 0.25 and 0.01 to 0.05 respectively. This method addresses well the R-problem in the Wolf's algorithm. However the introduction of three parameters, determined empirically, leaves the robustness of this method questionable [3].

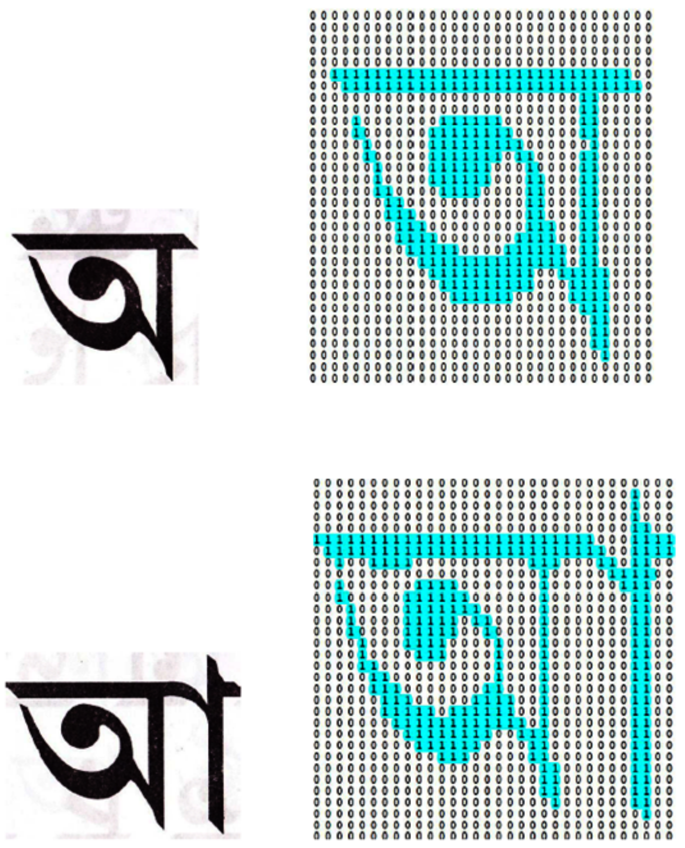


Figure 2.6: Character Images and Corresponding Binary Grid (32×32)

2.3.2 Segmentation

Segmentation checks connectivity of shapes, labels, and then isolate but difficulties with characters that aren't connected, e.g. a semicolon, or a colon. Segmentation is by far the most important aspect of the pre-processing stage. It allows the recognizer to extract features from each individual character. In the more complicated case of handwritten text, the segmentation problem becomes much more difficult as letters tend to be connected to each other. Segmentation process can be approached in various ways. The goal of segmentation is very clear. Best segmentation should be described the way which gives the shape of each character undistorted and in an isolated fashion. The task of separating each vowel or consonant would not have been so difficult if there were no vowel or consonant modifiers, no compound or touching characters. Modifiers take special shapes with the attaching character and many of them share the same vertical region with the main character.

2.4 Feature Extraction

Choice of suitable features for pattern classes is a domain specific design Issue. Feature Extraction of Bangla characters is an essential part during optical character recognition. Several feature extraction methods have been mentioned in literature along with many classification algorithms. The right combination of feature generation algorithm and classification method may greatly enhance the performance of an OCR. There are different types of features. Some of the features have been used for hand written scripts, while some others have been used for printed characters. A compilation of some of the features is given.

2.4.1 Shadow Features

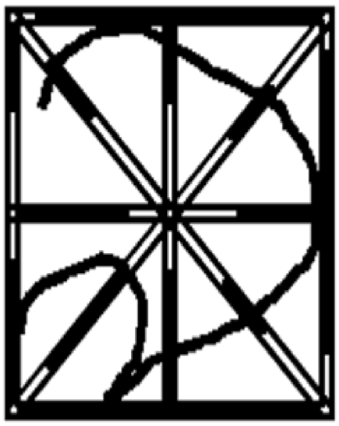
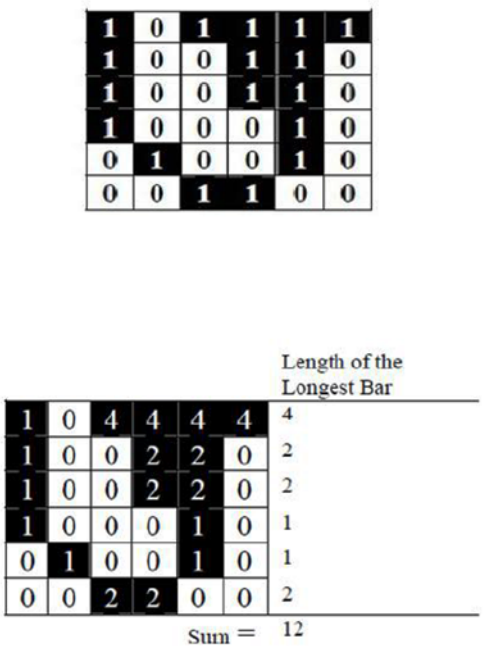


Figure 2.7: An Illustration of the 24 Shadow Features

Shadow features are computed by considering the lengths of projections of the character images, as shown in Figure 2.7, on the four sides and eight octants dividing sides of the minimal bounding boxes enclosing the same. Considering the lengths of projections on three sides of each such octant, 24 shadow features are extracted from each digit image, which is divided into eight octants inside the minimal box. Each value of the shadow feature so computed is to be normalized by dividing it with the maximum possible length of the projections on the respective side [4].

2.4.2 Longest Run Features

For computing longest-run features from a character image, the minimal bounding box enclosing the image is divided into several overlapping rectangular regions. Coordinates (r, c) of top left corners of all these regions, in terms of the row number r and the column number c, are given as follows: { (r, c) : r=0, $\frac{h}{4}$,$\frac{2h}{4}$ and c=0, $\frac{w}{4}$, $\frac{2w}{4}$ }, where h and w denote the height and the width of the minimal bounding box respectively. In each such rectangular region, 4 longest-run features are computed row wise, column wise and along two of its major diagonals.



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

						Length of the Longest Bar
1	0	4	4	4	4	4
1	0	0	2	2	0	2
1	0	0	2	2	0	2
1	0	0	0	1	0	1
0	1	0	0	1	0	1
0	0	2	2	0	0	2
					Sum =	12

Figure 2.8: An Illustration to the Computation of the row wise Longest-run Feature

The row wise longest-run feature is computed by considering the sum of the lengths of the longest bars that fit consecutive black pixels along each of all the rows of a rectangular region, as illustrated in Figure 2.8. In fitting a bar with a number of consecutive black pixels within a rectangular region, the bar may be extended beyond the boundary of the region if the same is continued there. The three other longest-run features are computed in the same way but along the column wise and two major diagonal wise directions within the rectangle separately. Thus, in all, 9×4=36 longest run features are computed from each character image. Each of these feature values is to be normalized by dividing the same with h×w.

2.4.3 Quad Tree-based Features

A quad-tree is a tree data structure in which each node except the leaf nodes has up to four children. Quad-trees are most often used for representation of a two dimensional space by recursively subdividing it into four equal quadrants or regions. A modified version of quad tree structure can be used to partition any pattern into multiple sub-images. Where, partitioning a character pattern (or a subpart of it) into 4 regions is done by drawing a horizontal and a vertical line through the Centre of Gravity (CG) of black pixels in that region. If the depth of the quad-tree structure is d, then total number of sub images for each digit pattern at leaf nodes would be 4d. The coordinates of the CG of any image frame, (Cx ,Cy) is calculated as follows:

$$C_x = \frac{1}{mn}\sum x.f(x,y) \text{ and } C_y = \frac{1}{mn}\sum y.f(x,y)$$

Where, f(x,y)= 1 (for black pixels) and 0 (for white pixels) [5].

2.4.4 Chain Code Features

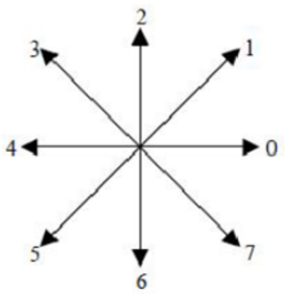


Figure 2.9: Slope Convention for Freeman Chain Code

Chain code is used to extract features for connected components of characters. Each connected component has been divided into four regions indicating four quadrants in 2D geometric system. There are several chain code conventions used for image representation, but the most popular one is Freeman chain code. Freeman Chain Code is based on the observation that each pixel has eight neighborhood pixels. A connected component is divided into 4 regions by a horizontal and a vertical line that go through the center of mass. In each zone, contour of the connected component is traversed. The frequency of each directional slope is counted. There are 8 directional slopes in a region, shown in Figure 2.9. As a result, in total, 32 directional slopes are found. These 32 values are the feature vectors, which are normalized by dividing them with square root of the sum of squares of all feature values [6].

2.4.5 Watershed Features

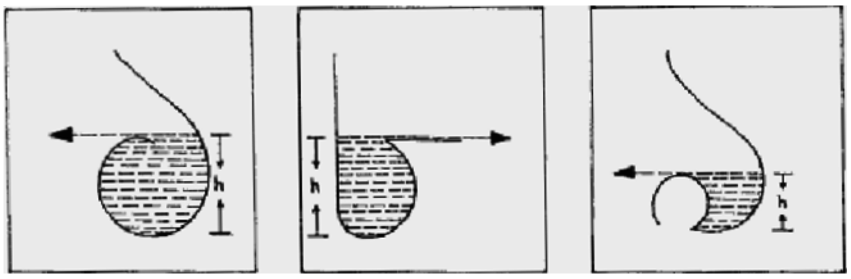


Figure 2.10: Features using Waterflow Model

Waterflow model was used from the concept of water overflow to find features of handwritten numerals. The principle is, if water is poured from the above of character, the position where water is stored as reservoir, the shape of the reservoir as hole. Feature vectors found from this model are,

1. Existence of holes and its number
2. Position of holes with respect to its bounding box
3. Ratio of hole-length to height of the characters
4. Center of gravity of the holes
5. Number of crossings in a particular region of the numeral
6. Convexity of holes, etc [7].

2.4.6 Stroke Features (Curvature based)

A stroke is a set of dark pixels such that for all except two of its members there are two dark neighbors from among the members of the set itself. A stroke consists of one or more segments. A segment is also a set of dark pixels. Except two pixels, all the other pixels have two dark neighbors from the same set. Of those two pixels, at least one has a junction. There will be no other junction pixel. A junction is a dark pixel, which has at least three dark 8-

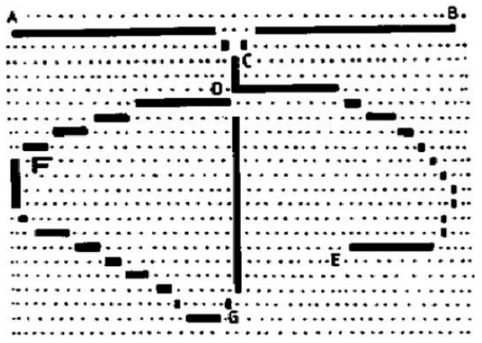


Figure 2.11: Segments: DE, AC, BC; Strokes: ACB, DE; Loop: DEG; Junctions: C,D

neighbors. A character may be represented in terms of the structural constraints imposed by junction points and the primitives/segments meeting at junctions. A stroke generates eight feature vectors having values,

1. Number of points of curvature maxima: if the curvature of a point along a stroke is greater (in magnitude) than that of its two immediate neighbors on both sides, the count of curvature maxima is increased by one.

2. Number of points of curvature minima: if the curvature of a point along a stroke is less (in magnitude) than that of its two immediate neighbors on both sides, the count of curvature minima is increased by one.

3. Number of points of inflexion from negative to positive curvature: for each pixel of a stroke, a count is incremented if its two predecessor points' curvature is non-negative and non-zero and two successor points' curvature is non-positive and non-zero.

4. Number of points of inflexion from positive to negative curvature: for each pixel of a stroke, a count is incremented if its two predecessor points' curvature is non-positive and non-zero and two successor points' curvature is non-negative and non-zero.

5. Normalized positions with respect to the stroke-length for the points considered in 1 (Number of components=4).

6. Normalized positions with respect to the stroke-length for the points considered in 2 (Number of components=4).

7. Normalized positions with respect to the stroke-length for the points considered in 3 (Number of components=4).

8. Normalized positions with respect to the stroke-length for the points considered in 4 (Number of components=4).

2.4.7 Stroke Features (Mostly Linear)

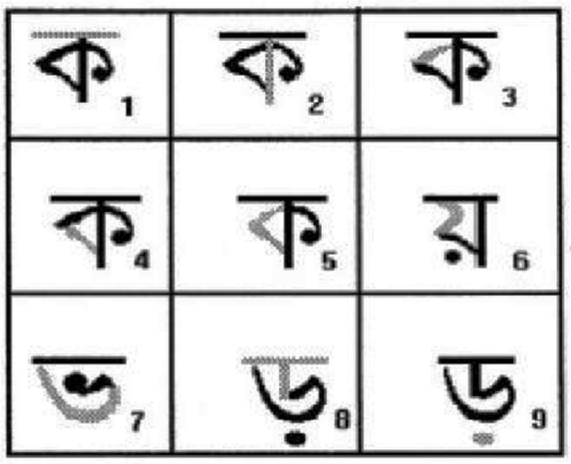


Figure 2.12: Stroke Features(Mostly Linear)

It denotes the representation of another way of identifying strokes from a character. These stroke features are mostly linear in structure. A total of 9 stroke features have been used. The information of existence or non-existence of these strokes are used in classification. These strokes are described below,

1. A horizontal continuous line over the character, known as matra line, and assumed to occupy 75% of character width.

2. A vertical continuous line assumed to occupy approximately 75% of the character middle zone.

3. A diagonal line along +45^o with horizontal, occupies 40% of the height of middle zone.

4. A diagonal line in the lower half part of middle zone along 45^o direction.

5. Existence of both stroke 3 and stroke 4.

6. Length of the arms is assumed to be 30% of the width of the middle zone of the character, and the angle between them is 315^o.

7. Stroke 7 is a cup-shaped feature where the bottom of the cup touches the base line.

8. A combination of stroke 1 and stroke 2, whose length is 40% of the height of the middle zone.

9. It is in the lower part of the middle zone.

The stroke lengths are standardized with respect to the character middle zone height because this height is constant for characters of single font and size [8].

2.5 Model Estimation

Given labeled sets of features for many characters, where the labels correspond to the particular classes that the characters belong to, next step is to estimate a statistical model for each character class. For example, if two features are computed for each realization of the character samples 0 through 9. Plotting each character class as a function of the two features provides- Each character class tends to cluster together. This makes sense; a given character should look about the same for each realization (provided that the size font type and size are used). It might be tried to estimate a probability density function (or pdf parameters such as mean and variance) for each character class.

2.6 Classification

According to Tou and Gonzalez, *The principal function of a pattern recognition system is to yield decisions concerning the class membership of the patterns with which it is confronted.* In the context of an OCR system, the recognizer is confronted with a sequence feature patterns from which it must determine the character classes. Two widely used classifiers are: MLP classifier and SVM classifier.

2.6.1 MLP Classifier

The MLP is a special kind of Artificial Neural Network (ANN). MLP has been chosen because of its well-known learning and generalization abilities, which is necessary for dealing with imprecision in input patterns. Architecturally, an MLP is a feed-forward layered network of artificial neurons. Each artificial neuron in the MLP computes a sigmoid function of the weighted sum of all its inputs. An MLP consists of one input layer, one output layer and a number of hidden or intermediate layers. The output from every neuron in a layer of the MLP is connected to all inputs of each neuron in the immediate next layer of the same. Neurons in the input layer of the MLP are all basically dummy neurons as they are used simply to pass on the input to the next layer just by computing an identifier function each. The numbers of neurons in the input and the output layers of an MLP are chosen depending on the problem to be solved. The number of neurons in other layers and the number of layers in the MLP are all determined by a trial and error method at the time of its training. An ANN requires training to learn an unknown input-output relationship to solve a problem.

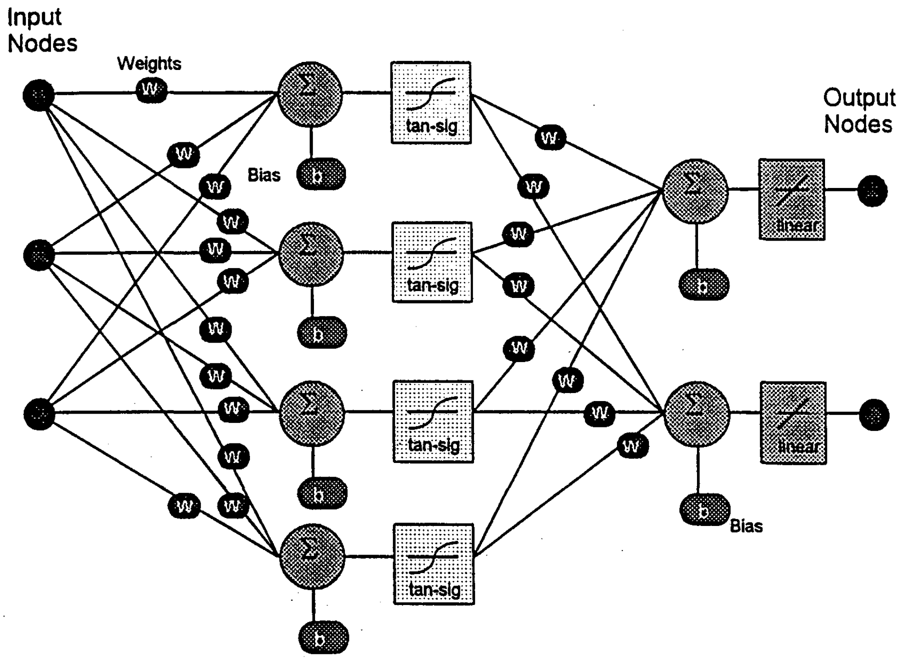


Figure 2.13: An MLP shown as a Feed Forward Neural Network.

Depending on the models of ANNs, training is performed either under supervision of some teacher (i.e., with labeled data of known input-output responses) or without supervision. The MLP to be used for the present work requires supervised training. During training of an MLP weights or strengths of neuron to neuron connections, also called synapses, are iteratively tuned so that it can respond appropriately to all training data and also to other data, not considered at the time of training. Learning and generalization ability of an ANN is determined on the basis of how best it can respond under these two respective situations. The MLP classifier designed for the present work is trained with the Back Propagation (BP) algorithm. It minimizes the sum of the squared errors for the training samples by conducting a gradient descent search in the weight space. The number of neurons in a hidden layer in the same are also adjusted during its training. The problem of pattern classification involves two successive transformations,

$$\mathrm{M} \rightarrow \mathrm{F} \rightarrow \mathrm{D}$$

where, M, F and D stand for the measurement space, the feature space and the decision space respectively. Once a feature set is fixed up, it is left with the design of a mapping (δ) as follows,

$$\delta : \mathrm{F} \rightarrow \mathrm{D}$$

ANNs with their learning and generalization abilities can approximate a general class of functions,

$$\mathrm{f} : R^n \rightarrow \mathrm{R}$$

Pattern classification with Artificial Neural Networks requires approximating as a discrete valued function,

$$R^n \rightarrow \{1,2,3,..,\mathrm{m}\}$$

where, n and m denotes the number of features and the number of pattern classes respectively. So, an ANN based pattern classifier requires n number of neurons in the input layer and m number of neurons in the output layer. Conventionally 1 out-of-m representations is used for its output [5].

2.6.2 SVM Classifier

Recently Support Vector Machine has been used successfully for pattern recognition and regression tasks formulized under the concept of structural risk minimization rule. It was mainly designed for binary classification, in order to construct an optimal hyper-plane, to maximize the margin of separation between the negative and positive data set. Although, SVM is used for two class pattern classification problem but multi-class problem can also be solved by extending the binary classification to multi-class classification. For the Support Vector Machine classifier, an open source software LibSVM tool is used. In general, a classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target value (class labels) and several attributes (features). The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes. Before considering the data directly from the linearly scaling of each attribute to the range [-1, +1] or [0, 1]. Given a training set of instance-label pairs (x_i , y_i); i =1,........,l where, $x_i \in R^n$ and y $\in$ { 1, -1 }, the support vector machines (SVM) require the solution of the optimization problem,

$$\min \text{w,b,}\xi \left(\tfrac{1}{2} w^T.\text{w} + \text{C} \sum \xi\text{i} \right);\ \text{i=1,........,l}$$

subject to :

$$y_i(w^T\ \varphi\ (x_i) + \text{b}\ ) \geq (\ 1 - \xi\text{i});\ \xi\text{i} \geq 0$$

Here, training vectors x_i are mapped into a higher dimensional space by the function φ. Then SVM finds a linear separating hyper plane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore,
K(x_i , x_j) = φ (x_i)T φ (x_j) is called the kernel function [5].

CHAPTER 3
ARTIFICIAL NEURAL NETWORK AND CLASSIFIER DESIGN

3.1 Emergence of Artificial Neural Network

The human brain can be described as a biological neural network-an interconnected web of neurons transmitting elaborate patterns of electrical signals. Dendrites receive input signals and, based on those inputs, fire an output signal via an axon. A neuron fires only if the total signal received at the cell body from the dendrites exceeds a certain level (the firing threshold). Computer scientists have long been inspired by the human brain. In 1943, Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, developed the first conceptual model of an artificial neural network. In their paper, *A logical calculus of the ideas imminent in nervous activity* they describe the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs, and generates an output. Their work, and the work of many scientists and researchers that followed, was not meant to accurately describe how the biological brain works. Rather, an artificial neural network was designed as a computational model based on the brain to solve certain kinds of problem.

It's probably pretty obvious that there are problems that are incredibly simple for a computer to solve, but difficult for human. Take the square root of 964324 for example. A quick line of code produces the value 982, a number processing computed in less than a millisecond. There are, on the other hand, problems that are incredibly simple for human to solve, but not so easy for a computer. If a picture of a kitten or puppy is shown to a toddler and they'll be able to tell very quickly which one is what. Saying hello and shaking hand of a person in one morning and that person should be able to pick the other out of a crowd of people the next day. But, if a machine has to perform one of these tasks, then it would be a totally different case. Scientists have already spent entire careers researching and implementing complex solutions. The most common application of neural networks in computing today is

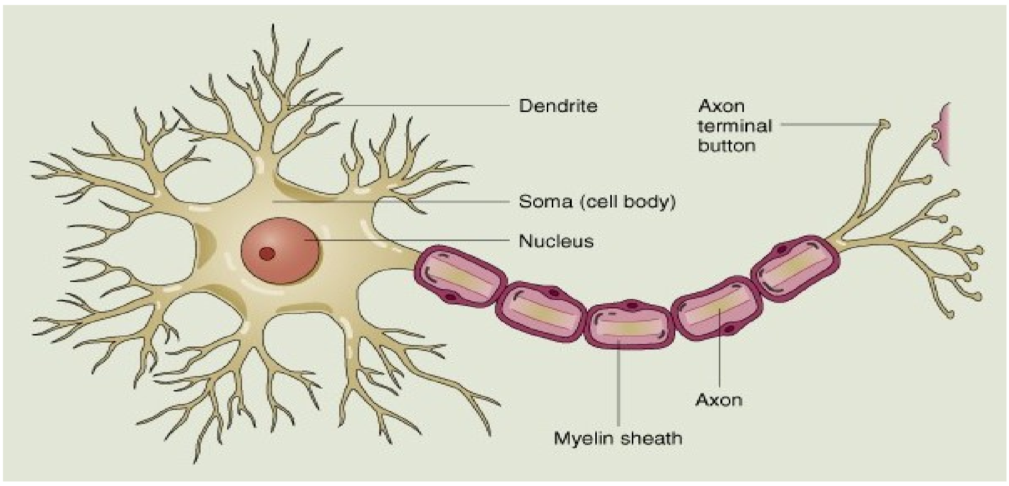


Figure 3.1: A Typical Neuron

to perform one of these easy-for-a-human, difficult-for-a-machine tasks often referred to as pattern recognition. Applications range from optical character recognition (turning printed or handwritten scans into digital text) to facial recognition.

3.2 What is an Artificial Neural Network (ANN)?

An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. One of the key elements of a neural network is its ability to learn. A neural network is not just a complex system, but a complex adaptive system, meaning it can change its internal structure based on the information flowing through it which is achieved through the adjustment of weights.

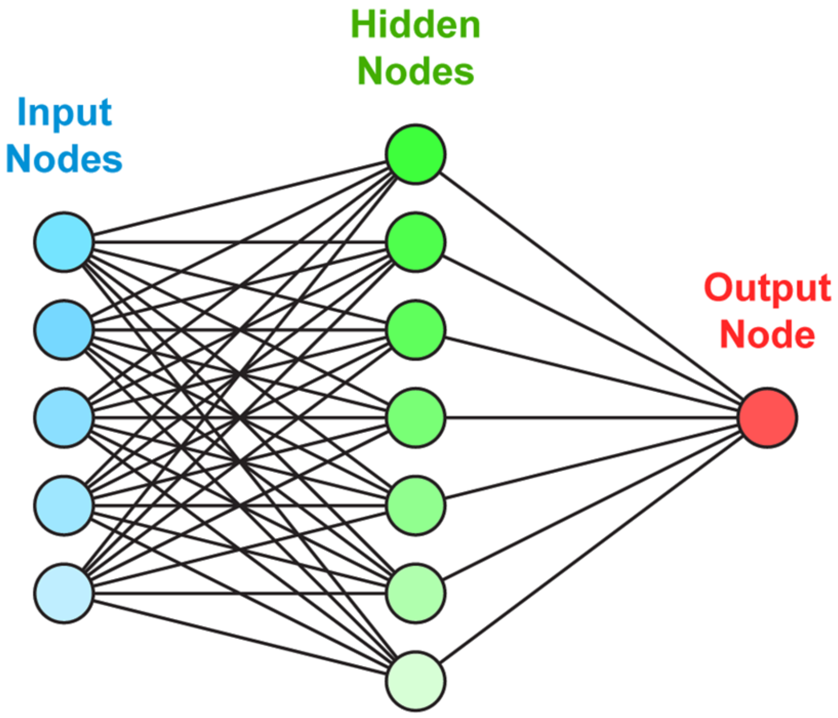


Figure 3.2: A Typical Neural Network

3.3 How does an Artificial Neural Network (ANN) work?

In Figure 3.2, each line represents a connection between two neurons and indicates the pathway for the flow of information. Each connection has a weight, a number that controls the signal between the two neurons. If the network generates a good output, there is no need to adjust the weights. However, if the network generates a poor output-an error, so to speak-then the system adapts, altering the weights in order to improve subsequent results. Basically, the internal activation or raw output of a neuron is a weighted sum of its inputs multiplied by weights connected to it. Mathematically of form Y=W_i X_i, where the elements of the feature vector (input) X_i are applied to the input nodes of the network. Then each one is multiplied by the corresponding weights, W_i (called synaptic weights or simply synapses). The products are summed up together with the threshold value, W_0(bias). The result then goes through a nonlinear device (activation function). The output, Y of a neuron is a function of the weighted sum of the inputs plus a bias.

3.4 Activation Function

An activation function is applied to the weighted sum of a neuron to produce smooth, continuous and monotonically increasing output. Commonly used forms are:- step function (f(x)=-1, if x<0 and f(x)=+1, if x>0) and sigmoid function (f(x) = $\frac{1}{1+e^{-ax}}$, a=slope parameter) [9]. The most common sigmoid function used is the logistic function: f(x) = $\frac{1}{1+e^{-x}}$.

3.5 Modes of Operation of Artificial Neural Network

A neural network is first subjected to training then testing. There are 2 ways to train a neural network, these are: supervised and unsupervised training. In supervised training, a set of training data are available whose true class is also provided simultaneously with the input (input-desired output) and the classifier is designed by exploiting this known information. And in case of unsupervised training, system is provided with a set of feature vectors x and the goal is to unravel the underlying similarities and cluster (group) similar vectors together. That means, with unsupervised learning, there is no feedback from the environment to indicate if the outputs are incorrect.

3.6 Types of Neural Network

3.6.1 Multi-Layer Feedforward Neural Networks

A multilayer feed-forward neural network consists of multiple layers of units, each of which are adaptive and parameterized by a weight vector. The network is not allowed to have cycles from later layers back to earlier layers, hence named feed-forward.

The feed forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes having no cycles or loops in the network. Feed-forward networks are advantageous as they are the fastest models to execute. Input, hidden and output layers units - each calculates its value of activation by taking the weighted sum of the outputs in the preceding layer after threshold adjustment. The result is then passed through a non-linear device (activation function) to produce the output of the neuron. Feed-forward ANNs are extensively used in pattern recognition [10].

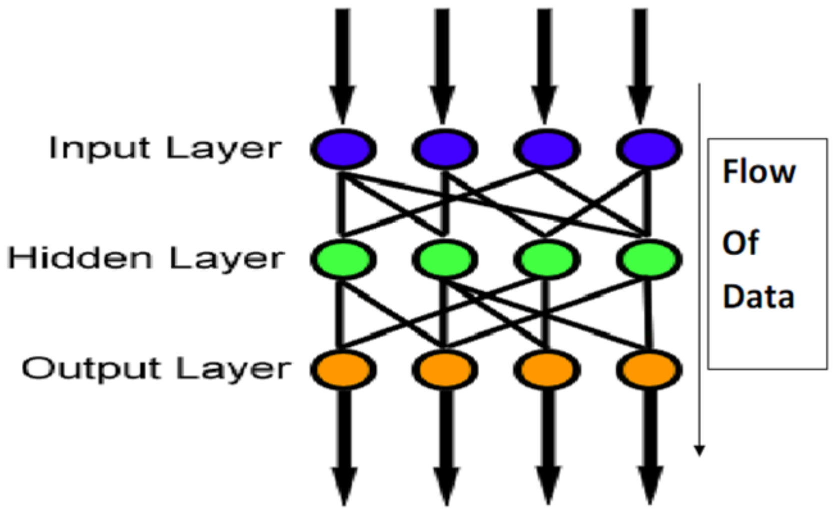


Figure 3.3: A Feedforward Neural Network

3.6.2 Recurrent Neural Networks

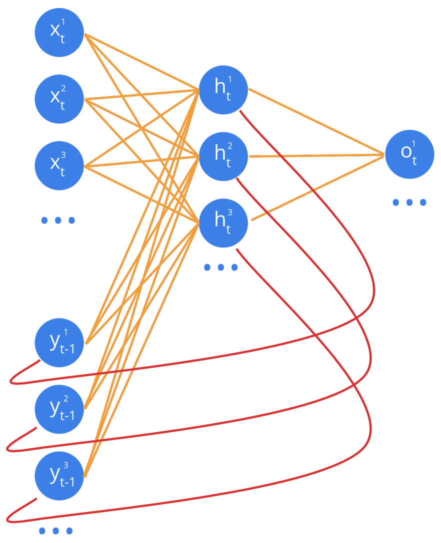


Figure 3.4: A Recurrent Neural Network

A recurrent neural network (RNN) is a class of neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feed forward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition, where they have achieved the best known results. The human brain is a recurrent neural network (RNN) as it consists of neurons with feedback connections. It can learn many behaviors, sequence processing tasks, algorithms, programs that are not learnable by traditional machine learning methods. This explains the rapidly growing interest in artificial RNNs for technical applications: general computers which can learn algorithms to map input sequences to output sequences, with or without a teacher. They are computationally more powerful and biologically more plausible than other adaptive approaches such as Hidden Markov models (no continuous internal states), feed forward networks and Support Vector Machine (no internal states at all) [11].

3.7 Back Propagation Algorithm

Training is the act of presenting the network with some samples and modifying the weights to better approximate the desired function. Most common method to obtain the weights in the network is Back propagation. The back propagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. Since this method requires computation of the gradient of the error function at each iteration step, it must be guaranteed that the continuity and differentiability of the error function hold. One of the more popular activation functions for back propagation networks is the sigmoid, a real function,

$$S_c : \mathbf{R} \rightarrow (0, 1)$$

defined by the expression,

$$S_c(\mathrm{x}) = \frac{1}{1+e^{-cx}}$$

The constant c can be selected arbitrarily and its reciprocal $\frac{1}{c}$ is called the temperature parameter in stochastic neural networks [12].

3.7.1 Steps of the Back Propagation Algorithm

Figure 3.5 shows the extended network for computation of the error function. In order to simplify the discussion, dealing is done with a single input-output pair (o, t) and generalize later to p training examples. The network has been extended with an additional layer of units. The right sides compute the quadratic deviation [$\frac{1}{2}(o_i^{(2)}$ - t_i)] for the i-th component of the output vector and the left sides store [($o_i^{(2)}$ - t_i)]. Each output unit i in the original network computes the sigmoid s and produces the output $o_i^{(2)}$. Addition of the quadratic deviations gives the error E. The error function for p input-output examples can be computed by creating p networks like the one shown, one for each training pair, and adding the outputs of all of them to produce the total error of the training set. After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps,

- Feed-forward computation
- Back propagation to the output layer
- Back propagation to the hidden layer
- Weight updates

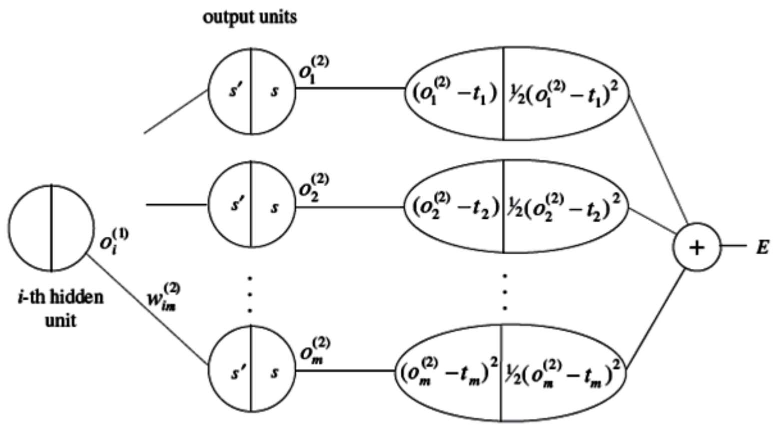


Figure 3.5: Extended multilayer network for the computation of Error

The algorithm is stopped when the value of the error function has become sufficiently small.

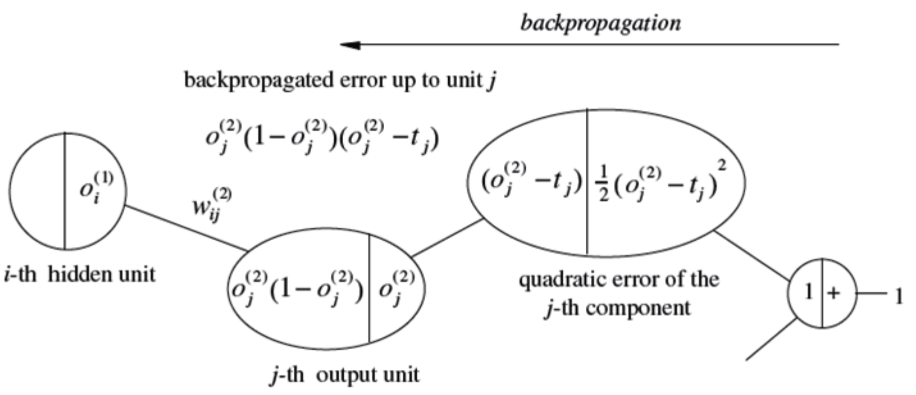

backpropagation
backpropagated error up to unit j
$o_j^{(2)}(1-o_j^{(2)})(o_j^{(2)}-t_j)$
$o_i^{(1)}$
$w_{ij}^{(2)}$
i-th hidden unit
$o_j^{(2)}(1-o_j^{(2)})$
$o_j^{(2)}$
j-th output unit
$(o_j^{(2)}-t_j)$
$\frac{1}{2}(o_j^{(2)}-t_j)^2$
quadratic error of the j-th component
1
+
1

Figure 3.6: Back propagation path up to output unit j

i. First step(feed-forward computation): The vector o is presented to the network. The vectors $o^{(1)}$ and $o^{(2)}$ are computed and stored. The evaluated derivatives of the activation functions are also stored at each unit.

ii. Second step(back propagation to the output layer): Searching for the first set of partial derivatives $\frac{\partial E}{\partial w_{ij}^{(2)}}$. The back propagation path from the output of the network up to the output unit j is shown in the diagram of Figure 3.6 .

From this path, by simple inspection all the multiplicative terms which define the back propagated error $\delta_j^{(2)}$ are collected . Therefore,

$$\delta_j^{(2)} = o_j^{(2)}\delta(1 - o_j^{(2)})\ (o_j^{(2)} - t_j)$$

and the desired partial derivative is,

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = [o_j^{(2)}(1\text{-}o_j^{(2)})(o_j^{(2)}\text{-}t_j)\ ]o_j^{(1)}$$

For this last step, it is considered that the weight $w_{ij}^{(2)}$ to be a variable and its input $o_i^{(1)}$ a constant.

iii. Third step(back propagation to the hidden layer): To compute the partial derivative $\frac{\partial E}{\partial w_{ij}^{(1)}}$. Each unit j in the hidden layer is connected to each unit q in the output layer with an edge of weight $w_{jq}^{(2)}$, for q = 1, . . . ,m. The back-propagated error up to unit j in the hidden layer must be computed taking into account all possible backward paths, as shown in Figure 3.7 .

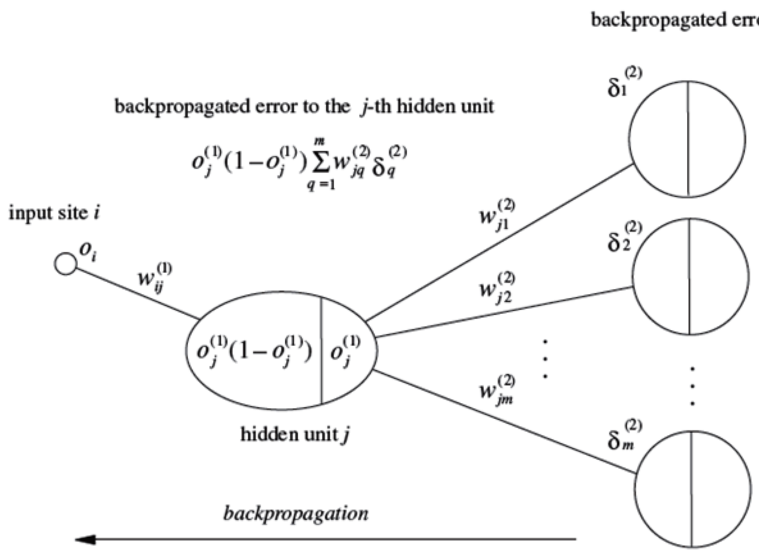


Figure 3.7: All paths up to input site i

The back-propagated error is then,

$$\delta_j^{(1)} = o_j^{(1)} \ (1 - o_j^{(1)}) \sum w_{jq}^{(2)} \delta_q^{(2)}$$

Therefore the partial derivative we are looking for is $\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_q^{(1)} \ o_i$ The back propagated error can be computed in the same way for any number of hidden layers.

iv. Fourth step(weight updates) : After computing all partial derivatives the network weights are updated in the negative gradient direction. A learning constant defines the step length of the correction. The corrections for the weights are given by,

$$\Delta w_{ij}^{(2)} = -\mu o_i^{(1)} \delta_j^{(2)}, \text{ for i=1,2,,k+1; j=1,2,........,m}$$

$$\text{and } \Delta w_{ij}^{(1)} = -\mu o_i \delta_j^{(1)}, \text{ for i=1,2,....,n+1; j=1,2,......,k}$$

where, used the convention that, $o_{n+1} = o_{k+1}^{(1)} = 1$. It is very important to make the corrections to the weights only after the back-propagated error has been computed for all units in the network. Otherwise the corrections become intertwined with the back-propagation of the error and the computed corrections do not correspond any more to the negative gradient direction.

In the case of p > 1 input-output patterns, an extended network is used to compute the error function for each of them separately. The weight corrections are computed for each pattern and so we get, for example, for weight $w_{ij}^{(1)}$ the corrections, $\Delta 1 w_{ij}^{(1)}$,$\Delta 2 w_{ij}^{(1)}$,......, $\Delta p w_{ij}^{(1)}$ the necessary update in the gradient direction is then,

$$\Delta 1 w_{ij}^{(1)} = \Delta 1 w_{ij}^{(1)} + \Delta 2 w_{ij}^{(1)} + ...+ \Delta p w_{ij}^{(1)}$$

Specifically, batch or off-line updates when the weight corrections are made in this way. Often, however, the weight updates are made sequentially after each pattern presentation (this is called on-line training). In this case the corrections do not exactly follow the negative gradient direction, but if the training patterns are selected randomly the search direction oscillates around the exact gradient direction and, on average, the algorithm implements a form of descent in the error function. The rationale for using on-line training is that adding some noise to the gradient direction can help to avoid falling into shallow local minima of the error function. Also, when the training set consists of thousands of training patterns, it is very expensive to compute the exact gradient direction since each epoch (one round of presentation of all patterns to the network) consists of many feed-forward passes and on-line training becomes more efficient [12].

3.8 Classifier Design

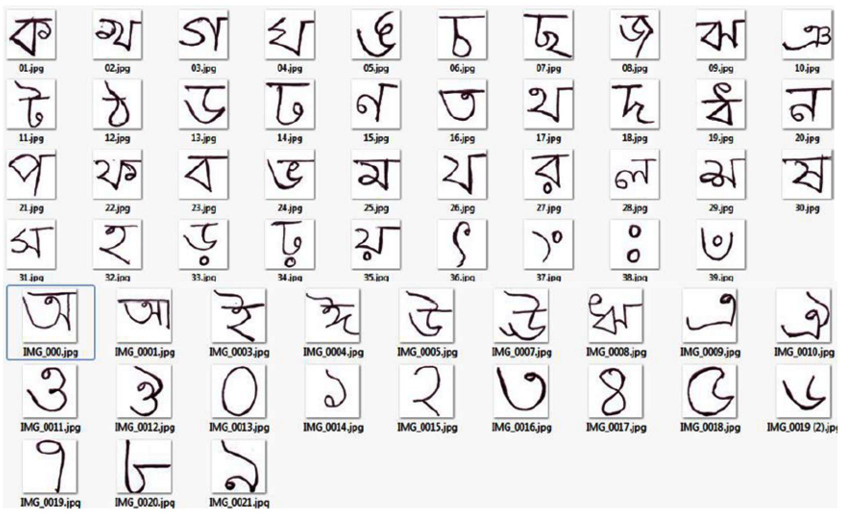
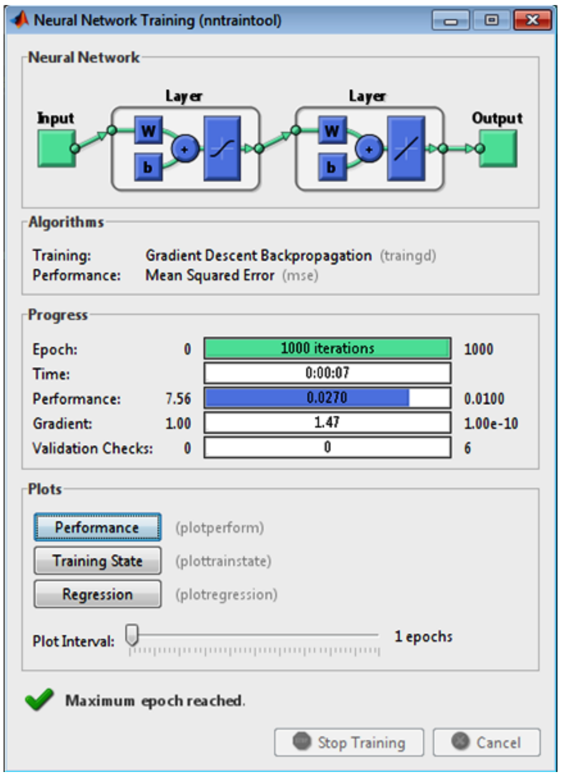


Figure 3.8: Data Patterns for Feature Generation



Algorithms

Training: Gradient Descent Backpropagation (traingd)
Performance: Mean Squared Error (mse)

Progress

Epoch:	0	1000 iterations	1000
Time:		0:00:07	
Performance:	7.56	0.0270	0.0100
Gradient:	1.00	1.47	1.00e-10
Validation Checks:	0	0	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

✓ Maximum epoch reached.

Figure 3.9: Neural Network forming Classifier

Classifier has been designed in the spirit of back propagation algorithm. The attempt is to study the working principle of the Back-Propagation Algorithm and its generalizing capability using a simple procedure. An image is specified right at the end, over which the testing takes place. Hand-written characters have been used as data patterns for feature generation. Features that are generated as a result of feature extraction processes which mainly include longest run features, shadow features and quad-tree features are also used as inputs to separate Neural Networks. That means, array of features (feature vectors) have been used as inputs to Neural networks individually. For instance, The neural network resulted when longest run features of first 10 Bangla vowels are used as training data pattern is shown in Figure 3.9.

3.8.1 Properties of Classifier

Neural Network Object:

- Architecture: numInputs: 1

  numLayers: 2

  biasConnect: [1; 1]

  inputConnect: [1; 0]

  layerConnect: [0 0; 1 0]

  outputConnect: [0 1]

  numOutputs: 1 (read-only)

  numInputDelays: 0 (read-only)

  numLayerDelays: 0 (read-only)

- Subobject Structures:

  inputs: {1x1 cell} of inputs

  layers: {2x1 cell} of layers

  outputs: {1x2 cell} containing 1 output

  biases: {2x1 cell} containing 2 biases

  inputWeights: {2x1 cell} containing 1 input weight

  layerWeights: {2x2 cell} containing 1 layer weight

- Functions:

  adaptFcn: 'trains'

  divideFcn: (none)

  gradientFcn: 'calcgrad'

  initFcn: 'initlay'

  performFcn: 'mse'

  plotFcns: {'plotperform','plottrainstate','plotregression'} trainFcn: 'traingd'

- Parameters:

  adaptParam: .passes

divideParam: (none)
gradientParam: (none)
initParam: (none)
performParam: (none)

- Weight and Bias values:
  IW: {2x1 cell} containing 1 input weight matrix
  LW: {2x2 cell} containing 1 layer weight matrix
  b: {2x1 cell} containing 2 bias vectors

- Other:
  name: ' '
  userdata: (user information)

The network is operating based on maximum matching technique particularly, cross-correlation between testing and training pattern. If features of 10th vowel is provided as test pattern, then the networks return the index values corresponding to which each one of them finds maximum match and then decider compiles the results to provide the final decision. The network gives the output of index-matching array as (in a sample case), index-match =

```
-0.0315
-0.0569
0.0375
-0.0611
0.1260
0.1010
0.1991
-0.0758
0.2186
0.9611
```

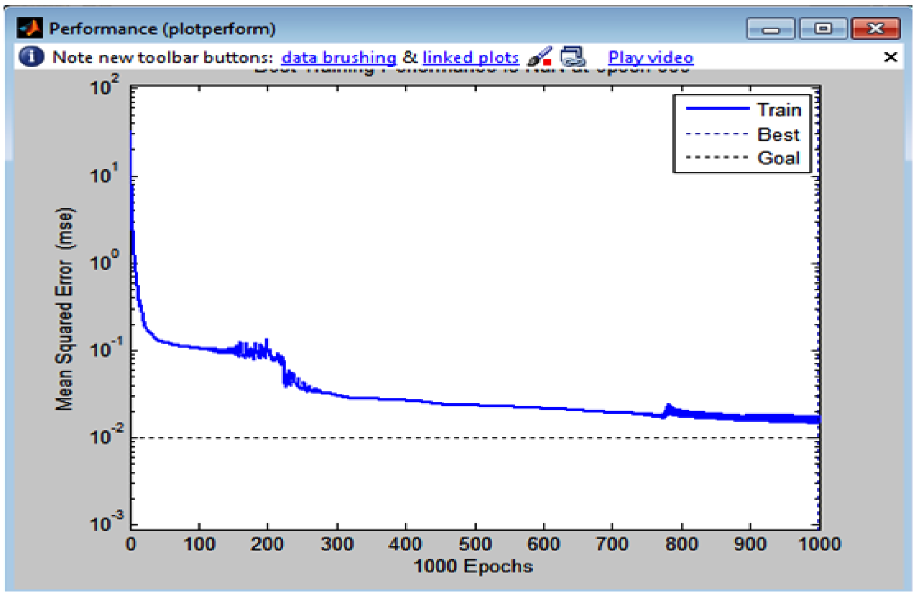


Figure 3.10: Sample Performance Curve

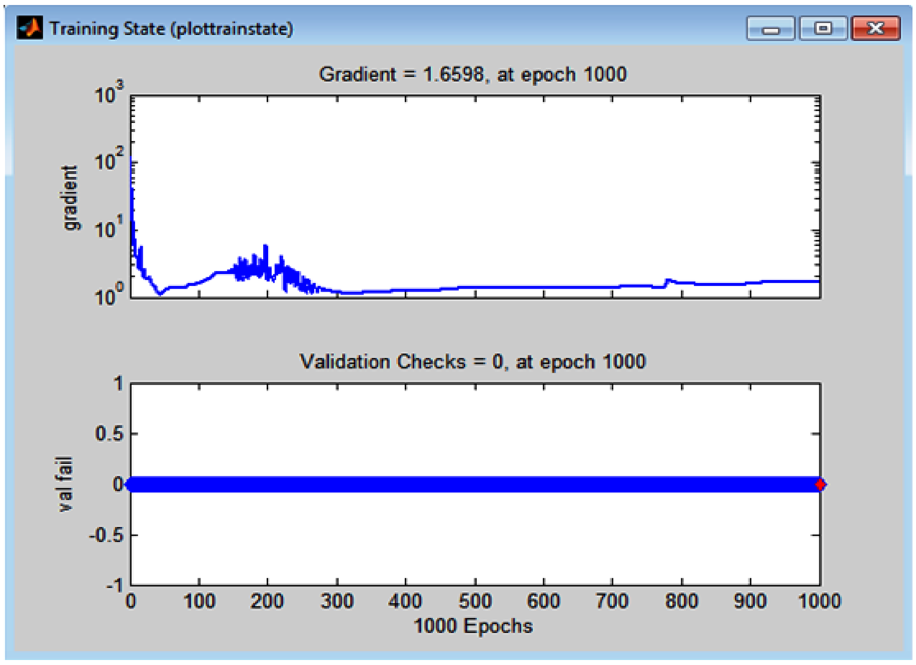


Figure 3.11: Sample Training State Curve

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Sample Inputs for Training of Neural Network

Several feature vectors are fed to the neural networks during the training phase that means, these are the inputs to the classifiers to ensure expected performance during testing phase.A few of them are enlisted,

Table 4.1: Quad Tree-based Feature Vector(Bangla vowels)

Sample	**Centroid1**	**Centroid2**	**Centroid3**	**Centroid4**	**Center of Gravity**
1	(0.30,0.67)	(0.07,0.11)	(0.52,0.44)	(0.20,0.36)	(0.27,0.39)
2	(0.26,0.12)	(0.14,0.42)	(1.32,1.01)	(0.27,0.59)	(0.50,0.54)
3	(0.61,0.90)	(0.32,0.49)	(1.23,0.75)	(0.90,0.52)	(0.76,0.67)
4	(0.42,0.55)	(0.07,0.16)	(0.60,0.24)	(0.38,0.52)	(0.37,0.37)
5	(0.56,0.77)	(0.43,0.65)	(0.59,0.21)	(0.66,0.34)	(0.27,0.49)
6	(0.03,0.09)	(0.88,0.86)	(0.93,0.71)	(0.98,0.78)	(0.71,0.61)
7	(1.03,0.58)	(0.29,0.61)	(0.12,0.07)	(0.21,0.15)	(0.41,0.35)
8	(0.00,0.00)	(0.34,0.41)	(0.81,1.28)	(0.29,0.54)	(0.36,0.56)
9	(0.00,0.00)	(0.85,0.71)	(1.41,1.04)	(1.06,1.28)	(0.83,0.76)
10	(0.56,1.03)	(0.63,0.88)	(1.21,0.65)	(0.69,0.65)	(0.77,0.81)
11	(0.46,0.73)	(0.55,0.66)	(1.65,1.06)	(1.02,0.94)	(0.92,0.85)

Table 4.2: Longest Run Feature Vector(Bangla Vowels)

Samples	**Feature Vectors**
1	0 0 0 1 1 0 4 1 2 4 0 0 0 0 0 2 0 0 2 1 1 1 1 2 1 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 3 3 2 2 2 1 2 3 2 2 1 2 3 2 1 1 1 0 0 0 0 0 0 0 0
3	0 1 0 1 9 1 0 1 0 5 7 8 1 2 1 1 1 2 1 1 2 2 2 4 2 2 2 2 3 3 3 1
4	0 0 0 0 0 0 1 1 0 0 7 2 1 2 0 1 2 2 0 2 2 1 1 0 1 1 2 0 0 0 0 0
5	0 0 0 3 3 3 2 0 0 0 1 0 12 1 0 1 1 0 0 1 1 1 1 7 1 1 2 7 0 0 0 0
6	0 0 0 1 6 2 2 0 0 0 0 1 2 7 1 2 1 1 2 1 1 1 1 2 2 2 3 5 1 3 6 0
7	0 0 0 0 0 0 1 2 2 2 1 1 2 1 1 6 2 0 1 2 1 1 1 2 1 0 1 0 0 0 0 0
8	0 0 0 0 6 2 2 5 1 1 1 1 0 1 1 1 1 0 1 2 5 3 2 2 1 0 0 0 0 0 0 0
9	0 0 1 1 1 1 2 2 1 1 1 1 2 3 2 3 2 4 1 1 1 2 2 1 1 2 7 8 4 2 0 0
10	0 0 5 3 3 2 2 3 3 1 1 1 1 1 1 6 4 2 1 2 2 2 2 1 1 2 2 9 0 0 0 0
11	0 1 2 1 3 2 2 2 3 3 2 4 3 2 2 3 3 4 3 4 3 1 2 1 1 1 1 2 2 2 7 0

Table 4.3: Shadow Feature Vector(First Bangla vowel)

Sample : First Bangla Vowel
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1
0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

4.2 Testing Phase

4.2.1 Bangla Consonants

Neural Networks have been trainred with the feature vectors resulted from feature generation process and then tested with all the characters individually. Outputs of classifier for feature set 1, 2, 3 have been enlisted sequentially when the networks are tested with first ten Bangla Consonants,

Table 4.4: Test for 1^{st} Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.993652	0.775908	1.009660	1
2	-0.064340	0.127957	-0.112392	0
3	0.044930	0.021575	-0.160095	0
4	0.030812	-0.021314	0.004965	0
5	0.119977	0.060678	0.109448	0
6	0.252205	-0.096063	0.107831	0
7	0.030292	-0.080495	0.142268	0
8	-0.079425	-0.029955	-0.135517	0
9	-0.078727	0.032368	-0.250091	0
10	-0.091589	-0.097475	-0.102950	0

Table 4.5: Test for 2nd Bangla Consonant

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.042410	0.150980	0.182294	0
2	0.913795	1.109801	0.886192	1
3	0.046132	0.436933	0.050558	0
4	0.054984	-0.194422	0.043476	0
5	0.166889	-0.221653	0.211133	0
6	0.096036	0.205377	0.100682	0
7	0.012262	0.114841	0.133164	0
8	0.095773	-0.027637	0.006061	0
9	-0.166470	0.014565	0.063915	0
10	0.079136	-0.049733	-0.096952	0

Table 4.6: Test for 3rd Bangla Consonant

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	0.147911	-0.272226	-0.055487	0
2	0.101937	-0.100868	-0.061789	0
3	1.012271	0.747678	1.004654	1
4	-0.016675	0.086077	-0.138352	0
5	-0.104847	-0.069458	-0.080781	0
6	0.183192	-0.081727	0.113287	0
7	-0.117609	-0.159792	-0.091024	0
8	-0.078895	0.125452	0.089317	0
9	-0.043996	0.064973	-0.124487	0
10	-0.075507	0.002020	-0.115094	0

Table 4.7: Test for 4^{th} Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.015733	-0.007526	-0.128882	0
2	0.058092	-0.145504	-0.061582	0
3	0.057067	0.176815	-0.098426	0
4	0.828585	0.586801	0.997552	1
5	-0.023923	0.233012	-0.069316	0
6	-0.193084	0.179522	0.084185	0
7	0.225751	-0.175286	-0.034596	0
8	-0.006827	0.023256	0.021659	0
9	-0.119736	-0.519571	0.143599	0
10	-0.131060	0.316050	-0.001119	0

Table 4.8: Test for 5^{th} Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.181591	0.381812	-0.155475	0
2	-0.049062	0.020035	0.067862	0
3	0.089220	0.002886	-0.299481	0
4	0.027254	0.245200	-0.230000	0
5	1.022899	0.892923	0.898145	1
6	-0.038855	-0.407382	0.015315	0
7	0.150143	0.232406	0.004889	0
8	-0.055871	0.037050	0.134626	0
9	-0.066252	-0.186057	-0.129410	0
10	0.080297	0.050485	-0.148674	0

Table 4.9: Test for 6th Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.000578	-0.132813	-0.004566	0
2	0.100003	0.078300	-0.011929	0
3	0.220817	0.135316	-0.035618	0
4	-0.009069	-0.003589	-0.020741	0
5	0.086751	-0.327439	-0.027188	0
6	0.838180	0.828382	0.960004	1
7	-0.087460	0.398091	0.077360	0
8	0.231456	-0.148418	-0.005748	0
9	0.037834	0.020606	-0.000351	0
10	0.022162	-0.423303	-0.018389	0

Table 4.10: Test for 7th Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.068912	0.067291	0.046112	0
2	-0.046129	-0.112551	0.041814	0
3	-0.130442	0.045187	-0.033502	0
4	0.000335	0.009108	-0.282848	0
5	0.073789	0.013637	-0.052958	0
6	0.047775	0.183934	0.087591	0
7	0.895572	0.868636	0.876633	1
8	-0.030165	0.016432	-0.089205	0
9	0.082139	-0.219639	-0.190661	0
10	-0.136069	-0.139548	0.007902	0

Table 4.11: Test for 8th Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.086053	-0.573843	0.117890	0
2	0.169649	-0.158308	0.022724	0
3	0.140492	0.099119	0.102655	0
4	-0.081143	0.241674	0.004018	0
5	-0.107395	-0.005858	-0.019542	0
6	-0.036772	-0.382217	0.085177	0
7	-0.049436	-0.022465	-0.048565	0
8	0.940161	1.025083	1.103690	1
9	-0.197509	-0.357408	0.003463	0
10	-0.080323	0.125407	0.018130	0

Table 4.12: Test for 9th Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.009198	-0.145395	-0.063471	0
2	-0.022642	0.024598	0.126339	0
3	-0.042835	-0.181625	-0.148712	0
4	0.027533	-0.139793	0.050590	0
5	-0.045046	-0.147586	0.221881	0
6	-0.083509	0.128605	-0.015147	0
7	-0.073799	-0.319208	0.053872	0
8	-0.029351	-0.029104	0.079631	0
9	0.891338	1.357470	1.174329	1
10	-0.090953	0.120911	-0.151763	0

Table 4.13: Test for 10th Bangla Consonant

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.006396	0.048564	-0.007698	0
2	-0.046380	-0.105488	-0.004570	0
3	-0.051721	0.082673	-0.031450	0
4	-0.007201	0.065092	0.056613	0
5	-0.024546	0.048133	-0.080968	0
6	0.064959	-0.085842	-0.095923	0
7	0.022226	-0.099129	0.033142	0
8	-0.060709	-0.142825	-0.021188	0
9	-0.021608	0.100841	0.007580	0
10	0.977892	0.846777	1.029104	1

4.2.2 Bangla Vowels

Outputs of classifier for feature set 1, 2, 3 have been enlisted sequentially when the networks are tested with the Bangla Vowels,

Table 4.14: Test for 1st Bangla Vowel

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	1.015342	0.788991	0.985174	1
2	0.124139	0.124130	-0.010840	0
3	0.102138	0.119756	-0.006071	0
4	0.001780	0.261951	0.013090	0
5	-0.048485	-0.775356	0.000452	0
6	0.058636	-0.607053	0.021240	0
7	-0.062347	-0.146250	0.005501	0
8	-0.092818	0.020854	0.008499	0
9	0.048418	0.083895	-0.001970	0
10	-0.047663	0.283794	0.008785	0
11	-0.002465	0.143186	0.006284	0

Table 4.15: Test for 2^{nd} Bangla Vowel

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.179665	0.252497	0.150892	0
2	1.319636	0.776328	1.037604	1
3	0.048369	-0.117731	-0.011150	0
4	0.026325	0.370670	0.010287	0
5	-0.333443	-0.169342	-0.039392	0
6	0.340861	0.741134	0.148026	0
7	-0.009611	0.014209	0.130546	0
8	-0.087281	0.350608	-0.103287	0
9	0.248169	-0.437089	0.194667	0
10	-0.010864	0.011621	0.061363	0
11	0.026334	-0.190740	-0.133166	0

Table 4.16: Test for 3^{rd} Bangla Vowel

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.026285	0.217590	-0.016641	0
2	-0.027903	-0.222746	0.025343	0
3	1.015808	0.636087	0.987600	1
4	-0.005743	0.094677	-0.025322	0
5	-0.011341	0.010161	-0.101047	0
6	0.044310	0.078500	0.068952	0
7	0.013123	0.024733	-0.066264	0
8	-0.010797	-0.117651	0.015648	0
9	-0.007711	-0.195817	-0.076707	0
10	0.021746	-0.025805	-0.018823	0
11	-0.015934	0.089628	0.020329	0

Table 4.17: Test for 4^{th} Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	0.053580	-0.154313	-0.120174	0
2	0.042274	-0.428809	-0.018960	0
3	-0.054239	-0.107946	0.080405	0
4	0.894697	0.463377	0.748503	1
5	0.092601	0.104455	0.160118	0
6	-0.121210	0.334174	-0.154594	0
7	0.107035	-0.338772	-0.096244	0
8	0.077764	-0.043237	0.068097	0
9	0.009813	-0.405902	0.024738	0
10	-0.046216	-0.144083	0.074908	0
11	-0.109605	0.011480	0.256606	0

Table 4.18: Test for 5^{th} Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.012458	-0.028986	-0.053120	0
2	0.046652	-0.171024	-0.053120	0
3	-0.023814	0.418831	0.042099	0
4	0.028930	0.345443	-0.093467	0
5	1.058113	0.425186	0.924543	1
6	0.015366	-0.154908	-0.071618	0
7	0.022842	0.143347	0.047145	0
8	-0.041765	0.294902	-0.085969	0
9	0.009082	-0.163565	0.009481	0
10	-0.020058	-0.124901	-0.030766	0
11	-0.004460	0.377604	-0.001215	0

Table 4.19: Test for 6th Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	0.046229	0.202222	0.071193	0
2	-0.181272	-0.261490	-0.198125	0
3	-0.063912	-0.101766	-0.094248	0
4	0.106126	0.073958	0.017365	0
5	-0.034026	0.229317	0.104025	0
6	1.029015	0.859198	1.008866	1
7	0.032159	0.267874	-0.052712	0
8	0.130948	0.372032	-0.125423	0
9	-0.063166	0.099014	0.002727	0
10	-0.044569	-0.052793	0.023589	0
11	-0.105274	0.054006	0.093855	0

Table 4.20: Test for 7th Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.127070	0.122926	0.036902	0
2	-0.041648	-0.076250	-0.007750	0
3	0.167163	0.241793	-0.035622	0
4	-0.354806	0.147243	-0.015162	0
5	-0.188678	-0.102057	0.059080	0
6	-0.039013	0.091968	-0.017092	0
7	0.929778	0.722625	0.980892	1
8	-0.052864	-0.005108	-0.050266	0
9	0.028608	-0.180658	0.023355	0
10	-0.141912	0.078189	-0.027813	0
11	-0.157649	0.086846	0.021151	0

Table 4.21: Test for 8th Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.027453	0.046391	0.011568	0
2	0.041935	0.280687	0.066252	0
3	0.089901	0.216210	-0.085728	0
4	-0.041838	0.037098	-0.025255	0
5	0.057721	-0.108371	0.071038	0
6	0.017223	0.300308	0.068028	0
7	0.010876	0.232713	-0.000377	0
8	0.992062	1.104313	1.028534	1
9	-0.003623	-0.263460	-0.019488	0
10	0.079290	-0.055744	0.019191	0
11	0.058788	0.183731	-0.024980	0

Table 4.22: Test for 9th Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.029301	-0.595749	0.000240	0
2	-0.072668	0.122950	-0.048243	0
3	0.004015	-0.046074	-0.153880	0
4	0.040839	0.294236	-0.010654	0
5	-0.023507	0.119802	0.015679	0
6	-0.007187	0.047937	-0.061078	0
7	-0.017847	0.490478	0.071942	0
8	-0.077111	0.253135	-0.158397	0
9	1.025588	0.707878	0.759562	1
10	0.089485	0.040044	0.169624	0
11	-0.064668	0.353217	0.009254	0

Table 4.23: Test for 10^{th} Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.053048	-0.046445	0.204205	0
2	-0.023094	0.547301	0.022251	0
3	-0.003064	0.104606	0.152758	0
4	-0.026520	-0.519856	-0.089255	0
5	-0.043207	0.409835	0.244850	0
6	0.054418	0.203680	0.076643	0
7	-0.046437	-0.148128	0.148563	0
8	0.055735	0.424765	-0.159072	0
9	0.019433	0.016375	0.140847	0
10	0.900793	0.813030	0.953546	1
11	0.001331	0.204122	-0.075845	0

Table 4.24: Test for 11^{th} Bangla Vowel

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	0.099140	-0.134274	-0.024082	0
2	0.034072	-0.002399	0.054992	0
3	-0.069021	0.082422	-0.025662	0
4	0.073243	0.200558	-0.102893	0
5	-0.067652	-0.256221	0.014939	0
6	0.034168	-0.205295	-0.076852	0
7	0.057622	-0.172987	-0.034277	0
8	0.066946	0.023861	0.007545	0
9	-0.011326	0.128488	0.099788	0
10	0.006990	-0.307785	-0.038612	0
11	1.056845	0.987285	0.887734	1

4.2.3 Bangla Numerals

Outputs of classifier for feature set 1, 2, 3 have been enlisted sequentially when the networks are tested with the Bangla Numerals,

Table 4.25: Test for 1st Bangla Numeral

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	1.015342	0.581498	1.003273	1
2	0.124139	-0.024589	0.196220	0
3	0.102138	0.510759	-0.108005	0
4	0.001780	0.367731	0.035329	0
5	-0.048485	-0.125310	0.010606	0
6	0.058636	0.147033	0.078735	0
7	-0.062347	0.262405	-0.003842	0
8	-0.092818	0.057164	-0.169264	0
9	0.048418	0.066272	0.062760	0
10	-0.047663	0.087971	0.089598	0

Table 4.26: Test for 2nd Bangla Numeral

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	-0.133270	-0.100108	-0.070703	0
2	0.907111	0.771873	1.068458	1
3	0.000678	0.435306	-0.000481	0
4	-0.041867	0.158054	0.004861	0
5	0.418835	0.135547	-0.065149	0
6	-0.230618	-0.147822	-0.050446	0
7	0.006360	0.298890	-0.119675	0
8	0.151674	0.177369	-0.031354	0
9	0.175824	-0.357904	0.052884	0
10	0.343323	-0.261845	-0.091157	0

Table 4.27: Test for 3rd Bangla Numeral

Training Sample	Feature Set 1	Feature Set 2	Feature Set 3	Decider
1	0.054578	0.272131	-0.096715	0
2	0.018932	0.240760	0.047767	0
3	0.945107	0.625783	0.821334	1
4	-0.000696	-0.221273	0.066924	0
5	0.029151	0.150467	0.041447	0
6	-0.080689	-0.282723	0.025172	0
7	-0.110553	-0.031686	-0.139564	0
8	0.016755	-0.060339	-0.098136	0
9	-0.097225	0.386729	0.002872	0
10	0.026635	0.175426	-0.086792	0

Table 4.28: Test for 4^{th} Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.016449	0.241948	0.083018	0
2	-0.008248	-0.118070	-0.025704	0
3	0.025736	-0.087480	-0.005514	0
4	0.999100	0.752189	0.950748	1
5	0.020919	0.138311	0.117124	0
6	-0.003210	-0.067652	-0.021323	0
7	0.000638	-0.248285	0.102321	0
8	0.017465	0.023128	-0.090772	0
9	-0.003996	-0.050014	-0.028735	0
10	-0.006614	0.021042	0.052002	0

Table 4.29: Test for 5^{th} Bangla Numeral

Training Sample	**Classifier1**	**Classifier2**	**Classifier3**	**Decider**
1	0.024383	-0.039754	-0.012288	0
2	0.002482	0.026001	-0.013970	0
3	0.037775	0.021365	-0.018983	0
4	-0.033851	-0.333348	-0.000258	0
5	1.026389	0.834967	1.012255	1
6	-0.018999	0.219706	-0.001214	0
7	-0.019033	0.156926	-0.024675	0
8	0.017331	0.049458	-0.031099	0
9	0.007182	0.115097	0.011009	0
10	-0.036295	0.314935	0.021427	0

Table 4.30: Test for 6th Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.029994	-0.018714	-0.043045	0
2	-0.000883	0.059381	-0.168652	0
3	-0.020128	0.099730	0.150713	0
4	-0.010254	-0.023970	0.043387	0
5	0.028150	-0.301664	0.034240	0
6	0.984255	1.028340	1.039798	1
7	0.028602	0.235663	-0.012663	0
8	-0.036943	0.161465	-0.010180	0
9	-0.007025	0.315166	0.050891	0
10	0.004763	-0.104099	0.074811	0

Table 4.31: Test for 7th Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.064162	-0.104369	0.074811	0
2	0.044082	0.225153	-0.117478	0
3	0.109481	-0.097777	0.088211	0
4	0.080621	0.000079	-0.180909	0
5	-0.104719	-0.153788	0.097938	0
6	0.010954	-0.193238	0.464761	0
7	0.917494	0.795323	0.758246	1
8	0.077999	-0.367415	-0.101449	0
9	0.018109	0.296208	0.009148	0
10	0.160680	0.183967	-0.391378	0

Table 4.32: Test for 8th Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	-0.041003	0.354793	-0.199303	0
2	-0.045856	-0.056872	-0.244234	0
3	0.056466	0.174447	0.235662	0
4	-0.015817	0.112789	0.078250	0
5	-0.030714	-0.232695	0.034172	0
6	0.091543	-0.051234	-0.143736	0
7	-0.004668	-0.333407	-0.009265	0
8	1.000507	1.243552	0.857759	1
9	0.025701	0.023861	-0.198459	0
10	-0.007341	0.490373	0.085963	0

Table 4.33: Test for 9th Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.001859	0.125772	-0.035989	0
2	0.003046	0.139160	0.026083	0
3	-0.002068	0.011438	0.009657	0
4	-0.002188	0.127900	-0.007183	0
5	-0.005154	0.099788	0.022824	0
6	0.004601	-0.126988	0.012833	0
7	-0.002162	-0.025202	0.004376	0
8	0.005112	-0.145194	-0.004601	0
9	1.002505	0.917386	0.991900	1
10	0.002783	-0.055800	0.006010	0

Table 4.34: Test for 10th Bangla Numeral

Training Sample	**Feature Set 1**	**Feature Set 2**	**Feature Set 3**	**Decider**
1	0.024353	-0.311239	0.094691	0
2	-0.124533	0.186661	-0.009212	0
3	0.011268	0.055348	-0.058371	0
4	-0.087668	0.062961	0.065742	0
5	-0.008110	0.012378	-0.056542	0
6	0.003307	0.087440	0.049616	0
7	-0.119212	-0.137138	0.092840	0
8	-0.002184	-0.206419	0.097009	0
9	-0.204658	0.283172	-0.155986	0
10	1.009706	0.654862	1.077152	1

CHAPTER 5

CONCLUSION

In spite of Bangla being the 5th among the most widely spoken languages with over 200 million speakers, its computerization has not gone much yet. In fact, dedicated research and development for Bangla computerization has begun from last decade. In digitization, more specifically computerization of any language, the first and foremost step is to develop an efficient and effective Optical Character Recognition (OCR) system for the respective language. In order to maintain a huge database of samples, in order to store millions of formats into electronic form, OCR is the prime tool. Since in no way, manual processing can be a thinkable option as it would hamper efficiency, effectiveness, correctness drastically. It's a matter of great hope and aspiration that this particular field that means, Bangla language development is now in view of our government also. Government has taken initiative by incorporating properly funded projects in this regard. The objective of our work is to develop a method of recognition of Bangla characters with a major improvement in efficiency. Artificial neural networks are used as their performance in character recognition is optimum due to their high noise tolerance and the ability to yield excellent results. Character recognition and classification is highly dependent on feature extraction step since a poorly chosen set of features will yield poor classification rates by any neural network.

5.1 Limitations

No research work is free from laggings so does our one. We have considered only individual characters. That means, processing of a complete Bangla document is out of scope of our work. No compound characters have been considered. Characters must be processed individually.

Accuracy obtained for supervised character recognition is satisfactory but in unsupervised case, performance is not up to the mark. We have considered only a few of the features. More features can be included to increase the accuracy rate by implementing more classifiers adopting different methodologies.

5.2 Recommendations for Further Expansion

Development of an efficient Bangla character recognition system can bring many related research works on the track. It can be used in the fields of Bangla Spelling Checker, Grammar Checker, Font Converter and many other relevant works. Our procedure is mainly efficient in single character recognition, with essential modifications it can be used to recognize compound letters, words and by enhancing the volume of pre-processing tasks, it can be used to process a whole Bangla document even. The accuracy rate can be improved remarkably for unsupervised patterns if a combination of different classifiers based on Hidden Markov Model, Viterbi algorithm, SVM, Mahalanobis distance is used. An improvised technique basing on this can be used to recognize signatures for authentication purpose.

REFERENCES

[1] Jesse Hansen, “A Matlab Project in Optical Character Recognition (OCR)”, *http://www.ele.uri.edu/hansenj/projects/ele585/OCR/OCR.pdf*,[Lastvisited:7/12/2013].

[2] Madhuri Latha, Chakravarthy, “An Improved Bernsen Algorithm Approaches For License Plate Recognition”, *IOSR Journal of Electronics and Communication Engineering(IOSR-JECE)*, ISSN:2278-2834, ISBN:2278-8735, vol. 3, pp. 01-05, Sep-Oct. 2012. Availabe at: http://www.iosrjournals.org/iosr-jece/papers/Vol3-Issue4/A0340105.pdf, [Last visited: 8/12/2013].

[3] Khurram Khurshid, Imran Siddiqi, Claudie Faure, Nicole Vincent,“Comparison of Niblack inspired Binarization methods for ancient documents”, *Proc. of SPIE-IST Electronic Imaging,SPIE-IST*, vol. 7, 2009.

[4] S.Basu, N.Das, R.Sarkar, M.Kundu, M.Nasipuri, D.K.Basu, “Handwritten Bangla Alphabet Recognition using an MLP Based Classifier”, *Ind. Proc. of the 2nd National Conf. on Computer Processing of Bangla*, pp. 407-417, Dec.2005.

[5] N.Das, B.Das, R.Sarkar, S.Basu, M.Kundu, M.Nasipuri,“Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier”, *Journal of Computing*, vol. 2, ISSN:2151-9617, Feb. 2010.

[6] Alam, M. M.,Kashem, “A complete bangla OCR system for printed characters”, *Journal of Computer and Information Technology* vol. 1, pp. 30-35.

[7] Pal, U. and Chaudhuri B. B., “Automatic recognition of unconstrained off-line bangla Handwritten Numerals”, *Int. Conf. on Multimodal Interfaces, Lecture Notes in Computer Science (LNCS)*, pp. 371-378, 1948.

[8] Roy A., Bhowmik, T.K., Parui, S.K. and Roy, U., “A Novel Approach to Skew Detection and Character Segmentation for Handwritten Bangla Words”, *Proceedings of the Digital Imaging Computing: Techniques and Applications*, pp. 30.

[9] Sergio Theodoridis, Konstantinos Koutroumbas, *Pattern Recognition*, 4^{th} edition.

[10] *http://en.wikipedia.org/wiki/Artificial_neural_network*, [Last visited: 8/12/2013].

[11] *http://en.wikipedia.org/wiki/Recurrent_neural_network*, [Last visited: 8/12/2013].

[12] *http://page.mi.fuberlin.de/rojas/neural/chapter/K7.pdf*, [Last visited: 8/12/2013].

APPENDIX A