FEATURE BASED OPTICAL BENGALI CHARACTER RECOGNITION USING MULTIPLE BACK PROPAGATION NEURAL NETWORKS

by

Kazi Wali Ullah

Exam Roll – 1404

Fariha Nazmul

Exam Roll – 1436

(Session 2005-2006)

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science (Hons.) in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF DHAKA

BANGLADESH

July 2008

APPROVAL

The Project Report "Feature Based Optical Bengali Character Recognition using Multiple Back Propagation Neural Networks" submitted by Kazi Wali Ullah, Roll No. 1404 and Fariha Nazmul, Roll No. 1436, Session 2005-2006, to the Department of Computer Science and Engineering, University of Dhaka, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science (Hons.) in Computer Science and Engineering and approved as to its style and content.

(Supervisor)

Syed Monowar Hossain

Lecturer

Department of Computer Science

and Engineering

University of Dhaka

ACKNOWLEDGEMENTS

This project has been supervised by Syed Monowar Hossain, Lecturer, Department of Computer Science and Engineering, University of Dhaka. We are very much grateful and indebted to him for his kind suggestion, guidance, instruction and overall supervision that he offered us during our project investigation. We also thank him for reviewing the preliminary versions of this project and making useful corrections as well as providing valuable suggestions.

We would also like to express our gratitude to our parents and teachers for bringing us up where we are today. We are thankful to the Department of Computer Science & Engineering, University of Dhaka, for providing us with an excellent educational environment and computing facility. We also offer our best regards to Dr. Md. Haider Ali, Chairman, Dept. of Computer Science & Engineering, University of Dhaka, and all other esteemed teachers of the department for their affectionate feelings and encouragement throughout the period of our research work.

Finally, Warm thanks to all of our well wishers and friends for moral support and inspiration.

ABSTRACT

Optical character recognition (OCR), an important area in pattern recognition and image processing, refers to a process whereby printed documents are transformed into ASCII files for the purpose of compact storage, editing, fast retrieval, and other file manipulations through the use of a computer. The recognition stage of an OCR process is made difficult by added noise, image distortion, and the various character typefaces, sizes, and fonts that a document may have. Research in the field of character recognition of Bangla (Bengali) script faces major problems mainly related to the unique characteristics of the script like connectivity of characters on the headline, a large number of compound characters and two or more characters in a word having intersecting minimum bounding rectangles. This project has been an attempt towards the development of a full phased OCR for Bangla alphanumeric characters.

A method for character recognition, invariant under rotation and scaling, using Artificial Neural Network (ANN) classifier is presented here. The method is tested to classify characters from documents of standard Bangla font. The method in this system consists of two parts. The first is a preprocessor and the second is an ANN classifier. In the first part images digitized by flatbed scanner are subjected to skew correction, line, word, character and sub-character segmentation and feature extraction. A set of very simple and easy to compute features is used to classify all the Bangla characters into some smaller character sets. Preprocessing like thinning and skeletonization is not necessary in our scheme. The second part of the method is an ANN classifier, which is trained by the back propagation algorithm. Outputs of the preprocessor are fed into this classifier. The classifier is expected to classify the input character patterns correctly. The recognition system presented here operates at sub-character level and combines them to form Bangla characters.

Performance analysis of the model is presented which shows satisfactory accuracy. The performance also depends on the number of neurons in the hidden layer. So, the effect of the structure of the Neural Network classifier on the performance is also studied.

TABLE OF CONTENTS

Chapter 3 Literature Review

Chapter 4 Proposed Method

Chapter 5 Design and Implementation of the OCR

Chapter 6 Experimental Results

Chapter 7 Discussion

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction to OCR

Pattern recognition is a process which associates a symbolic meaning with objects drawn on an image. In character recognition, the objects represent letters, symbols, or numbers.

This chapter is an overview of what is meant by character recognition and optical character recognition and of the applications in which this process in involved.

1.1 Character Recognition

Any symbol, digit, letter, or punctuation mark stored or processed by computing equipment is known as a character. Character recognition is the technology of using machines to automatically identify human-readable symbols, most often alphanumeric characters, and then to express their identities in machine-readable codes. This operation of transforming numbers and letters into a form directly suitable for electronic data processing is an important method of introducing information into computing systems. In short, character recognition is the ability of a machine to read legible text.

1.1.1 Historical Background

The roots of the character recognition technology go back a long way [1]. In 1870, Carey (Boston) patented an image transmission system using a mosaic of photocells (the retina scanner). In 1890, Nipkow (Poland) invented a device where an image was scanned line by line (sequential scanner). Those systems were not intended to perform recognition but rather transmission of the images. However the idea of transforming the data into a more suitable form for processing was underlying.

The first trace we can find of a true reader, that is a machine that converts printed characters into code, is in the telegraph applications. In 1912, Goldberg patented a machine to read characters and convert them into the telegraph code. It was then possible to send messages without human intervention [1]. However, we need to wait the late 40's to observe the real beginning of the development of characters recognition research and technology.

The original development of computers was for military and scientific purposes, where (relatively) few data have to be entered while a considerable amount of computations had to be done. On the other hand, in business applications, the amount of data to enter is fairly large while the amount of computations to be performed is low. The following problem arose: how should the data is

interchanged between a human and a computer? Even today, the mostly used method is the direct keyboard entry by an operator. This process is very slow, and the required human intervention can (and most of the time does) introduce errors and slows the process of data acquisition. Since working with computers, a natural solution would be to let the computer do the job for us: the machine will transform the original document into a more suitable form and process it. The presence of a human operator would only be necessary if the system has problems with recognition, for correction purposes.

1.1.2 Different Families of Character Recognition

The different areas covered under the general term "character recognition" are either the on-line or the off-line CR, each having its dedicated hardware and recognition methods (see Figure 1.1).

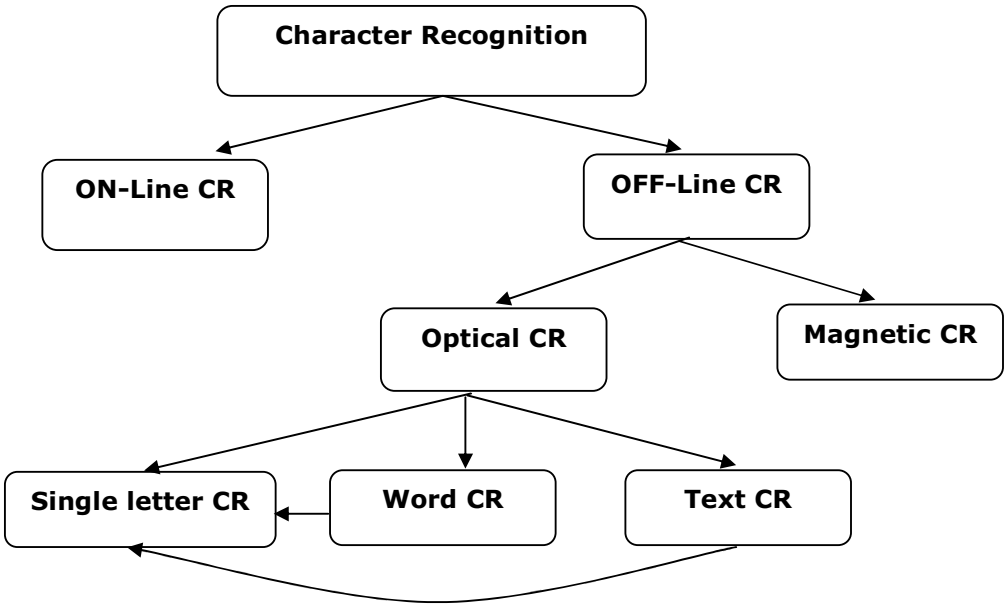


Figure 1.1: The different families in Character Recognition

a. On-line Character Recognition

In on-line character recognition applications, the computer recognizes the symbols as they are drawn (see [2] for some references). The typical hardware for data acquisition is the digitizing tablet, which can be electromagnetic, electrostatic, pressure sensitive, and so on; a light pen can also be used. As the character is drawn, the successive positions of the pen are memorized (the usual sampling frequencies lay between 100Hz and 200Hz) and are used by the recognition algorithm. The interested reader may refer to [3] for references.

b. Offline Character Recognition

Off-line character recognition is performed after the writing or printing is completed. Two families are usually distinguished: magnetic and optical character recognition.

- In magnetic character recognition (MCR) the characters are printed with magnetic ink and are designed to modify in a unique way a magnetic field created by the acquisition device. MCR is mostly used in banking applications, as for example checks reading, because overwriting or overprinting these characters does not affect the accuracy of the recognition.
- Optical character recognition deals with recognition of characters acquired by optical means, typically a scanner or a camera. The symbols can be separated from each other or belong to structures like words, paragraphs, figures, etc. They can be printed or handwritten, of any size, shape, or orientation [4].

1.2 Optical Character Recognition

Optical character recognition (OCR), which is the field investigated in this document, is an important area in pattern recognition and image processing and refers to a process whereby printed documents are transformed into ASCII files for the purpose of compact storage, editing, fast retrieval, and other file

manipulations through the use of a computer. The recognition stage of an OCR process is made difficult by added noise, image distortion, and the various character typefaces, sizes, and fonts that a document may have.

For optical character recognition, a computer must be compensated with an image capturing device (e.g. a camera or a scanner) and special software which is able to recognize text in an image. Such images are captured through optical means as almost all kinds of image capturing devices rely on their optical lenses or sensors. The job of the special software then is to recognize characters from images taken with optical devices. Hence this software is called the optical character recognition software.

1.3 Applications of OCR

Optical Character Recognition (OCR) is a tool. The number of applications where optical character recognition is involved is very large. They can, however, be roughly separated in tree classes:

Letter readers: Only separated characters have to be recognized. For example form reading engines read symbols that have been written in separated boxes.

Word readers: The input document is composed of only few words, as for example addresses on envelopes in mail sorting applications.

Text readers: The application attempts to read and rebuild entire documents. It first divides the page into its components (paragraphs, header, footer, figures...). Each written component is divided into words and finally into single characters which are sent to the OCR engine. The input documents can be printed (articles, books), handwritten, or both (facsimile). The recognized characters can be used for example to reconstruct the input document for further processing with a text processing application. A system for helping blind people has been proposed in the late sixties; it forms phonemes with the recognized characters, and sends them to an audio device.

At the lower level of each of these applications we find a single character recognition engine. However, if the number of words is limited (mail sorters) and/or if separation of words in single characters is difficult (the characters overlap or are written cursively), an application may attempt to perform recognition of groups of symbols rather than single letters. This method is however limited by the exponentially growing number of classes.

At the present time, optical scanning is the easiest way to convert text from paper to electronic form - it is fast and gives a reasonable good result if the paper originals are good. The gain in time depends of course on the quality of the OCR, a lot of errors demand a lot of post processing which is very time consuming. If more than 5% of the words are incorrectly recognized (which corresponds to about 1% of the characters incorrectly recognized), the gain in time is getting very small compared to a good typist.

Optical scanning is still very useful in linguistic research, books, newspaper material and other texts can be made searchable without hours of typing. In many cases the fact that texts become searchable is so important that this compensates for the errors. Some of the most widely uses of OCR are mentioned here [5]:

- Office Automation: A necessary step of office automation is to convert a huge amount of paper-based document into digital form. This can be done by employing human data entry operators or a mixture of a human and an OCR software. The second approach reduces time and cost by a significant margin.
- Form Reading: People require filling up and submitting forms in almost every point of their day-to-day life. After submission, these forms are read by personnel who generally input the information given in the forms into a computer system. This step can be eliminated by using an OCR to read the relevant portion of the form.

- Digital Library: Digitization of books guarantees preservation, easier distribution and reduced storage space. Although data entry operators can be employed to digitize the books, use of an OCR always speeds up this process by a large factor. The most prominent digital library of today's world is perhaps run by the Gutenberg project [6], which is a repertory of mostly copyright expired books.

- Card Reader: Cards are used today in a wide range of applications. From business cards to identity cards, cards have become a very convenient medium for carrying and distributing small amounts of information. Use of OCR for card reading in one of the earliest commercial applications of OCR.

- Mail Sorting: Mail sorting is the process of separating mails according to their destination addresses. In most countries, this is done by human sorters. Sorting machines read the destination addresses using an embedded OCR and sort mail like human sorters.

1.4 Current State of Bengali OCR Research

Bengali OCR research is going on for nearly two decades. As a result, a good number of research publications on the subject have come out from time to time. These research works can be divided into three main categories:

- Recognition of isolate characters in both printed and handwritten form irrespective of fonts.

- Recognition of font dependent printed text.

- Recognition of font independent printed text.

Mahmud [7] has given a concise description of Bengali OCR research scenario till the year 2002:

"In the past few years, many researchers have worked on recognition of both handwritten and printed characters. Bengali character recognition is a special interest for the researchers of this area and researchers of the sub continent

extensively worked on this field. A. K. Roy and B. Chatterjee [8] proposed a nearest neighbor classifier. Abhijit Dutta and Santanu Chaudhuri [9] suggested a curvature based feature extraction strategy for printed Bengali characters. B. B. Chaudhuri and U. Pal [10] combined primitive analysis with template matching to detect compound Bengali characters. Most of the works on Bengali character are recognition of isolated characters. Very few deal with a complete OCR for printed document in Bengali."

In 2003, Atunu Chakraborty published his PhD thesis [11] on a Bengali OCR from Stanford University. His work highlighted some new findings which turned out to be extremely valuable in our experiment. This work too was confined to font-dependent optical character recognition.

So far only a few quality research works have been carried out on font independent optical character recognition involving something more than isolate characters. Mahmud et al [7] is an exception.

To date, BOCRA (Bangla Optical Character Recognition Application) by Deepayan Sarkar [12] is the only open-source Bengali OCR software that has appeared in the Internet. BOCRA is a trainable font dependent OCR. "Chitrankon", an OCR for the Devnagari script released by C-DAC India [13] promises to support Bengali, but that promise has so far remained unfulfilled.

Off late, a team of BRAC University led by Mumit Khan is working to develop a font independent Bengali OCR under a grant of IDRC. China Post (postal dept. of China) has been contracted to develop a mail sorting machine for the Postal Department of Bangladesh. Zhou et al have already published [14] on English/Bengali script identification as part of this project.

As can be evident from the above description, Bengali optical character recognition research is increasingly approaching a point where development of full fledged OCR software is very much a matter of reality.

1.5 Goals

Research in the field of character recognition of Bangla (Bengali) script faces major problems mainly related to the unique characteristics of the script like connectivity of characters on the headline, a large number of compound characters and two or more characters in a word having intersecting minimum bounding rectangles. This project has been an attempt towards the development of a full phased OCR for Bangla alphanumeric characters.

A method for character recognition, invariant under rotation and scaling, using Artificial Neural Network (ANN) classifier is presented here. The method is tested to classify characters from documents of standard Bangla (Bengali) font. The method in this system consists of two parts. The first is a preprocessor and the second is an ANN classifier. In the first part images digitized by flatbed scanner are subjected to skew correction, line, word, character and sub-character segmentation and feature extraction. A set of very simple and easy to compute features is used to classify all the Bangla characters into some smaller character sets. Preprocessing like thinning and skeletonization is not necessary in our scheme. The second part of the method is an ANN classifier, which is trained by the back propagation algorithm. Outputs of the preprocessor are fed into this classifier. The classifier is expected to classify the input character patterns correctly. The recognition system presented here operates at sub-character level and combines them to form Bangla characters.

Performance analysis of the model is presented which shows satisfactory accuracy. The performance also depends on the number of neurons in the hidden layer. So, the effect of the structure of the Neural Network classifier on the performance is also studied.

1.6 Report Organization

The remaining part of this report is divided into six chapters.

To represent each phoneme of a language, a symbol is used. These symbols are called letters. The Bengali language has a set of 50 letters which is called the Bengali alphabet. Chapter 2 is an overview of the alphabet set and the word construct of Bengali script.

In chapter 3, a literature survey in the area of optical character recognition is given. Here the research works on different steps of the OCR process, namely: preprocessing, feature extraction, and classification are presented. Classification process uses a neural network. Detailed description of the neural network model and learning algorithm is also provided in this chapter.

Chapter 4 describes the problems associated with the design of an Bengali OCR system and the methods that are used in this project to alleviate these problems to some extent. This chapter clarifies different difficulties that were encountered during the whole procedural advancement.

The design and implementation of the OCR system & the Graphical User Interface are presented in details in chapter 5. The implementation of the System Modules is written using MATLAB, the language of technical computing [version 7.5.0.342 (R2007b)].

The simulation results and performance analysis of the proposed model are given in chapter 6. Performance analysis is based on the accuracy and execution time of the OCR system.

Conclusions of the obtained results are presented in this chapter 7. It addresses some areas of future works, too.

Chapter 2

Bengali Script

To represent each phoneme of a language, a symbol is used. These symbols are called letters. The Bengali language has a set of 50 letters which is called the Bengali alphabet.

This chapter is an overview of the alphabet set and the word construct of Bengali script.

2.1 Bengali Alphabet

When the speech sound of any language is analyzed, what is found as a result are the phonemes of that language. To represent each such phoneme, a symbol is used. These symbols are called letters. For example মা is a Bengali word. By analyzing মা, two phonemes are found, namely ম and আ. So the word মা is written using the symbols of the ম and আ phonemes.

After detailed analysis of the Bengali language, linguists have selected a total of 50 letters. The set of all these 50 is called the Bengali alphabet.

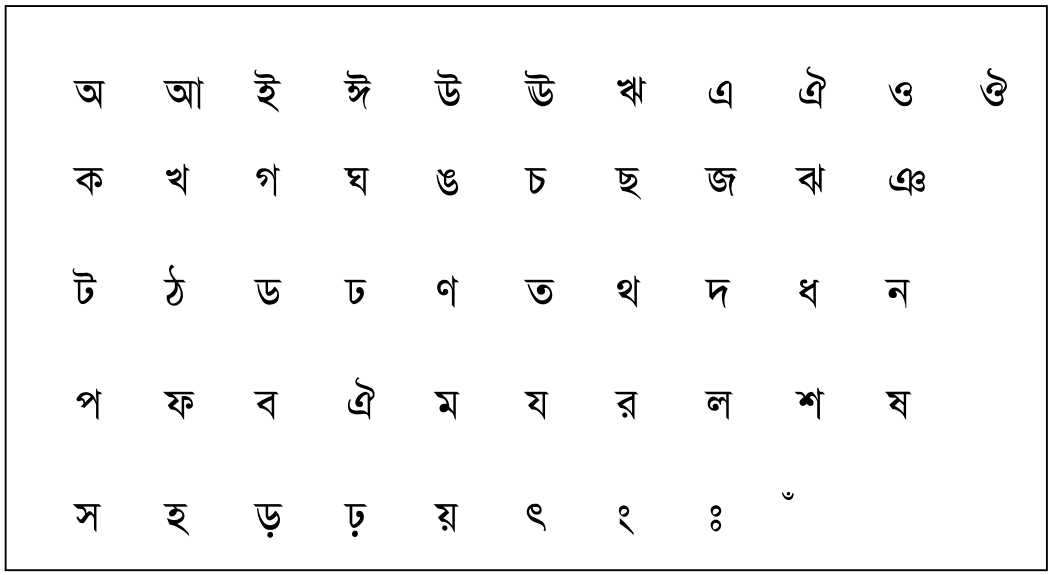


Figure 2.1: Bengali Alphabet

Bengali phonemes have been grouped into two classes:

Vowels: Vowels sounds do not get obstructed anywhere when emerging from one's mouth-hole. Symbols used to denote such sounds are called vowel letters. Examples of Bengali vowel letters are অ, আ, ই, etc.

Consonants: Those sounds that get obstructed while coming out of the mouth-hole are called consonant. Representative symbols of such sounds have been named consonant letters. Examples of Bengali consonant letters include ক, খ, গ, ঘ, ঙ, etc.

For the purpose of clearly uttering Bengali consonants, an implicit অ is added with each one of them. When any consonant is written without অ, a hashanta symbol is appended with the letter.

2.1.1 Matra

A horizontal line '—' is written above many letters of the Bengali alphabet. This line is called 'Matra'. For example, letters like ক, ন, etc have the 'Matra' symbol above them. On the other hand, some letters have no 'Matra' at all. These include ঙ, এ, etc. There is a third group also, comprising ণ, ধ, etc which possesses half-'Matra'.

2.1.2 Kar

When a vowel letter is combined with a consonant letter, the shape of the vowel letter changes to a different form which is a somewhat smaller symbol than the vowel. These short symbols are called 'Kar'. Every kar is named according to its corresponding vowel letters.

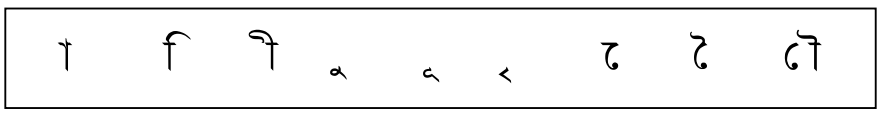


Figure 2.2: 'Kar' symbols in Bengali Script

2.1.3 Consonant Clusters

Consonant clusters (or ligatures, or compound characters) are a very important feature of the Indic languages. Consonant clusters are formed when two consonants are added together such that there is no implicit অ between them. Consonant clusters may have similarity in look with their parent consonants or it can be that they look completely different. There is no consensus on how many consonant clusters actually exists. This has made the recognition of all consonant clusters a difficult issue. According to the Bengali Academy Bengali-to-Bengali dictionary, there are 216 consonant clusters in Bengali. It has also listed two blocks of consonant clusters implicitly. This puts the total number of consonant clusters above 275. The implicit blocks are –

ক+র, খ+র, গ+র .....

ক+য, খ+য, গ+য .....

Besides, the problem of consonant cluster recognition has been worsened by the use of multiple symbols for representing the same consonant cluster. The following figure shows some consonant clusters:

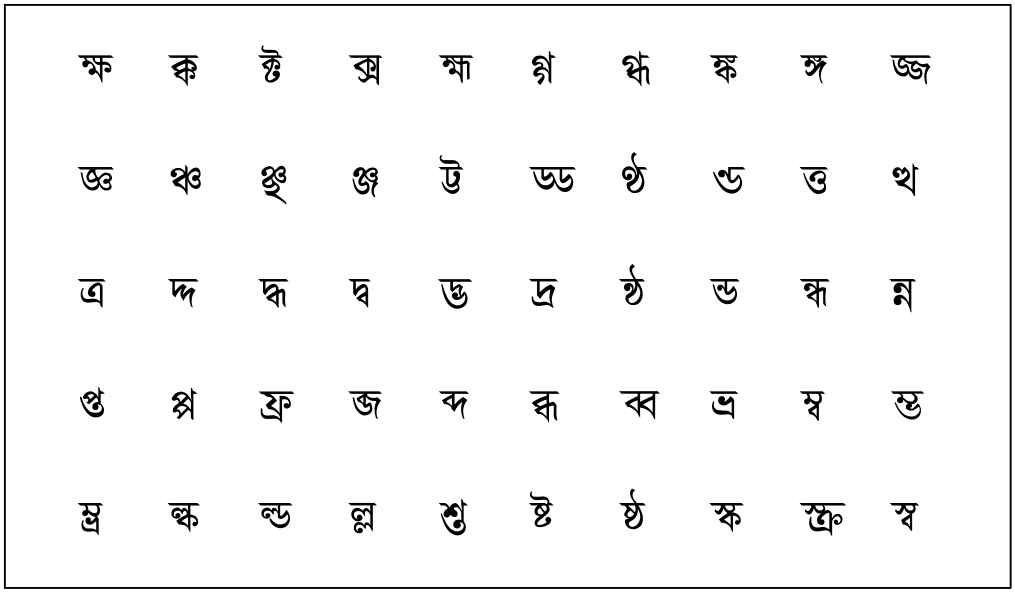


Figure 2.3: Some of the consonant clusters

2.2 Word Structure

Before plunging into the details of the Bengali word structure, a Context Free Grammar (CFG) notation is presented below [5]:

Bengali-WORD	→	INITIAL REST
INITIAL	→	VOWEL| SYLLABLE
REST	→	SYLLABLE REST| SYLLABLE
SYLLABLE	→	KAR1 C | CKAR2 |
		C | BROKEN-KAR-CONSONANT
BROKEN-KAR-CONSONANT	→	O-KAR-BEGINS C O-KAR-ENDS
		| AU-KAR-BEGINS C AU-KAR-ENDS
C	→	CONSONANT| CONSONANT-CLUSTER
VOWEL	→	অ, আ, ই, ঈ, উ, ঊ, ঋ, এ, ঐ, ও, ঔ
CONSONANT	→	ক, খ, গ, ঘ, ...,ষ, স, হ, ড়, ঢ়,য়, ৎ, ং, ঃ, ঁ
KAR1	→	ে , ৈ , ি
KAR2	→	া , ী , ু , ূ , ৃ
O-KAR-BEGINS	→	ে
O-KAR-ENDS	→	া
AU-KAR-BEGINS	→	ে
AU-KAR-ENDS	→	ৗ
CONSONANT CLUSTERS	→	ক্ষ, ক্ক, ষ্ট, ...

Figure 2.4: CFG of Bengali Word Structure

It should be mentioned here that some kar (◌া , ◌ি , ◌ে) are written below the consonants and consonant clusters. These cannot be properly fitted into a CFG. So the CFG given in the above figure shows them as if they are written next to consonants or consonant clusters.

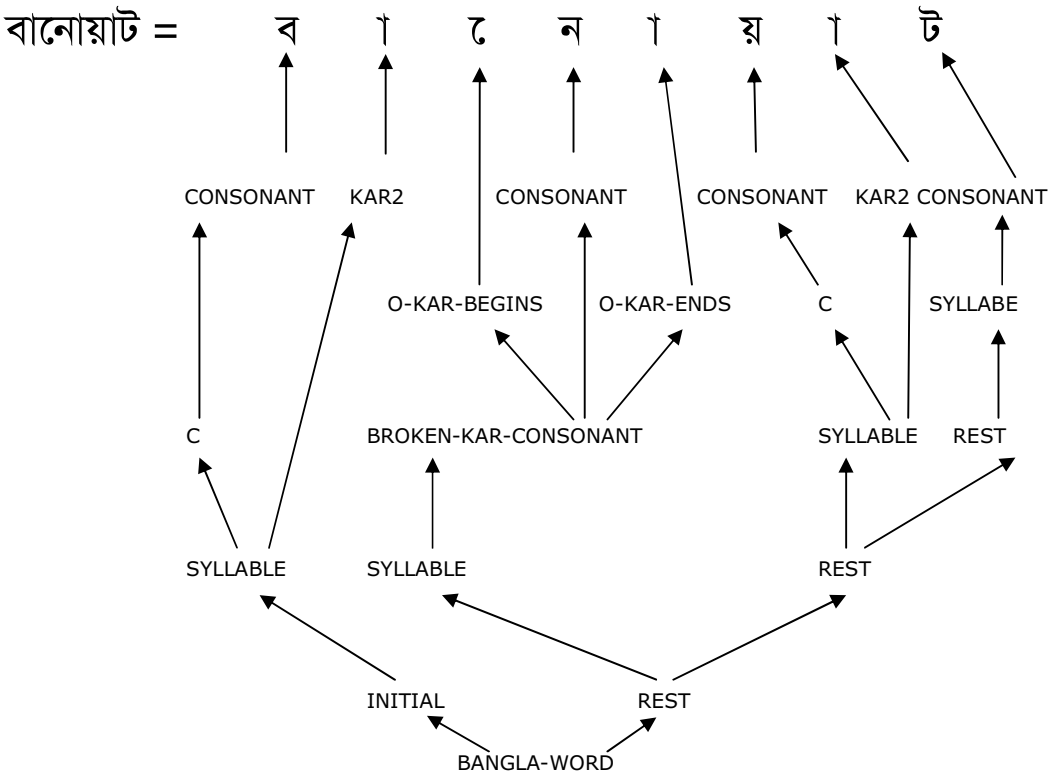


Figure 2.5: CFG of a Bengali Word

The main properties of Bengali words are as follows :

- Letters are written from left to right.
- Kar is not written as a separate letter. Instead it can be below, partially above or to the left or right of a consonant or consonant cluster.

- 2 kars, o-kar and au-kar, are split into two parts and a consonant or consonant cluster is horizontally wrapped with the parts.

About the connection between letters and characters, Wikipedia [15] says the following:

"In computer and machine-based telecommunication terminology, a character is a unit of information that roughly corresponds to a grapheme, or symbol, in the written form of a natural language. An example of a character is a letter, numeral, or punctuation mark."

So the term "character" and not "letter" will be used in the next chapters which are mostly concerned with the computing aspect of OCR.

Chapter 3

Literature Review

In this chapter the different steps of the OCR process, namely: preprocessing, feature extraction, and classification are described.

Classification process uses a neural network. Detailed description of the neural network model and learning algorithm is provided.

In this chapter, a literature survey in the area of optical character recognition is given. Here the research works on different steps of the OCR process, namely: preprocessing, segmentation, feature extraction, and classification are presented in detail. The necessity of preprocessing, the methods for preprocessing, different levels of segmentation process, statistical, structural and some other features and some post processing methods are discussed. The classification process uses a neural network. Detailed description of the neural network model and learning algorithm is also provided in this chapter.

3.1 Steps in an OCR Process

There are two steps in building a classifier (see *figure 3.1*):

a. Training
b. Recognition

These steps can be broken down further into sub-steps.

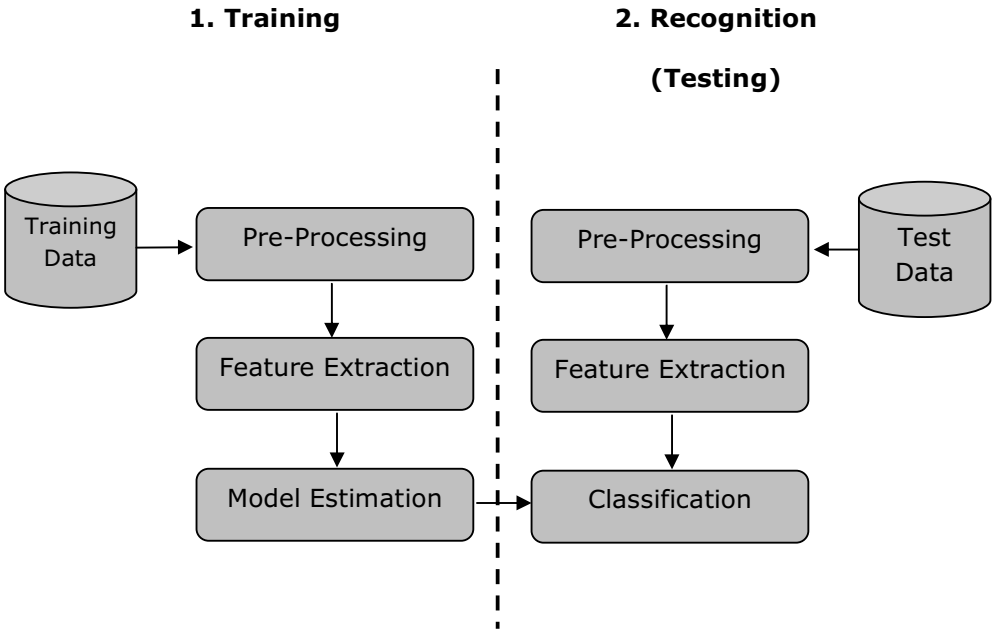


Figure 3.1: The Pattern classification process

1. Training

a. *Pre-processing* – Processes the data so that it is in a suitable form for the classifier.
b. *Feature extraction* – Reduce the amount of data by extracting relevant information—usually results in a vector of scalar values. (For distance measurements the features need to NORMALIZED).
c. *Model Estimation* – from the finite set of feature vectors, need to estimate a model (usually statistical) for each class of the training data.

2. Testing

a. *Pre-processing*-(same as above)
b. *Feature extraction* – (same as above)
c. *Classification*

Both of the above steps for an OCR usually do the following tasks:

- Preprocessing
  - Scanning
  - Error correction
  - Binarization
  - Skew correction
- Classification
  - Line segmentation
  - Word segmentation
  - Character segmentation
  - Character recognition

Finally, the recognition step may involve the following task in order to improve the correctness of the recognition process:

- € Post-processing
    - Dictionary matching

3.2 OCR Preprocessing

If the quality of the original image is low and/or if the resolution of the scanner isn't sufficient, the digital image of the document may need to be preprocessed. At this stage of the application, three actions are usually performed: gap filling, noise cleaning, and thresholding.

Gap filling and noise cleaning

For some reason, there can be instances when a character had not been scanned correctly and a gap cuts it into two parts. The segmentation algorithms will very likely extract each part as separate characters, resulting at the end of the recognition process in a misclassification. Such images may below-pass altered in order to close the gaps and clean out the noise. For this purpose, median filters are often applied, but other filters and techniques (like the morphological operation "closing") can be used as well [16].

Thresholding

The output of an optical scanner is a color, a graylevel, or a binary image while the segmentation is usually performed on binary data. Color images can first be transformed into gray level rasters. A graylevel raster can be binarized using a thresholding algorithm which sets the value of a pixel to `0' or `1', for example, depending on how it compares to a threshold value. Many algorithms for automatic thresholding have been described along with adaptive methods which chose threshold value locally, with respect to the current pixel or region in the image.

3.2.1 Need for Preprocessing

After preprocessing (of the whole image) and segmentation, one may believe that the patterns have to be recognized are, in a way, "well prepared". However several problems may arise. Differences in position, size, and orientation of the pattern on the bitmap, as well as presence of noise, may be able to result in different features for characters belonging to the same class.

Translation. The patterns are not always located at the same position on the bitmap. This problem is usually caused by the segmentation step in applications where the characters are required to be written in boxes (forms, zip code on envelopes...). The segmentation algorithm detects only the boxes and extracts what is inside. An algorithm based on contour detection will not produce these errors. In order to avoid the translation problem, the enclosing rectangles of patterns can be detected and, for example, regionprops function of the Matlab image processing toolbox gives the smallest bounding box of each image segment.

Scaling. The size of the patterns and/or of the bitmaps is not always the same. This problem happens almost always for handwritten characters and is due to the use of different fonts sizes for printed characters. Bitmaps of different widths and heights must all be resized to the same dimension, also if the widths and heights are less or greater than predefined threshold values we ignore them (these are considered as noise) or try to split them (these are considered as images of two characters touching).

Rotation. The orientation of the patterns is not always the same. This can be due to: the orientation of the original document during scanning, the orientation of the symbols on the original document, for example maps where the names of the objects (rivers,roads,etc) are often written in the same direction as the objects themselves, the orientation of handwritten characters depends on the author, etc. If the neural network is trained with images with slight rotations, it should be able

to classify such images correctly. But, image rotation may require a lot of CPU resources.

Noise. Although noise is supposed to have been cleaned during the first preprocessing stage, it may still be present. It is essentially due to the quality of the original document, quantization errors during scanning, and the binarization algorithm. Noise cleaning bitmaps is usually performed by simple algorithms. If noise is present over the entire bitmap it can be cleaned by detecting the contour of the biggest connected set of pixels (which will be assumed to be the pattern) and unsetting all the pixels outside this contour. Other possibilities are to use majority white or majority black filters, or morphologic operations, such as opening, closing, dilation, erosion, smoothing, etc. [17].

Because of all the problems listed above, two patterns P1 and P2, known to belong to the same class may be incorrectly classified, because the features extracted from P1 and P2 differ importantly. The need of normalization (correction) is obvious. Normalization can be performed either in the pattern space (the pattern will be altered), in the feature space, or both. The preprocessing stage aims at performing two tasks: bitmap error correction (normalization) and preparation of the bitmap if a special feature extraction scheme is used. Many references on bitmap preprocessing can be found in [17].

3.2.2 Morphological Operation

Morphology is a technique of image processing based on shapes. The value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image.

There exist many morphological functions which can be used to perform common image processing tasks, such as contrast enhancement, noise removal, thinning, skeletonization, filling, and segmentation.

Dilation and Erosion

Dilation and erosion are two fundamental morphological operations. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion. This table 3.1 lists the rules for both dilation and erosion. [18]

Table 3.1- Rules for Grayscale Dilation and Erosion

Operation	**Rules**
Dilation	The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

The following figure 3.2 illustrates the dilation of a binary image. In the figure, note how the structuring element defines the neighborhood of the pixel of interest, which is circled. (See Structuring Elements for more information.) The dilation function applies the appropriate rule to the pixels in the neighborhood and assigns a value to the corresponding pixel in the output image. In the figure, the morphological dilation function sets the value of the output pixel to 1 because one of the elements in the neighborhood defined by the structuring element is on.

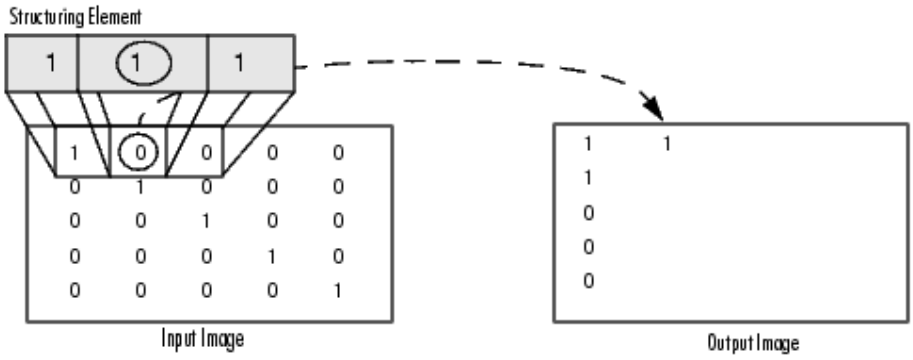


Figure 3.2: Morphological Processing of a Binary Image

Structural Element

An essential part of the dilation and erosion operations is the structuring element used to probe the input image. Two-dimensional, or flat, structuring elements consist of a matrix of 0's and 1's, typically much smaller than the image being processed. The center pixel of the structuring element, called the origin, identifies the pixel of interest--the pixel being processed. The pixels in the structuring element containing 1's define the neighborhood of the structuring element. These pixels are also considered in the dilation or erosion processing. Three dimensional, or nonflat, structuring elements use 0's and 1's to define the extent of the structuring element in the x- and y-plane and add height values to define the third dimension. [18]

Thinning

The thinning process peels off layers of pixels from an image and leaves it in a single pixel wide format. According to Hilditch [19], a thinning algorithm should satisfy the following conditions:

- Each thinned stroke must finally be exactly one pixel wide.
- The one pixel wide result must run through the center of the original stroke.
- It should not completely thin away some part of the stroke in a way that changes connectivity.
- No stroke should be shortened by the thinning process.

3.3 Segmentation

Segmentation first separates a document into its components as for example the header, the footer, the paragraphs, the figures, the images, the mathematical formulae, etc. Russell & Norvig [20] describes image segmentation as:

"Segmentation is the process of breaking an image into groups, based on similarities of the pixels. The basic idea is the following: Each image pixel can be associated with certain visual properties, such as brightness, color, and texture. Within an object, or within a single part of an object, these attributes vary relatively little, whereas across an inter-object boundary there is typically a large change in one or the other of these attributes. We need to find a partition of the image into a set of pixels such that these constraints are satisfied as well as possible."

Because the gray level images contain more information than the binary ones, this can be done before the thresholding stage. The components can be roughly classified as being: purely written (printed or hand written), purely drawn (figures or images), or mixed. The second segmentation step will extract the symbols from the first and third category.

If the characters are well separated, their extraction is quick and easy to implement. One can first compute the boundaries of the pattern, or its enclosing rectangle and then extract the pixels which lay inside. In other cases (i.e. the characters touch or overlap each others, are drawn on figures, are cursively written, etc.) the extraction can be difficult, very difficult, or even impossible. In these cases, specialized methods have to be used for segmentation, feature extraction, and classification.

3.3.1 Line Segmentation

A page full of text contains many lines. Generally there can be 30-40 lines in a page full of Bangla characters. The line segmentation process identifies line boundaries and thus separates one line from another for further processing. The

ultimate goal of the segmentation process is to divide each character and line segmentation is the first step towards that goal.

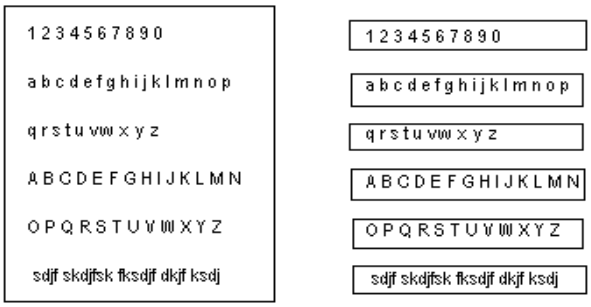


Figure 3.3: Line segmentation

The text lines are segmented by finding the valleys of the histogram computed by row wise sum of gray values**.** A threshold t_i has been chosen at the training phase by observing a large number of images. The position where the histogram height is greater than t_i denotes the on-set of lines. Thus, a text line can be found between two consecutive positions marked by the threshold. [21]

3.3.2 Zone Segmentation

A text line in Bengali script can be partitioned into three horizontal zones. The upper zone denotes the region above the head line, where vowels reside, while the middle zone represents the area below the head line where the consonants, compound characters and some sub-parts of vowels are present. The middle zone is the busiest zone. The lower zone represents the area below middle zone where some of the vowel modifiers and certain half characters lie in the foot of consonants.

From the word, row histogram is constructed by counting frequency of each row in the word. The row with the highest frequency value indicates that head line which has been denoted as 'Matra'. Some times there are consecutive two or more rows with almost same frequency value. In that case, 'Matra' row is not a single row. [22]

The detection of zones is based on the position of 'Matra' and line separation. The upper zone is the zone above the 'Matra'. For detection of lower boundary of middle zone an imaginary horizontal line in the middle of the text line is considered. Now from the last row of the text line column scans are made upwards. For each character or its isolated parts the lowest black pixels is noted. The row which contains maximum number of such pixels denotes the lower bound of the middle region. [21]

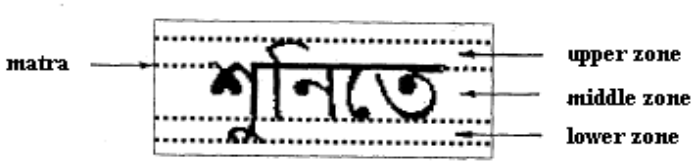


Figure 3.4 : Zone segmentation

3.3.3 Word Segmentation

Word segmentation is the third step of the whole segmentation process. Here the input is an image of a line of text and the outputs are the image of all the words contained in that line. Since words are the basic elements that represent an entity, it's important to identify and extract each word separately. Generally the wider spaces in the line image are used as word boundaries.

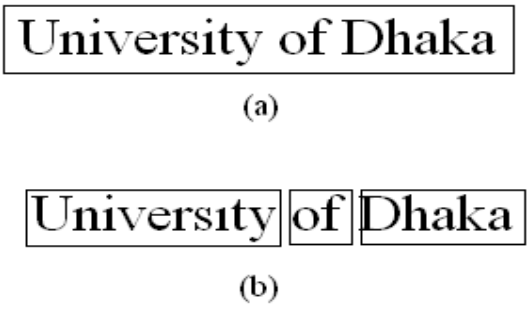


Figure 3.5: (a) Line image (b) Segmented word images

A segmented text line is scanned vertically. If in one vertical scan two or less black pixels are encountered then the scan is denoted by 0, else the scan is denoted by 1. Thus, for each line a sequence of 0 and 1 is generated as follows

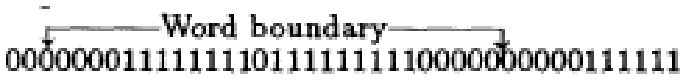


Figure 3.6: Word boundary

Now if a run of 0's is longer than a pre-specified integer threshold then the middle point of that run of 0's is considered as the boundary of a word. The integer threshold is a function of line width. [21]

3.3.4 Character Segmentation

Character segmentation is the process of separating every character image from the given word image. Character segmentation carries lot significance as its output is the single unit of data which is used in the recognition process. Unlike most of the previously described tasks, there is no hard and fast rule on how to do character segmentation given a word image. The wide variation among different scripts and even the variation among the characters of the same script have made the formation of a unified approach to character segmentation almost impossible. As a result, different algorithms have been devised for different scripts.

The Latin script can be taken as an example – in printed texts of Latin scripts, each character maintains a small gap with the previous or next character. In case of a connection between two adjacent characters, it is through a small stroke at the lower part of both the characters. These features are utilized to separate one printed Latin character from another. But this technique proves lacking when handwritten text of Latin script is considered. So it is evident that a single algorithm is not enough even for segmenting both printed and handwritten characters of the same script. This calls for a good amount of research when character segmentation algorithms are developed for a new script. As for the Bangla script, character segmentation has turned out to be the most daunting task in the whole character recognition process.

Figure 3.7: Segmented character images

For segmentation of characters the middle region is considered. Segmentation is done by deleting the 'Matra' region from the middle region. If the 'Matra' region is deleted form middle region then the characters in a word get topologically disconnected. [21]

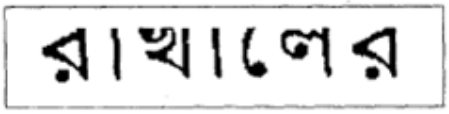


Figure 3.8: A 'Matra' deleted text

Another way to find the demarcation line between characters is to initiate a vertical scan from the row that is just beneath the 'Matra' row. If the scan can reach the bottom of the word then it has successfully found the demarcation line between characters. But only linear vertical scan fails to find the demarcation line for some words. In this case, the vertical scan is not linear only. It is piecewise linear in the sense that the scan takes turns whenever it sees an obstacle (i.e. black pixel) and tries to reach the bottom of the word. The following figure shows the approach.

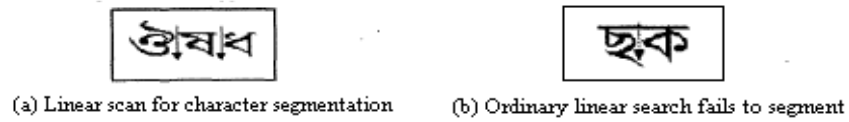


Figure 3.9: Ordinary linear scan for character segmentation

To find the portion of any character above the 'Matra' a check is made whether it is possible to move upward from the 'Matra' row from a point- just adjacent to

the ‘Matra’ row and between the two demarcation lines. If it is, then a Greedy search is initiated from that point and the whole character is found.

Figure 3.10: Piecewise linear scan for character segmentation

To segment the characters below another character, base line of the word image has been calculated. Each word can be considered to have an imaginary line that crosses at the middle of the word. If a greedy search is initiated from the word image that searches for presence of black pixels below the imaginary line then the search will result some connected components below that imaginary line. All those components contain a lowest point which is called 'Base Point’. Base line is the highest frequency row of those points. After determining the base line, a depth first search easily extracts the characters below base line. [22]

A character is assigned to one of the three groups namely basic, modifier and compound character group. The basic, modifiers and compound character subgroups is done using the properties: (i) bounding box width, ti) number of border pixels per unit width, and (iii) accumulated curvature over the border per unit width. [23]

Once a character is decided as compound character it is subject to another tree classifier. At the first level of this tree classifier, the compound characters are grouped according to the character width. At the next level the presence or absence of vertical line is tested. The next level feature is ‘Matra’, the presence or absence of which puts the character in one or other subgroup. For final classification, a template matching approach [24] can be used on each subgroup. The template matching approach is made efficient by two heuristics. Let the test character X belong to a subgroup of n characters. The ideal templates of these characters are stored in the computer with information about the character widths. The second heuristics relate to the stroke feature common to the subgroup. [21]

Sub images marked for further segmentation due to its width may contain either a conjunct or a shadow character pair. An outer boundary traversal is initiated from the top-left most black pixel of the image. During the traversal, the right most pixel visited will not be on the right boundary of the image box if the image box contains a shadow character pair. If the image contains a conjunct, the right most pixel visited will be on the right boundary of the image box. A conjunct is segmented vertically at an appropriate column to yield constituent characters. [25]

3.4 Classification

Classification may be seen as a function C from the feature space F to the space *C(C: F->C)* where C is the set of all the possible classes. If F has a big dimension, and this will be the case if bitmaps are used as features, the application of C may require a lot of CPU time. By extracting features from the pattern the dimensionality of F is decreased, and therefore the classification is speeded up. The second important reason is that normalization is faster and more accurate in feature space than it is in pattern space.

Once the feature extraction scheme and the dimension of the feature vector (say N) are chosen, a question may be asked: can a lower number of features (say M), be used without increasing the misclassification rate? The only guaranteed technique for choosing the best subset of M features from a set of N features is to try all the ${}^{N}C_{M}$ possible combinations. This is computationally impractical for sets of even moderate size, so heuristic techniques are required. In [26], Mucciardi et al. present a comparative study of such methods.

Features are grouped in a feature vector which are compared to the feature vectors of known patterns during classification. Features can be either numbers or more complicated structures. In the first case classification is based on statistics or neural nets. In the latter case, a decision tree is usually used.

Classifier using statistical features

From a finite set of feature vectors, a model (usually statistical) is estimated for each class of the training data. Feature vectors of the test data is compared to the various models and using the distance measure the closest match is found.

Classifier using Neural Network

Alternatives to the classifier described above are the artificial neural networks models or simply neural nets. Artificial neural net structure is based on present understanding of biological nervous systems, which is a dense interconnection of simple computational elements. Although these elements, also called nodes, are nonlinear, typically analogue and slow compared to modern digital circuitry, good performance is achieved because of the massively parallel structure.

Neural nets can perform as various tasks as: Identification of the class which represents best an input pattern (classical classification), associative memory where only a part of a pattern is available and the complete pattern is required, vector quantization or clustering. Such computations can, however, not be performed by single nodes or neurons. Nodes are therefore cascaded, forming multi-layer networks.

When applied to OCR, neural nets can be used in two ways:

- The inputs of the net are directly connected to the pixels of the image (binary or gray level).
- More classically, the feature vector is sent on the input of the net and feature vector classification is then performed.

3.5 Post-processing

During post-processing two actions are typically performed: detection/correction of errors and reconstruction of the original document.

Dictionary based methods have proven to be most effective for error detection and correction. Once all the characters of a word have been recognized, the word is looked up in a dictionary. If it is present, we can assume the characters are correct (although this is not absolutely sure), if not, an error has been detected. The word can be changed to the most similar word in the dictionary. If there are more than one candidate, the system has to ask the user unless it has some higher level knowledge about the language it has to recognize: its grammatical structure, for example. The main disadvantage with this method is that searches and comparisons are time consuming and increase with the size of the dictionary. Some mail sorters use this method since the number of words, i.e. names of the geographical sites, is reasonable. Correction methods based on n-grams (pairs, triplets, etc. of letters) use the knowledge about structural information of the words, namely the probabilities of the n-grams.

Reconstruction depends on what the application is supposed to do. In signature verification and authentication applications, the action can be as simple as "Accept'' or "Reject'' the check. On the other side of the complexity scale, a document can be really reconstructed from the knowledge of recognized characters, positions of paragraphs, figures, etc. (known from the segmentation stage).

Many references on post-processing can be found in [2].

3.6 Feature Extraction

Features are characteristics that describe the pattern and are extracted from the bitmap. Features can be roughly classified by their nature in four groups:

Bitmap itself: This group is apart from the others since the features are the bitmaps themselves.

Distributions: The features are statistical measures of distributions of points on the bitmap, the contour curve, the profiles, or the HV-projections. Widely described methods are: zoning, n-tuples, characteristic loci, and moments.

Series expansion coefficients: Images or curves can be expressed as infinite series (Fourier, Walsh, Cosinus, etc) and approximated by retaining only the first K terms of the sums. The coefficients or more often, combinations of the coefficients of these terms are used as features.

Structural features: The patterns are analyzed in terms of their structure: number of strokes, their relative orientations and positions, number of loops, etc. These information are used as features and classified using a special classification method.

Features in the three first groups are also called by the general denomination global features [18].

3.6.1 Methods of Moments

The first significant study of two-dimensional moments used as features for pattern recognition is due to Hu in 1962.

The two dimensional ($p+q$) th, $p,q \geq 0$, order moment (usually called *conventional moment*) of a density distribution function $f(x,y)$ is defined in terms of Riemann integrals as:

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y) dx dy$$

If we assume that $f(x,y)$ is piecewise continuous and has non zero values in a finite region of the plane, then the moments of all orders exist and the following uniqueness theorem can be proved:

Uniqueness Theorem: The sequence $\{m_{pq}\}$, $p, q \geq 0$, is unequally determined by $f(x,y)$ and conversely, $f(x,y)$ is unequally determined by $\{m_{pq}\}$.

This theorem is very valuable to pattern recognition since it insures that an image (assumed to be a bounded piecewise continuous function) may be assigned a unique set of features, and therefore characterized by them.

Properties of lower order conventional moments

The conventional raw moments of an image *f* of size *WxH* may be computed in the discrete plan by:

$$m_{pq} = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} x^p y^q f(x,y)$$

The complete set $\{m_{pq}\}$ of order n is composed of moments m_{pq}, $p + q \leq n$ and its cardinality is equal to: $1/2(n + 1).(n + 2)$. Conventional and central moments (which will be defined below) can be used to compute simple mechanical and statistical values that may be used as features.

Zero order moment The zero th order conventional moment m_{00} represents the mass of *f* and is proportional to the average intensity of *f*. If *f* is binary ($f(x, y) = 1$ or 0), m_{00} is the area of the object drawn on the image.

First order moments m_{10} and m_{01} are used to locate the centroid Ω(center of mass) of the object. The coordinates $\bar{x}$ and $\bar{y}$ of Ω are defined by:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

Note that if the origin of the coordinate system is moved to Ω, the moments extracted from the image are called *central moments,* designated as μ_{pq} and computed by:

$$\mu_{pq} = \sum \sum (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

Second order moments: The second order conventional and central moments are used to compute some mechanical features:

Eccentricity: - The eccentricity of the ellipse that has the same second-moments as the region. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases; an ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)

Orientation: - The angle (in degrees) between the x-axis and the major axis of the ellipse that has the same second-moments as the region. This property is supported only for 2-D input label matrices.

Third and fourth moments of the HV-projections Features may be extracted from the horizontal and vertical projections of the images on the *Oy* and *Ox* axes respectively. In the continuous domain, the projections *h(y)* and *v(x)* are defined by:

$$h_y = \int_{-\infty}^{+\infty} f(x,y)dx \quad \text{and} \quad v_x = \int_{-\infty}^{+\infty} f(x,y)dy$$

One-dimensional moments from the functions *h* and *v are:*

$$m_p = \int_{-\infty}^{+\infty} x^p v(x)dx = \int_{-\infty}^{+\infty} x^p \left[\int_{-\infty}^{+\infty} f(x,y)dy \right] dx = m_{p0}$$

and

$$m_q = \int_{-\infty}^{+\infty} y^p h(y)dy = m_{0q}$$

Identical relations hold for the central moments.

If we assume that the functions *h* and *v* are probability distributions, classical statistical measures can be extracted as measures:

- The third order central moments μ_{30} and μ_{03} define the *skewness* S_x and S_y of the image projections, and measure the degree of deviation from the symmetry about the mean of the distribution. S_x and S_y are defined by:

$$S_x = \mu_{30} / \mu_{20}^{3/2} \quad \text{and} \quad S_y = \mu_{03} / \mu_{02}^{3/2}$$

- The fourth order central moments μ_{40} and μ_{04} define the *kurtosis* of the distributions *h* and *v* and measures the "peakness": K = 0 for a Gaussian distribution, K < 0 for a flatter distribution and K > 0 for a more peaked one. The kurtosis K_x and K_y of *v(x)* and *h(y)* respectively, are defined by:

$$K_x = \mu_{40} / \mu_{20}^2 - 3 \quad \text{and} \quad K_y = \mu_{04} / \mu_{02}^2 - 3$$

In 1988, Al-Yousefi and Udpa presented an OCR system based on feature extraction (moments) from the HV-projections of the image [4]. The good results they report motivated to implement their method.

3.6.2 Statistical Features

According to G S Lehal and Chandan Singh [27] the statistical classifying features for Gurumukhi characters are:

Number of junctions with the headline equals 1: Each sub-symbol merges with the headline at one or more than one point.

Number of endpoints and their location: A black pixel is considered to be an end point if there is only one black pixel in its 3 x 3 neighborhood. The number of endpoints as well as their positions in terms of 9(3*3) quadrants are considered.

Number of junctions and their location: A black pixel is considered to be a junction if there are more than two black pixels in its 3 x 3 neighborhood. The number of junctions as well as their positions in terms of 9(3*3) quadrants are considered.

Horizontal Projection Count: Horizontal Projection Count represented as HPC(i) = ∑F(i, j), where F(ij) is a pixel value (0 for background and 1 for foreground) of a document image, and i and j denote vertical and horizontal coordinates of the pixel respectively, when the image's top left corner is set to F(0,0).

Right Profile depth: The maximum depth of the right projection profile of sub-symbol image is stored as percentage with respect to total width of the box enclosing the sub-symbol image.

Left Profile Depth: The maximum depth of the upper half and lower half of the left projection profile is stored as percentage with respect to total width of the box enclosing the sub-symbol image.

Right and Left Profile Direction Code: The profiles are scanned from top to bottom and local directions of the profile at each pixel are noted. Starting from current pixel, the pixel distance of the next pixel in left, downward or right direction is noted. The cumulative count of movement in the three directions is represented by the percentage occurrences with respect to the total number of pixel movement and stored as a 3 component vector with the three components representing the distance covered in left, downward and right directions respectively.

Aspect Ratio: Aspect ratio is obtained by dividing the sub-symbol height by its width.

Another statistical feature described by Veena Bansal [25] is:

Horizontal Zero Crossings: Image of a character is treated as an array of pixels. Tracing the whole array row by row, number of transitions from black pixel to white pixel is recorded for each row. Let V, be the number of transitions for ith row. Let this sequence be S. The sequence S is divided into three sub-sequences

of equal length, S1, S2, and S3. For each subsequence the most frequent number is stored in Sl-Most, S2-Most, SS-Most. Further, the sequence S is analyzed and the most frequent number in the sequence is stored as S-Most. The feature vector consists of (S-Most, SlMost, S2-Most, S3Most).

Some other simple but effective statistical features are:

Horizontal Ratio: Ratio of the pixel sum of the upper and lower strip in the image

Vertical Ratio: Ratio of the pixel sum of the left and right strip in the image

3.6.3 Structural Features

Structural features aim at capturing the essential shape information of the characters, generally from their skeletons, sometimes from their contours. The features include: loops, junction, crossing and end points, concavities and convexities, arcs and strokes. A symbol may then be described by these features and the topological relations that exist between them.

Structural features will be naturally classified by a rule based classifier, specially designed for a given set of characters. Rules may be of the form:

IF *P* has (*f1,. . . ,fn*) THEN (…) ELSE (...)

where the actions performed in the "THEN" or "ELSE" parts can be further tests on the features, extraction of more specific features, or assignment of the pattern to a given class. A description and comparison of four systems based on structural features can be found in [28].

A stroke is described in terms of its length, height and expected angle. Several structural features called strokes for middle zone are shown in the following figure. These features are detected without doing any preprocessing like thinning or skeletonization. [21]

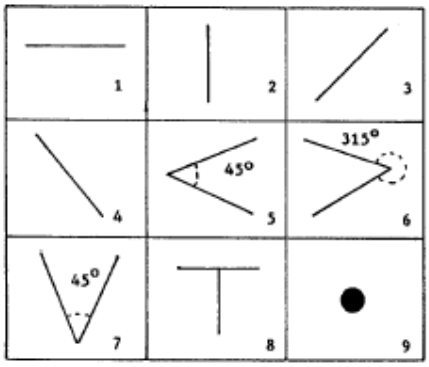


Figure 3.11: Stroke features used for recognition

To detect the presence of the stroke no 1, of Fig, it is assumed that the length of the stroke l_a, is at least 75%of the width of the particular character. Now the upper quarter part of the character middle zone is scanned horizontally The method of detection of the stroke no 2 is similar to that of stroke no 1, but here length of the stroke l_b is 75% of the height of the character middle zone and the scanning mode is vertical.

To detect the stroke no 5, the length of the arms of the stroke is assumed to be 30% of the width of the middle zone of the character and the angle between them is nearly 45°. A two way scan is performed where starting point of the scan is from the left middle part of the middle zone of the character. If in a scan each arm contains continuous black pixels of length greater than or equal to the length of the stroke then the stroke is assumed present in that character. The detection method of the stroke no 6 is similar to that of stroke no 5, only the angle between the arms is 315°.

The detection method of the stroke no 8 is the combination of method of detection of stroke no 1 and that of stroke no 2. Here at first it is checked whether the stroke no 1 is present or not. If this stroke is present then a stroke like stroke no 2, whose length is 40% of the height of the middle zone, is tested at the middle portion of stroke no 1. If such a stroke is also present then it is assumed that stroke no 8 is present in that character.

For detection of feature no 9, a black circle of diameter equal to the thickness of 'Matra' is checked. If such a circle is detected and if it is not contained in another stroke of the character then the existence of feature no. 9 is deemed.

A structural feature for printed hindi text is:

Vertical Bar Feature: The characters are divided into three groups based on the presence and position of vertical. Some of the characters belonging to each of these classes are shown in the following figure.

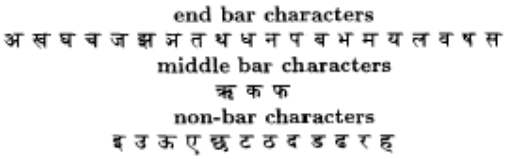


Figure 3.12: Three classes of characters in Hindi based on vertical bar feature

The character image is divided in three equal vertical zone and vertical projection for each zone is computed. The column that contains desired number of pixels corresponds to the vertical bar column. [25]

Some other structural features are [27]:

Presence of a loop: The presence of a loop in the sub-symbol is another important classification feature. This feature false for the loop formed along the headline.

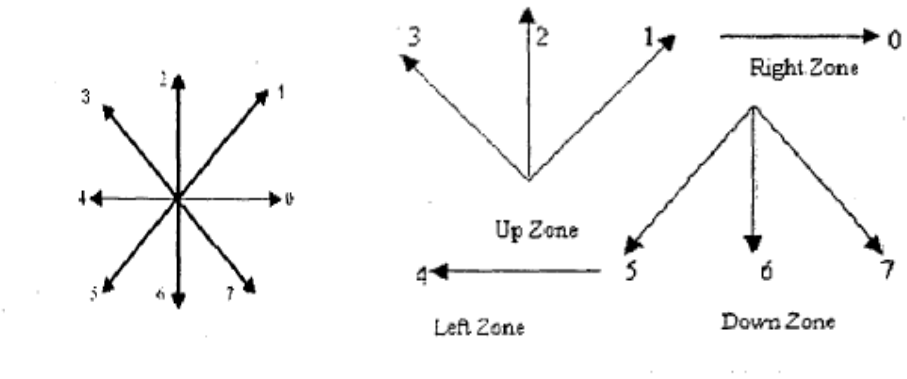
Loop along the headline: This feature is true if the sub-symbol forms a loop with the headline.

Euler Number: Euler Number is the total number of objects in the image minus the total number of holes in those objects. It is a useful feature to check whether a character has a loop or not.

3.6.4 Other Features

According to Mahmud [22] some other classifying features are:

Chain Code Generation: After the character has been divided into connected components and boundary of the connected components are established, chain code is to be calculated. There are several chain code convention 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) *(b)*

Figure 3.13: (a) Slope Convention for Freeman Chain Code
(b) Direction zones for searching

The 8 transitional positions defined by freeman chain code are then divided into 4 transitional zones in order to facilitate the searching and to keep the correct order of searching.

Slope Distribution Generation: When searching for a closed contour continues, there is a variation of slope in each region. The frequency of each directional slope at each region is recorded and updated during the traversal. There are eight directional slopes in a region, therefore total 32 directional slopes for the whole component. The frequency of jth directional slope at ith region is local feature **S** i j, where j = 0, 1, ... , *7* and i = 0, 1, 2, 3.

3.6.5 Guidelines to the Selection of Features

Selection of a set of features must not be done randomly: their quality is vital for good behavior of the system. As a conclusion, here are some guidelines:

- Features need to be informative: each feature added to the feature vector must bring new information.
- The number of features should be kept as low as possible.
- Features must be chosen with respect to the characters to be recognized.
- Features need to be robust or invariant to transformations as translation or scaling, as well as to noise, in order to avoid as much as possible the preprocessing stage. A special attention should be given to orientation invariance in order to decide whether it is desirable or not (discrimination between `W' and `M', for example).
- Computation of the features must be as fast as possible.

As a consequence, it may be of interest to build multi-layer systems, based of several sets of features, rather than monolithic ones, based on a unique set.

3.7 Neurocomputing : Basics of Neural Networks

Neurocomputing is the technological discipline concerned with parallel, distributed, adaptive information processing systems that autonomously develop operational capabilities in response to exposure to an information environment [29]. The primary information processing structures of interest in neurocomputing is neural networks. This is the first alternative to programmed computing. Solving a problem using programmed computing involves devising an algorithm and/or a set of rules for solving the problem and then correctly coding these in software and also making necessary revisions and improvements.

Clearly, programmed computing can be used in those cases where the processing to be accomplished can be described in terms of a known procedure or a known set of rules. If the required algorithmic procedure and/or the set of rules are not known, then they must be developed and this has found to be costly and time consuming. In fact, if the algorithm is not simple, the development process may

have to await a flash of insight. Obviously, such an innovation process cannot be accurately planned or controlled. Even when the required algorithm or rule set can be devised, the problem of software development must be faced. Because current computers operate on a totally logical basis, software must be virtually perfect if it is to work. The exhaustive design, testing, iterative improvement that software development demands makes it a lengthy and expensive process.

A new approach to information processing that does not require algorithm or rule development and that often significantly reduces the quantity of the software that must be developed has recently been available. This approach, called neurocomputing, allows, for types of problems (typically in areas such as sensor processing, pattern recognition, data analysis and control), the development of information processing capabilities for which the algorithms or rules are not known (or where they might be known, but the software to implement them would be too expensive, time consuming, or inconvenient to develop). For those information processing operations amenable to neurocomputing implementation, the software that must be developed is typically straight forward operations such as data file input and output, peripheral device interface, preprocessing and post-processing. The Computer Aided Software Engineering (CASE) tools often used with neurocomputing systems can frequently be utilized to build these routine software modules in a few hours. These properties make neurocomputing an interesting alternative to programmed computing, at least in those areas where it is applicable.

3.8 Introduction to Neural Networks

The term neural network originally referred to a network of interconnected neurons. The average human brain consists of 1.5 X 10^{10} neurons of various types, with each neuron receiving as many as 10^4 synapses. With this kind of complexity, it's no wonder that the human brain can be considered as the most complex piece of biological machinery on earth.

Today the term neural network has come to mean any computing architecture that consists of massively parallel interconnections of simple "neural" processors.

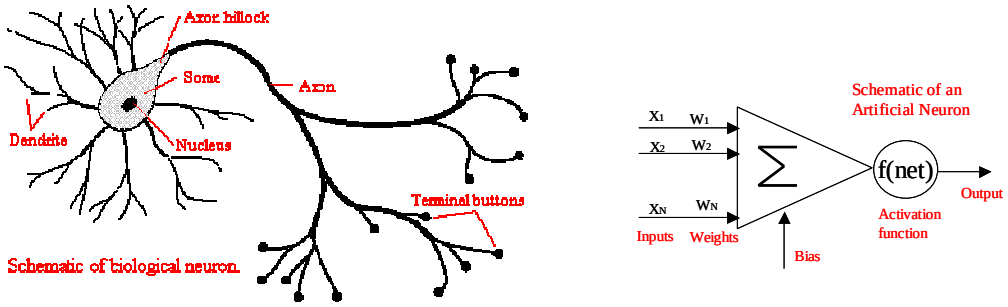


Figure 3.14: The Biological Neurons and the Artificial Neural Model

Neural networks are designed to work with patterns. They can be pattern classifiers wherein they can take a vector (series of numbers), then classify the vector. For example, an OCR (Optical Character Recognition) program takes an image of a character and output the character itself. Neural networks can also be used in military radars where radar returns can be classified as enemy vehicles or trees. Neural networks can also be pattern associators wherein they take one vector and output another. For example, a program can be written that takes a 'dirty' image and outputs the image that represents the one closest to the one it has learnt. Associative networks can be used in more complex applications such as signature, face, and fingerprint recognition.

3.8.1 Definitions of Neural Networks

There are many possible definitions of the neural networks. This is because the concept of the neural network is still vague. The formulation is changing rapidly and there seems to be little hope of standardization in the near future. Thus, any definition for neural network must be given tentatively. Each definition will attempt to capture as much of the flavor of our neural networks as possible.

1. A neural network is a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore, each element operates asynchronously; thus there is no overall system clock [30].

2. Neural networks are mathematical models that abstract parallel instruction handling features of biological systems made up of many relatively simple elements called neurons [31].

3. A neural network is a parallel, distributed information processing structure consisting of processing elements (which can posses a local memory and carry out localized information processing operations) interconnected via unidirectional or bi-directional signal channels called connections. Each processing element has a single output connection that branches ('fans out') into as many collateral connections as desired; each carries the same signal – the processing element output signal. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it must be completely local; that is, it must depend only on the current values of the input signals arriving at the input processing element via impinging connections and on values stored in the processing elements local memory [29].

3.8.2 Characteristics of Neural Network

Neural networks exhibit the following characteristics:

- A large number of very simple neuron like processing elements.
- A large number of weighted connections between elements and the weights on the connections encode the knowledge of the network.
- Highly parallel, distributed control.
- Each processing element operates on purely local information.
- Each of the processing element operates asynchronously.

3.8.3 Desirable Properties of Neural Network

A neural network should be able to:

- Self-organized using supervised/unsupervised learning.
- Form stable category codes.
- Operate under the presence of noise.
- Operate in real time.
- Perform fast and slow learning.
- Scale well to large problem.
- Perform context-sensitive recognition.
- Process multiple patterns simultaneously.
- Modify categories when necessary.

3.8.4 Single and Multi-layer Perceptrons

A perceptron is a simple neural network model introduced by Frank RosenBlatt in 1958, and is perhaps the most widely used term in neural networks. A single layer perceptron is used to classify an input vector into several classes. In a single layer perceptron, the input values and activation level of the perceptron are either –1 or 1; weights are real-valued (between 0 and 1). The activation level is given by summing the weighted input values $\Sigma \mathbf{x_i w_i}$. Perceptrons use a simple hard-limiting threshold function, where activation above a threshold results in an output value of 1, and –1 otherwise.

$$\text{Perceptron output} = \mathbf{sign}(\Sigma \mathbf{x_i w_i})$$
$$= \mathbf{1} \quad \text{if} \quad \Sigma \mathbf{x_i w_i} >= \mathbf{t}$$
$$= \mathbf{-1} \quad \text{if} \quad \Sigma \mathbf{x_i w_i} <= \mathbf{t}$$

The perceptron uses a simple form of supervised learning. The way a perceptron learns to distinguish patterns is through modifying its weights to reduce error.

The adjustment for the weight Δw_i on the i^{th} component of the input vector is given by:

$$\Delta w_i = c\, x_i\, \delta$$

where c = learning rate

d = desired output

δ = (desired output) – (actual output) = $d - \mathbf{sign}(\Sigma x_i w_i)$

Single layer perceptrons can only solve problems where the solutions can be divided by a line (or hyperplane). The classes to be distinguished should be *linearly separable*. Therefore, a single layer perceptron cannot express non-linear decisions like the XOR problem.

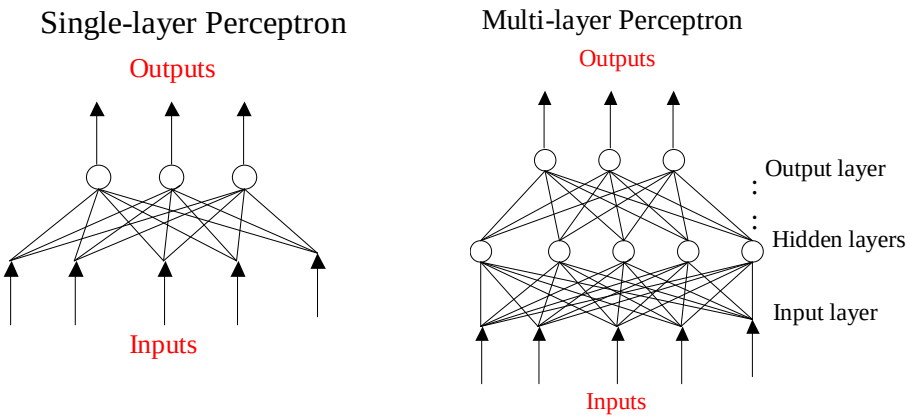


Figure 3.15: Different Perceptron Networks

Multi-layer perceptrons are feed-forward nets with one or more layers of nodes between the input and output nodes. These additional layers contain hidden units or nodes that are not directly connected to both the inputs and outputs. Multi-layer perceptrons overcome many of the limitations of the single layer perceptrons. The capabilities of multi-layer perceptrons stem from the nonlinearities used within nodes. In multi-layer networks, when adjusting a weight anywhere in the network, one has to be able to tell what effect this will have on the overall effect of the network. To do this, one has to look at the derivative of the error function with respect to that weight. The hard-limiter function for the single-layer perceptron is non-continuous, thus non-differentiable. The most popular continuous activation function used within

backpropagation nets is the sigmoid function or the logistic function given by the equation:

$$\mathbf{f(net) = 1 / (1 + e^{-\lambda * net}), \text{ where net} = \Sigma x_i w_i}$$

The shape of the sigmoid function is shown in the figure 2.4. As λ (called the squashing parameter) gets large, the sigmoid function approaches a linear threshold function over {0, 1}; as it gets closer to 1, it approaches a straight line. This activation function is non-linear, scaled and differentiable.

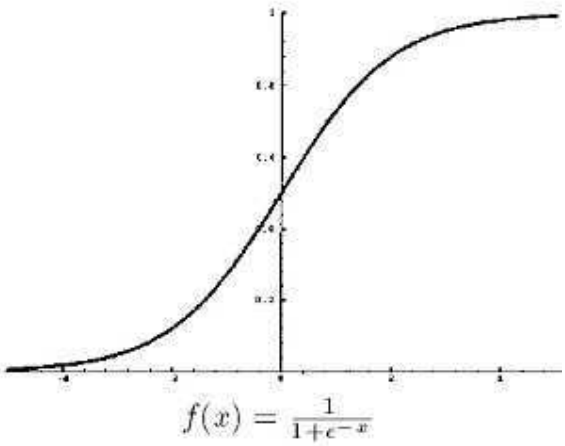


Figure 3.16: Sigmoid Function.

3.8.5 Back-propagation Neural Network

The back-propagation algorithm is perhaps the most widely used supervised training algorithm for multilayered feedforward networks. The back-propagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feedforward perceptron and the desired output. In the back-propagation algorithm, a feedforward phase is first done on an input pattern to calculate the net error. Then, the algorithm uses this computed output error to change the weight values in the backward direction. The error is slowly propagated backwards through the hidden layers - and hence its name.

The actual derivations for the different formulas used in the back-propagation algorithm come from the generalized delta rule. The delta rule is based on the idea of the error surface. The error surface represents cumulative error over a data set as a function of the network weights. Each possible network weight configuration is represented by a point on this error surface. By taking the partial derivative of the network error with respect to each weight we will learn a little about the direction the error of the network is moving. In fact, if we take the negative of this derivative (i.e. the rate change of the error as the value of the weight increases) and then proceed to add it to the weight, the error will decrease until it reaches a local minimum. The taking of these partial derivatives and then applying them to each of the weights, takes place starting from the output layer to hidden layer weights, then, from the hidden layer to input layer weights. For details about the derivation, look in Luger's book [24].

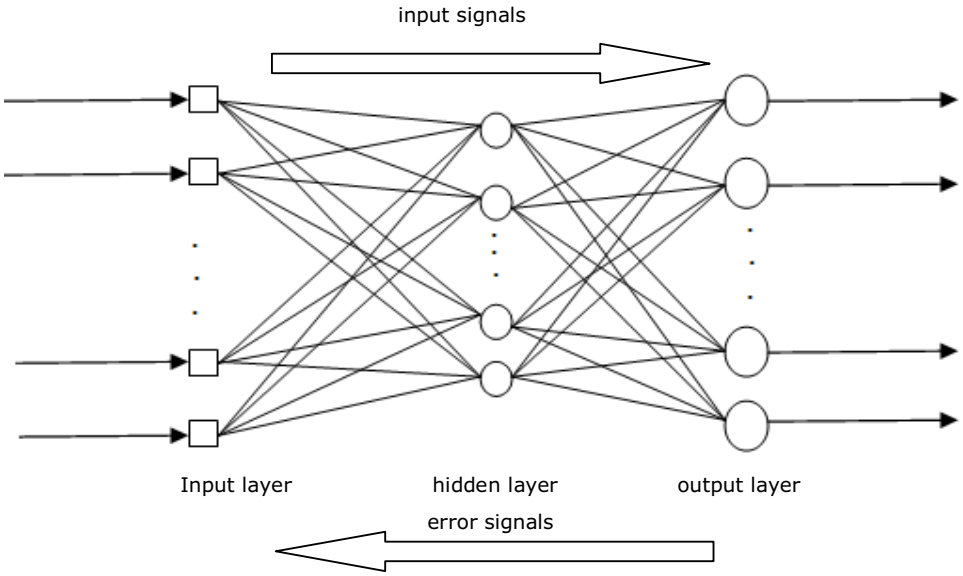


Figure 3.17: Three layer back-propagation neural network

The steps involved in the back-propagation training algorithm are listed below [18]:

Step 1. Initialize weights and offsets: Set all weights and node biases to small random values.

Step 2. Present input and desired outputs: Present a continuous valued input vector $\mathbf{x_0, x_1, \ldots, x_{N-1}}$ and specify the desired outputs $\mathbf{d_0, d_1, \ldots, d_{M-1}}$. If the net is used as a classifier, then all desired outputs are typically set to 0 except for that output corresponding to the class the input is from, which is set to 1. The input could be new on each trial or samples from a training set could be presented cyclically until weights stabilize.

Step 3. Calculate actual outputs: Use the following formulas to calculate outputs $\mathbf{O_1, O_2, \ldots, O_{M-1}}$ of every neuron in the network.

$\mathbf{O_i = f(\Sigma x_i w_i + b_i)}$

$\mathbf{f(y) = 1 / (1 + e^{-\lambda * y})}$

where **x** → input vector, **w** → weight vector denoting to weights linking the neuron unit to the previous neuron layer, **b** → bias vector, λ → squashing parameter.

Step 4. Adapt weights: Use a recursive algorithm starting at the output nodes and working back to the first hidden layer. This has many sub-steps.

Step a: Compute the sum-squared error of the network.

$$\mathbf{Error} = \tfrac{1}{2} \Sigma_{i \, \varepsilon \, outputs} (d_i - O_i)^2$$

Step b: Calculate the error term of each neuron in the output layer, δ_i.

$$\delta_i = O_i (1 - O_i) (d_i - O_i)$$

Step c: Calculate the error term of each neuron in the hidden layer, δ_i.

$$\delta_i = O_i (1 - O_i) \Sigma_j \delta_j w_{ij}$$

where, **j** is the index of the nodes in the next layer to which **i**'s signals fan out.

Step d: Compute the weight deltas. η is the learning rate. A low learning rate can ensure more stable convergence. A high learning rate can speed up convergence in some cases.

$$\Delta w_{ki} = \eta \, \delta_k \, x_{ki}$$

where $\mathbf{w_{ki}}$ is the weight from the hidden (or input) node **k** to node **i**.

Step e: Add the weight deltas to each of the weights

$$\mathbf{w_{ki}(t+1) = w_{ki}(t) + \Delta w_{ki}}$$

where **t** denotes the iteration step.

Step 5. Repeat by going to step 2.

3.8.6 Training the Network

Training basically involves feeding training samples as input vectors through the neural network, calculating the error of the output layer, and then adjusting the weights of the network to minimize the error. Each "training epoch" involves one exposure of the network to a training sample from the training set, and adjustment of each of the weights of the network once layer by layer. Selection of training samples from the training set may be random, or selection can simply involve going through each training sample in order.

Training can stop when the network error dips below a particular error threshold (eg: 0.001 squared error). However, it is found that excessive training can have damaging results in such problems as pattern recognition. The network may become too adapted in learning the samples from the training set, and thus may be unable to accurately classify samples outside of the training set. When this happens we can either include these samples in the training set and retrain, or we can set a more lenient error threshold.

These "outside" samples make up the "validation" set. This is how the network's performance is assessed. Tests must be done to confirm that the network is also capable of classifying samples outside of the training set.

Chapter 4

Proposed Method

This chapter describes the problems associated with the design of an Bengali OCR system and the methods that we have used to alleviate these problems to some extent.

4.1 Why a new method?

OCR literatures are infested with algorithms that try to thin and segment characters before recognizing them. But these techniques are not suitable for Bengali Script due to the complex shape of Bengali characters. The next three sections elaborate these issues.

4.1.1 Problems with Skeletonization of Bengali Character

Generally, character skeletons are created by applying thinning on the character image. But thinning is not suitable for Bengali character images. Chakraborty beautifully describes the limitations of thinning as a skeletonization approach for Bengali character images [11]:

"Our experience in using thinning algorithms suggests that they are very vulnerable to noise, and are often hard to use for OCR purposes. Figure 2.14 shows a few samples of the same letter taken from the same page. It may seem at first sight that the skeletonization is satisfactory. But notice that it is more difficult to recognize the thinned versions than the originals. The first four images are glyphs corresponding to the same letter ক্ষ. But note how different they look after thinning. The thinned version of the last glyph is difficult to recognize even for a human being. In the original images, parts contributed by noise tend to be thinner than the strokes themselves. This allows the human eye to recognize the glyph in spite of the noise. But a thinning algorithm does distinguish between noise and signal. It thins everything to one pixel wide lines, reducing the signal to noise ratio."*

Subramanian & Kubendran [32] describes a skeletonization method that came close to giving a perfect skeleton of Bengali characters. But it too produced incomplete skeletons in many cases.

**See Figure 4.1*

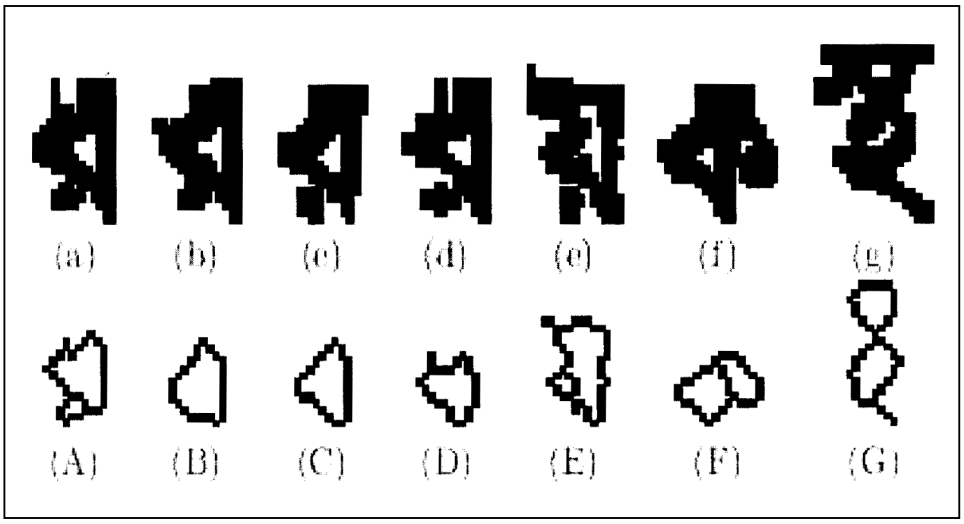


Figure 4.1: Result of applying thinning on various samples
[Figure 2.14 of Chakraborty 2003]

4.1.2 Problems with Segmentation of Bengali Character

Some characteristics of Bengali Script are:

- Characters in Bengali words are written both horizontally and vertically.
- Almost all characters are connected to some other character.

ক + ে + থ + া + প + ক + থ + ন = কথোপকথন

Figure 4.2: Connectedness in Bengali Word

Generally used character segmentation algorithms, which were developed keeping the Latin script under consideration, assume that every character is separated from the surrounding characters by a small space. According to the 2nd point above, Bengali writing system does not fulfill this space requirement. This renders the traditional character segmentation algorithms totally useless in the context of Bengali character segmentation.

To overcome the limitations of the traditional character segmentation algorithms, some researchers like Chakraborty have proposed to remove the 'Matra' from Bengali words [11]:

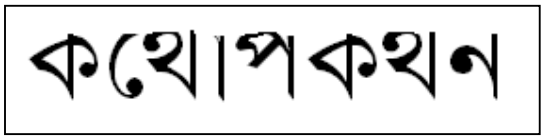


Figure 4.3: Effect of 'Matra' removal

This technique successfully segments Bengali characters when the word structure is relatively simple where characters are placed only horizontally. This type of simple word, however, is not the only kind of word in Bengali. The 1st point above says that Bengali characters can be vertically placed in Bengali words. The 'Matra' removal technique does not address the segmentation problem of such words.

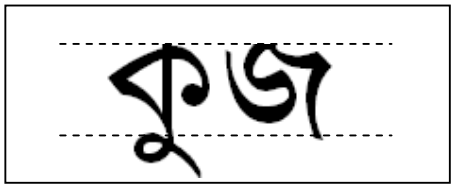


Figure 4.4: Failure of 'Matra' removal to segment every character

For the above mentioned issues, an elegant algorithm for Bengali character segmentation has still remained elusive.

Another problem that arises from 'Matra' deletion is character partition. Some Bengali characters such as, গ, ঞ, ণ are not connected by a 'Matra'. When the 'Matra' of a word is deleted, the characters are partitioned into two disjoint parts. The scenario for Bengali letter গ is depicted in the following figure.

Figure 4.5: 'Matra' removal may lead to partitioning of some characters

Such a partition causes the segmentation process as well as the recognition process to fail for the occurrence of some characters.

4.1.3 Problems with Single ANN in Recognizing Bengali Characters

The basic idea behind character recognition using neural network is to train the neural network using some training patterns/characters, which are typically given as the resized image of the character for which the neural network is to be trained, and then input those patterns to the trained neural network to classify the input pattern/character. This has been the usual method to recognize characters for many languages. But there are several problems with this usual method when it is to be used to recognize all the Bengali characters. These are described in the followings.

1. When there are too many characters to recognize, the neural network becomes unable to detect the characters effectively.

2. Giving only the resized character image as input to the neural network for Bengali character recognition is not a good option. This is because, in Bengali, there are a lot of characters with some of them very similar in shape with each other, which makes it harder to recognize. Again an resized image of a character contains too little information about a character to be recognized correctly by the neural network. Also a little translated image, due to the segmentation process, can cause the neural network to fail to recognize a character.

3. When there is only one neural network module to recognize a character, there is no other way to identify a character when the neural network

module fails. Either the character goes undetected or it gives an erroneous result.

4. When there are too many characters to be recognized and/or if the training pattern is the image of the characters then training the neural network takes a large number of epochs. It may be in the range of several thousands, implying a very slow training phase. Sometimes it may even be possible that a neural network for this sort of patterns can never be trained.

4.2 Architecture of the OCR System

The OCR system consists of two major components.

a. Training Component
b. Recognizer Component.

Three neural network based classifiers are used in the recognition stage. All the classifiers need to be trained from properly labeled input examples. And the combined result of all three classifiers is considered as the correct classification.

4.2.1 Training Component

This component is responsible for the training of the classifiers. The major steps are shown in figure 4.6.

The Training Image: We have trained the classifiers using black and white image consisting of the characters. The characters are then segmented and were put in a file which is then used to train the classifiers. The image should be noise free for proper training.

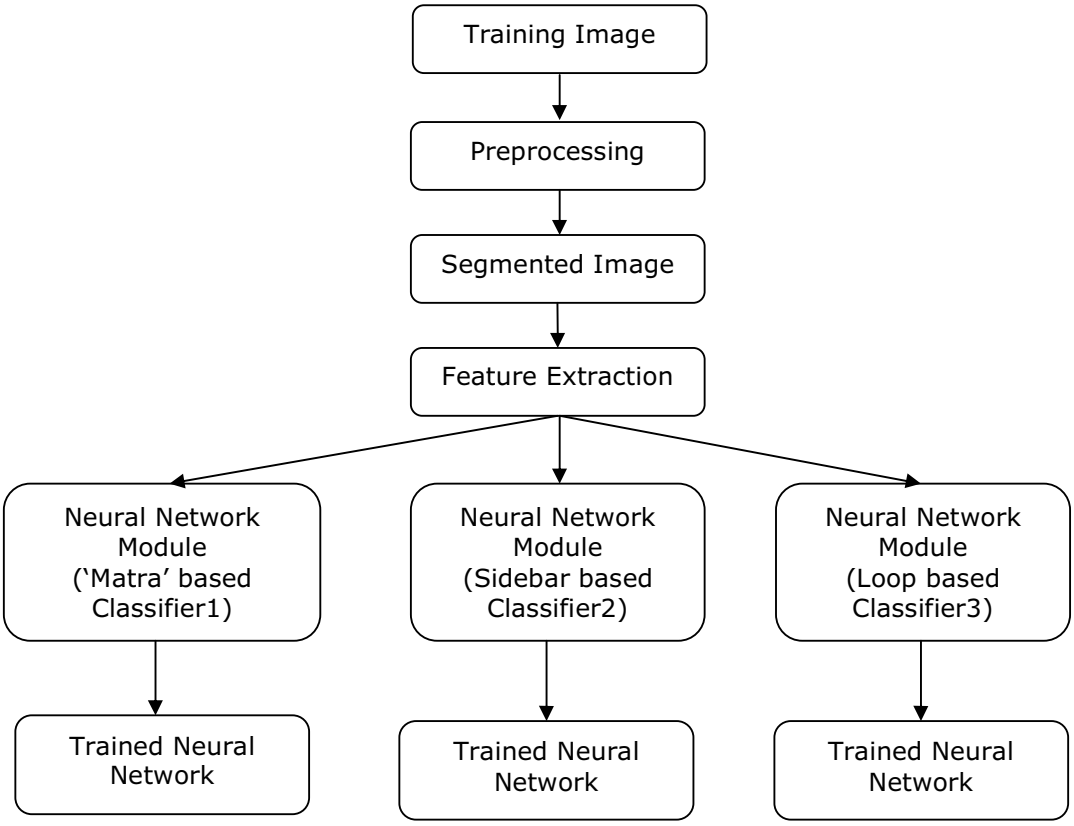


Figure 4.6: The Training Module

Preprocessing: The training image is then segmented and the user is asked to label the images to create the training examples.

Feature Extraction Module: This module extracts the features from the training examples. The features and the extraction methods are described in Section 3.6. The features that are used in this project are listed in Section 4.3.2.

Neural Network Module: The character images are resized to 20 pixels in each dimension and then the extracted features from the resized image are given as the input of the neural networks and then the neural networks are trained using the back propagation algorithm (see Section 4.2.3 for more details).

4.2.2 Recognizer Component

This component is responsible for the testing of the classifiers. The major steps are shown in the following figure:

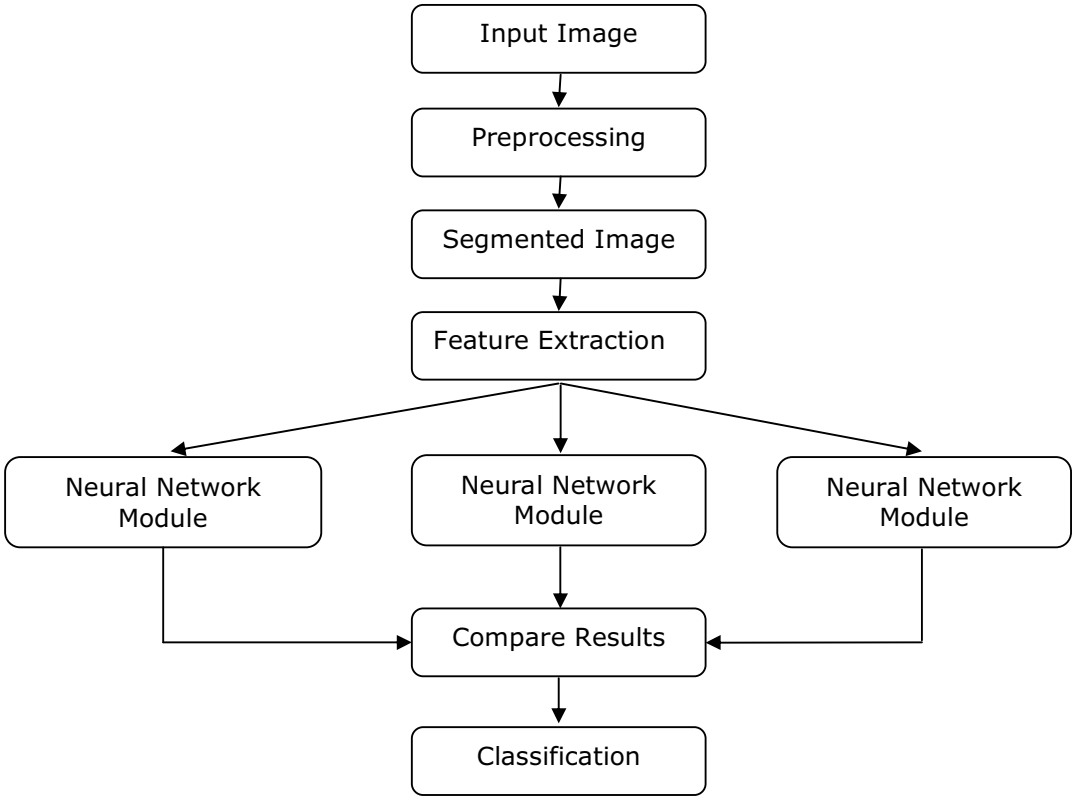


Figure 4.7: The Recognizer Component

The Input Image: The input image may consist of several lines of texts.

Preprocessing: The input image is then segmented and to some extent noise is removed from it. Segments that are assumed to be an image of touching digits are split while segments with height and width below certain threshold values are considered to be noises and eventually ignored.

Classification: From the figure it can be understood that there are two classifiers that perform the recognition in parallel. A module then compares the results. From the experimental results we have found that the neural network, operating

on the binary image directly, performs quite well. To improve the accuracy the results of both the classifiers are combined.

4.3 Our Method

A method for character recognition, invariant under rotation and scaling, using Artificial Neural Network (ANN) classifier is presented here. The method is tested to classify characters from documents of standard Bangla (Bengali) font. The recognition system presented here operates at sub-character level and combines them to form Bangla characters.

The model presented here recognizes the first 25 consonants of Bengali alphabet and some vowel modifiers (Kars) from any text written in standard Bengali font (primarily SutonnyMJ) of same size. It can be easily enhanced to recognize all the characters in Bengali alphabet by training all of them. The limitation of this model is that it is unable to recognize the compound characters.

4.3.1 Character Segmentation

The problems that are associated with character segmentation based on deletion of 'Matra', are alleviated in our method to some extent using the character width to height ratio. It has been observed that the aspect ratio of any Kar falls between the range 0.1 to 0.4 excluding a-kar (া) and any valid character falls in the range of 0.6 to 1.5. For convenience we have included a-kar (া) in the valid character range. Using this ratio, all the connected components in a word are grouped into three classes. These are:

Garbage: The connected component of a word that has either too small width to height ratio or too small height or both are treated as garbage and is eventually bypassed by the recognition process.

Kars: The connected components of a word that has a width to height ratio between 0.1 to 0.4 and has the height as at least 75% of the average height of a character is again grouped into two classes. These are:

- Partial Characters: This class consists of those character parts, which becomes disconnected because of the deletion of 'Matra' or some kind of erosion operation. These partial characters are first checked whether they can be constructed as a whole character otherwise ignored.

- Kars: Here, in this class, falls the 'akars'. It is further checked whether it is an 'akar' or a 'rosho-ikar' or a 'dirgho-ikar' or a 'ou-kar'.

Valid Characters: The connected components of a word that has a width to height ratio of 0.5 and above and has the height as at least 75% of the average height of a character are treated as valid characters and is simulated with three out of seven neural networks to be classified.

Thus with the above grouping method, which uses aspect ratio, the width to height ratio, as a basis, helps to alleviate the partially segmented character problem which occurres primarily by deletion of 'Matra'.

4.3.2 Features Rather than Image to Train the Neural Network

Since training or recognizing a Bengali character using a neural network with an image suffers from several drawbacks, it often fails to recognize characters effectively. It is because of the fact that a resized small image carries too little information to recognize a large set of characters in Bengali and again a little translated image can cause a failure to detect a character. For these serious drawbacks, in our method we have used features as the basis of both training and recognizing of a character.

Using features, rather than image gives us several advantages:

1. It takes **several order of magnitude less epochs** to train the neural network using only the features extracted from the resized image.

2. An small 20×20 image needs **400 neurons** in the input layer. Where as it is enough to use only **29 neurons** in our case in the input layer to represent the features.

3. When all the features are combined, it carries **more information** about a character than a small 20×20 image.

4. Since, there is a huge reduction of neurons in the input layer, it also means faster simulation or **faster recognition** process.

The features that are extracted from a 20×20 image are as follows:

1. 'Matra'/Headline
2. Left Sidebar
3. Right Sidebar
4. Eccentricity
5. Centroid
6. Horizontal Projection Skewness
7. Vertical Projection Skewness
8. Horizontal Projection Kurtosis
9. Vertical Projection Kurtosis
10. Euler number
11. Horizontal Ratio
12. Vertical Ratio
13. Sum of on pixels in 5×5 sub-image

Here all the features except the 'centroid' and the 'sum-of-on-pixels' needs only one neuron each in the input layer. The 'centroid' needs two neurons and the 'sum-of-on-pixels' needs 16 neurons in the input layer. So there are in total 29 neurons in the input layer.

4.3.3 Multiple ANN rather than Single ANN

To ensure that an artificial neural network (ANN) recognizes all the characters effectively it is required that it works only with a small set of characters. Again when only a single ANN is used to recognize all the characters, it is almost impossible to recognize a character in case of a failure of that ANN.

To alleviate these serious drawback, we have designed seven artificial neural networks rather than using just a single one to recognize a character. To

recognize a single character three out of seven ANN is simulated and their results are then compared to derive the final result.

The seven ANN is grouped into three classes. This classification is based on some structural features of the characters which can uniquely separate characters into separate classes.

Matra based Classifier #1

The first grouping of characters is based on the presence of 'Matra'. This class consists of two ANN and recognizes two disjoint sets of characters.

The first ANN in this class recognizes only those characters that have 'Matra' and the second ANN recognizes only those characters that don't have 'Matra'.

Table 4.1: Character Set of Classifier 1

ANN	No. of Characters	Character Set
Net1 (characters with 'Matra')	15	ক, ঘ, চ, ছ, জ, ঠ, ড, ঢ , ত, দ, ন, ফ, ব, ভ, ম
Net2 (characters without 'Matra')	10	খ, গ, ঙ, ঝ, ঞ, ণ, থ, ধ, প, ে

Sidebar based Classifier #2

The second grouping of characters is based on the presence of 'sidebar'. This class consists of three ANN and recognizes three disjoint sets of characters.

The first ANN in this class recognizes only those characters that have a 'left sidebar', the second ANN recognizes characters those have 'right sidebar' and the third ANN recognizes those that do not have any 'sidebar'.

Table 4.2: Character Set of Classifier 2

ANN	No. of Characters	Character Set
Net3 (characters with left sidebar)	2	চ, ঢ
Net4 (characters with right sidebar)	13	ক, খ, গ, ঘ, ঝ, ঞ, ণ, থ, ধ, ন, প, ব, ম
Net5 (characters without sidebar)	10	ঙ, ছ, জ, ঠ, ড, ত, দ, ফ, ভ, ে

Loop based Classifier #3

The last grouping of characters is based on the presence of 'loops'. This class consists of two ANNs and recognizes two disjoint sets of characters.

The first ANN in this class recognizes only those characters that have 'loop' and the second ANN recognizes those that do not have any 'loop'.

Table 4.3: Character Set of Classifier 3

ANN	No. of Characters	Character Set
Net6 (characters with loops)	12	ক, ঘ, ঙ, চ, ছ, ঝ, ঞ, ঠ, ধ, প, ব, ম
Net7 (characters without loops)	13	খ, গ, জ, ড, ঢ , ণ, ত, থ, দ, ন, ফ, ভ, ে

Since three out of seven ANNs are simulated from all the three class to recognize a single character, the probability that all the ANNs will give a wrong result is reduced greatly. So in this case if one ANN fails to recognize a character correctly, there are two other networks to choose the correct answer. So the

drawback that was associated with a single ANN for recognizing characters is alleviated here to a greater extent.

Again, since each of the artificial neural network has to be trained with only a small subset of characters from a larger set, the ANN can be trained in less amount of time. It also becomes much more effective as it has to recognize characters only from a small subset.

4.3.4 Simulation Process

To recognize a single character, in our method, three out of seven neural networks needs to be simulated based on its structural features. If the given character to be recognized has a 'Matra' then it will be simulated using Net1 of the above table. And if it doesn't have 'Matra', then it will be simulated using Net2. So in classifier #1, only one neural network will be simulated for each character, since a character cannot have or not have a 'Matra' at the same time. The same thing is true for the other two classifier stages. So there are in total three ANNs that get simulated.

Let us consider the simulation process for the letter ক. Then it is obvious that the letter has a 'Matra'. So the Net1 gets simulated to recognize what character it is. Now the feature extraction module finds that the letter has a right sidebar. So the Net4 gets simulated. Again, it is found that the letter contains a loop. So, lastly the Net6 gets simulated.

4.3.5 Decision Making Process

The results of all the three ANNs are combined to generate the overall result of the recognition process. If at least two of the neural networks results matches, then this result is considered as the final result. And if none of the neural networks results matches with one another, then the final result is considered as the result of the neural network that has the highest percentage of matching.

The process of decision making described above is only one part of the overall recognition process. This part is used to recognize only the sub-symbols or sub-characters with the help of ANNs. Further we have to determine what is above or below that sub-character and whether those modifies the character recognition process.

In the last stage of the character recognition process, the information about coordinates of bounding box of sub-characters and the context is used to merge some of the sub-characters. The sub-characters are then converted to actual Bengali characters.

To illustrate this process, let us consider the recognition process of the Bengali letter ট. To recognize this character, first we have to recognize the sub-character ট. Then we have to figure out that whether there is anything above the letter ট. If we can find any thing above the ট then we can figure out that the recognized character is ট.

But the above process is complicated by the fact that, there can be anything above a ট. To visualize the problem, let us consider the following figure.

Figure 4.8: Problems with merging a sub-character

There may be a roshho-ikar or a dirgho-ikar or an ou-kar. All these things may lead to an erroneous recognition of ট. So the sub-character merging process has to be very effective and should be based on statistical features.

Chapter 5

Design and Implementation of the OCR

This chapter describes the design and implementation of the OCR system & Graphical User Interface. The implementation of the System Modules is written using the MATLAB.

5.1 Design of the OCR

The OCR system consists of two major components.

a. Training Component
b. Recognizer Component.

Three neural network based classifiers are used in the recognition stage. All the classifiers need to be trained from properly labeled input examples. And the combined result of all three classifiers is considered as the correct classification.

a. Training Component:

This component is responsible for creating and training the classifiers. The major steps are presented in the following algorithm:

Algorithm 5.a: *training()*

1. Call *create_All_Nets() [Algorithm 5.8].*
2. Call *train_All_Nets() [Algorithm 5.9].*

b. Recognizer Component:

This component is responsible for the recognization of the characters. The major steps are presented in the following algorithm:

Algorithm 5.b: OC*R_Engine()*

1. Read the image file by calling *read_Image() [Algorithm 5.1].*
2. Call *line_segmentation() [Algorithm 5.2]* to break the lines.
3. For each line call *zone_Segmentation() [Algorithm 5.4]* and *word_Segmentation() [Algorithm 5.3]*.
4. For each word of a line call *character_Segmentation() [Algorithm 5.5]*, which segments and recognizes all characters in a word.
5. Store recognized words in an array.
6. call *send_to_Word() [Algorithm 5.10]*to output the characters in an MSWord document.

5.2 The MATLAB Implementation

The modules of the OCR system along with the GUI are written in the MATLAB environment.

MATLAB® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve 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.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. 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.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular

classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

In this project, we have extensively used the image processing and neural networks toolboxes.

5.2.1 Algorithms for Different Stages of the OCR System

Algorithm 5.1

This algorithm reads an image file and converts it to a binary image.

read_Image()

1. Read image file.
2. Resize the image to 1024x*width*. Here *width* is determined by the imresize() automatically to keep the actual aspect ratio of the image.
3. Convert the image into binary image.
4. Negate the image.

MATLAB Implementation:

Here following functions are used:

a. **imread() -** Read image from graphics files. This function can read bmp, jpeg, tiff, gif, ppm etc image file formats.
b. **Imresize()** – Used to resize the image into a fixed size.
c. **im2bw() -** Convert an intensity image to a binary image, based on threshold.
d. **bwlabel() -** Label connected components in a binary image.

Algorithm 5.2

This algorithm segments the image into separate lines.

line_Segmentation()

1. Calculate the row sum of the binary image matrix.
2. Define a threshold (row sum less than 3) to delimit each line.
3. For each row check whether it is below threshold or not.
4. Segment lines as the rows which are above the threshold.

5. Returns the starting and ending row number of each line.

MATLAB Implementation:

Here following functions are used:

a. **size()** – Returns the size of the dimensions of the matrix given as its input argument.
b. **Sum() –** calculates the summation of the row/column and returns a column/row vector.

Algorithm 5.3

This algorithm segments a line into words.

word_Segmentation()

1. Calculate the column sum for the given line.
2. Define a threshold (column sum less than 2) to delimit each word.
3. For each column check whether it is below threshold or not.
4. Segment words as the columns which are above the threshold.
5. Returns the starting and ending column number of each word.

MATLAB Implementation:

Here following functions are used:

a. **size()** – Returns the size of the dimensions of the matrix given as its input argument.
b. **Sum() –** calculates the summation of the row/column and returns a column/row vector.

Algorithm 5.4

This algorithm segments a line into three zones.

zone_Segmentation()

1. For a line calculate the row sum for each row.
2. Find the maximum row sum value.
3. The rows that are at least 75% of the maximum value and are consecutive in upper or lower direction from the max valued row, consider these rows as 'Matra' rows.

4. Find the location of the base line.
5. Return the line and its location of 'Matra' and base line.

MATLAB Implementation:

Here following functions are used:

a. **sum() –** calculates the summation of the row/column and returns a column/row vector.
b. **max() –** Calculates and returns the maximum element of a row or column vector.

Algorithm 5.5

This algorithm segments a word image into individual character images.

character_Segmentation()

1. For a word delete the 'Matra' rows.
2. Delete everything above the 'Matra' row.
3. Delete the bottom portion of the characters.
4. Segment the characters as the connected components.
5. If the bounding box of a character is too small, ignore it.
6. If the width to height ratio falls between 0.1 to 0.4 range then try to detect it as an kar (akar/ ekar).
7. If the width to height ratio falls in the range 0.6 or above, call neuralNetSimulation() *[Algorithm 5.6]* to detect which character it is.
8. Detect the u-kars in the lower portion of a character.
9. Put the detected charcters, akars, ekars, ukars in an array and return it for each word.

MATLAB Implementation:

Here following functions are used:

a. **regionprops –** measure properties, such as area, bounding box, centroid, eccentricity, euler number, etc of image regions.
b. **max() –** Calculates and returns the maximum element of a row or column vector.
c. **sim()-** Simulate a neural network
d. **round() –** Rounds the input argument to the nearest integer.

e. **size() -** Returns the size of the dimensions of the matrix given as its input argument.

Algorithm 5.6

This algorithm simulates three out of seven neural networks and classifies the character.

neural_Network_Simulation()

1. Call find_Features() *[Algorithm 5.7]* to extract the feature of the given character.
2. Based on 'Matra' simulate this character on Network 1 or Network 2.
3. Based on sidebar simulate this character on Network 3 or Network 4 or Network 5.
4. Based on loops simulate this character on Network 6 or Network 7.
5. Combine the result of these three modules.
6. Return the result with the percentage of match.

MATLAB Implementation:

Here following functions are used:

a. **Sim()** – Simulates a neural network and returns a vector.
b. **max() –** Calculates and returns the maximum element of a row or column vector.

Algorithm 5.7

This algorithm extracts the features from a 20x20 character image.

find_Features()

1. Calculate the row sum in the upper portion of the image of the character.
2. If the row sum is 80% or above of the total width then the character has 'Matra'.
3. Calculate the column sum in the first half and in the second half of the image of the character.
4. If the column sum is 80% or above of the total height then the character has a left or right sidebar.

5. Calculate the eular number, eccentricity, centroid, sum of on pixels in the 5×5 sub image, horizontal ratio, vertical ratio.
6. Calculate the horizontal projection skewness, vertical projection skewness, horizontal projection kurtosis, vertical projection kurtosis using the method of moments.
7. Return all the features as an structure.

MATLAB Implementation:

Here following functions are used:

a. **Sum() –** calculates the summation of the row/column and returns a column/row vector.
b. **max() –** calculates and returns the maximum element of a row or column vector.
c. **regionprops –** measure properties, such as area, bounding box, centroid, eccentricity, euler number, etc of image regions.

Algorithm 5.8

All the neural networks are created using this algorithm.

create_All_Nets()

1. Load training character file.
2. For each character call find_Features(). *[Algorithm 5.7]*
3. Create Neural Network.

MATLAB Implementation:

Here following functions are used:

a. **newff()-**Create a feed-forward backpropagation network

Algorithm 5.9

To train all the networks this algorithm is used.

train_All_Nets()

1. Load training character file.
2. Load previously created neural networks.
3. For each character call find_Features(). *[Algorithm 5.7]*

4. Create feature vectors for each characters.
5. Train the network using feature vectors.
6. Save training data.

MATLAB Implementation:

Here following functions are used:

a. **train()-** training with adaptive learning rate

Algorithm 5.10

This algorithm is used to show the output as an MSWord document.

Send_to_Word()

1. Invoke MSWord using activeX server.
2. Open a new document.
3. Set the font type to AdarshaLipiNormal.
4. Set the font size to 20.
5. Copy the final output as type char in the clipboard.
6. Invoke the 'Paste' command in MSWord.

MATLAB Implementation:

Here following functions are used:

a. **actxserver() –** returns a handle for a remote application.
b. **set() –** set properties for object.
c. **invoke() –** invoke method on object or interface, or display methods.
d. **clipboard() -** copy and paste strings to and from system clipboard.

5.2.2 Graphical User Interface (GUI)

a. The Training Program

A Matlab GUI was written to encapsulate the steps involved with training an OCR system. This GUI permits the user to load images, to binarize and segment them, to train each character and to save these trained neural networks for future analysis. It saves the data in a **.mat* file as training examples for the neural network and trains the neural network by loading this.

Corresponding MATLAB File: train.m train.fig trainAllNets.m trainingCharacters.m findFeatures.m letters.mat trainingChars.mat trainedNeuralNets.mat

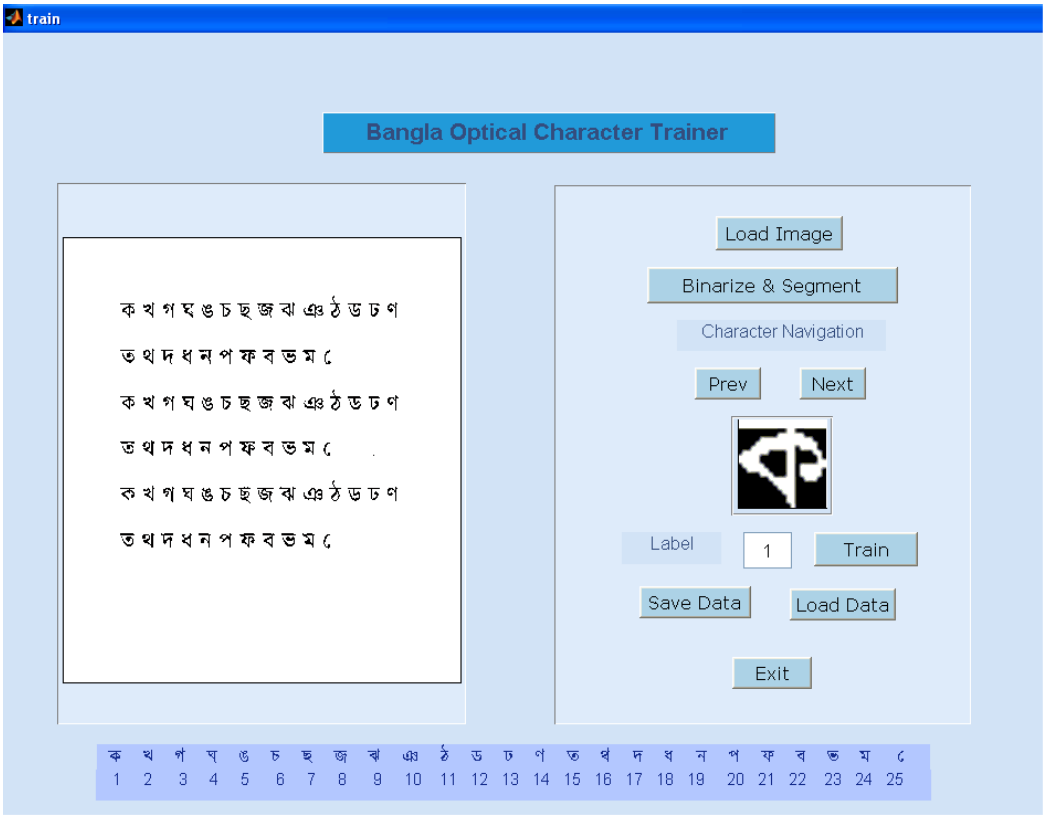


Figure 5.3: The Training Component - Graphical User Interface

Loading an Image: Images can be imported into the GUI by clicking on the "**Load Image**" button. BMP, JPG and TIF file formats are supported. Most of the testing was done with grayscale BMP images.

Binarization and Segmentation: After opening an image, it can be converted to black and white and segmented by clicking in the button in the upper right corner of the window. This button will also extract various features.

Labeling the Characters: Once the training image is segmented, a character will appear above the text box titled **Label.** It's the user's job to label each

segmented character appropriately. The labels corresponding to each character are shown at the bottom of the window. Once a character label has been entered into the text box, "**Next**" button is clicked to move to the next character. One can navigate back by clicking "**Prev**".

Training the Neural Network: Using the segmented and labeled characters, the neural networks are trained by clicking on the "**Train**" button. The training process of the neural networks is shown to the users using graphical figures.

Saving and Loading the Trained Data: The trained neural netoworks can be saved by clicking on the "**Save Data**" button. Any previously saved data of neural networks can be loaded for further training by clicking on the "**Load Data**" button.

b. The Bangla Character Recognition Program

A Matlab GUI was written to encapsulate the steps involved with running an OCR system. This GUI permits the user to load images and perform the recognition of the individual character images. The trained data are loaded automatically. The user is allowed to save the recognized characters as an MSWord Document.

Corresponding MATLAB File: banglaOcr.m banglaOcr.fig ocrEngine.m lineSeg.m zoneSeg.m wordSeg.m charSeg.m findFeatures.m neuralNetSimulation.m send2Word.m

Loading an Image: Images can be imported into the GUI by clicking on the "**Load Image**" button. TIF, BMP and JPG file formats are supported. The load image will be shown on the left side of the window in the "**Original Text**" panel.

Zoom In/Out: User can zoom in/out the image using "**Zoom**" button. Clicking the left mouse button on the image will perform zoom in operation and clilcking the left mouse button along with the shift key will perform zoom out operation.

Perform OCR : Clicking the "**Start OCR**" button the recognition process starts. The image is converted to black and white, segmented and then recognized. Each character recognized from the original image is printed on the Text box titled **Output Text**.

OCR Options: To perform the OCR operation, the users can choose one of the two options by clicking either on the "**Full Text**" button or the "**Step by Step**" button. In the **Full Text** option, the whole image is processed and recognized at a time without any user intervention. But in the **Step by Step** option, each character of the segmented image is individually recognized. Image of each character along with the recognized character are shown on the small panel placed in the middle of the window one by one. In this mode the user can verify the result. Then the next character can be processed by clicking on the "**Next**" button.

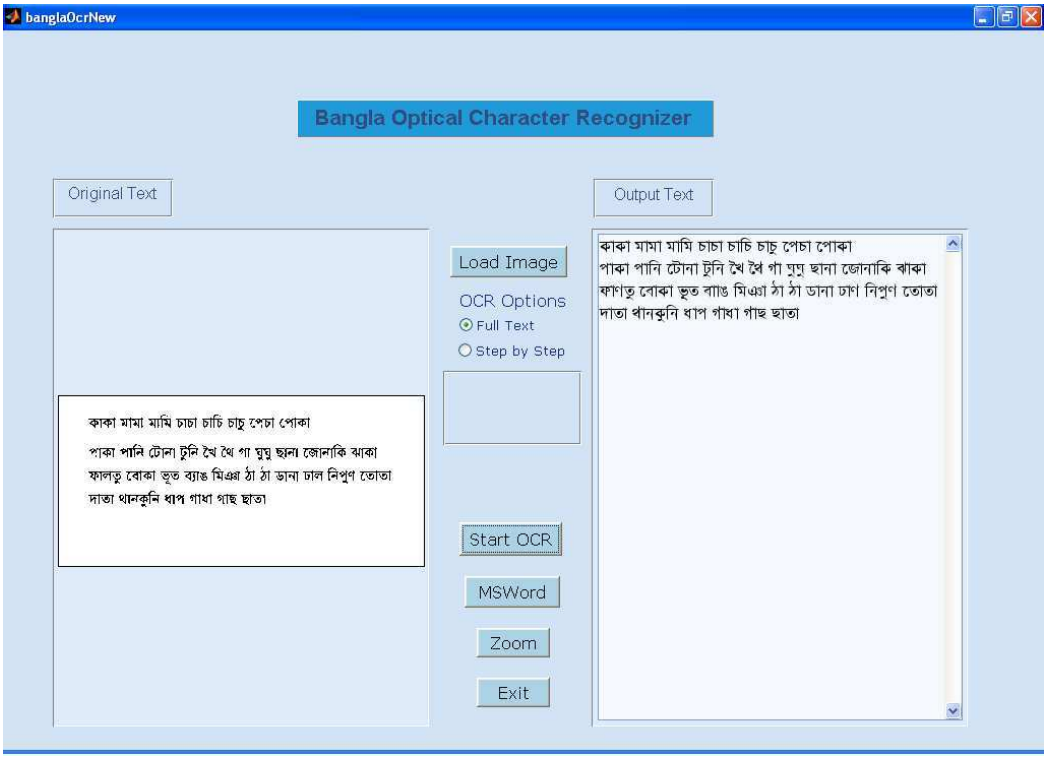


Figure 5.4: The Recognition Component - Graphical User Interface

Saving the Characters: Once the image is processed and recognized, the user is allowed to pass the recognized characters to an MSWord Document by clicking on the "**MSWord**" button. Then the user can edit and then save it as an MSWord Document.

Chapter 6

Experimental Results

This chapter presents the performance analysis of the developed OCR system by showing the accuracy and execution time.

OCR system, developed in this project, can recognize 25 Bengali characters from scanned image of written text of standard size. The recognizable characters are shown in the following figure.

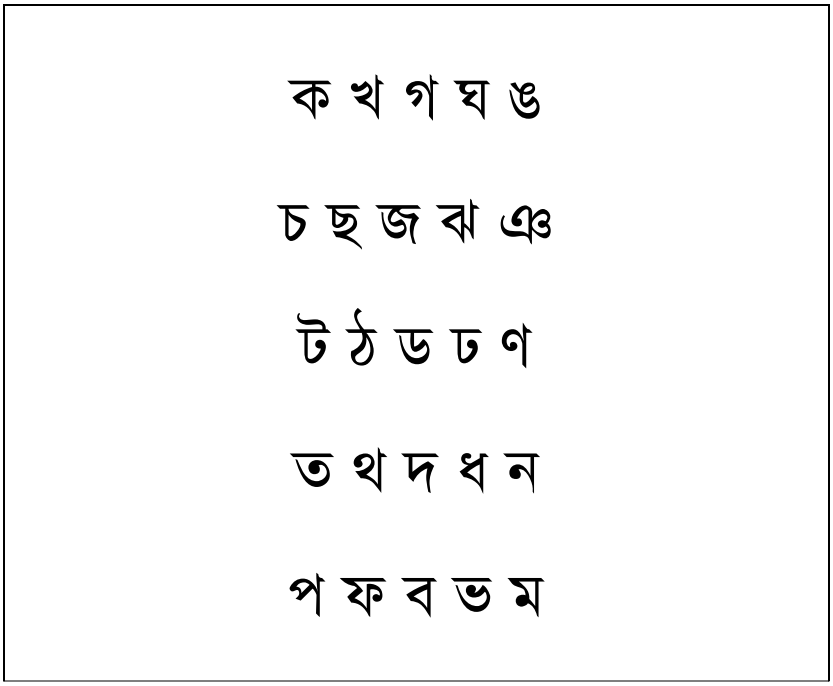


Figure 6.1: List of characters recognizable by the OCR

Along with this 25 characters, our OCR system is able to recognize some of the vowel modifiers or 'Kars'. These are shown in the following figure.

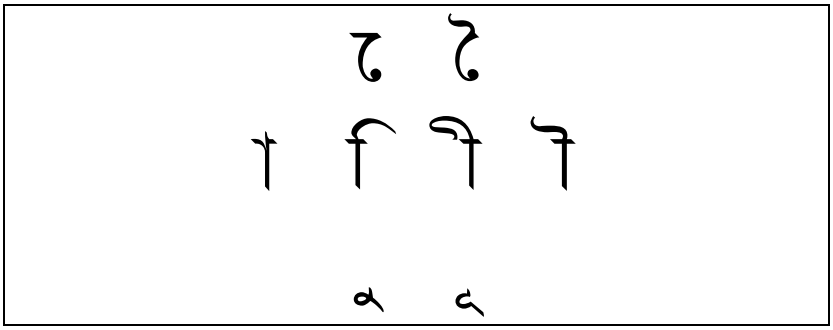


Figure 6.2: List of vowel modifiers recognizable by the OCR

6.1 Performance Analysis

To measure the success rate and efficiency of the character recognition process, we ran the OCR software on several text images each of which consists of several characters written as Bengali text.

We tested the OCR for the following properties:

- Number of unrecognized characters or errors in a text image.
- Total number of characters in that text image.
- Amount of time taken by the OCR system to recognize all that characters.
- Success rate of the OCR system.

Using this neural network OCR was performed on several test images with resolution 2500x2500 which are resized to 1024 x *width*. Here *width* is determined by the imresize() automatically to keep the actual aspect ratio of the image. The table 6.1 shows the experimental results.

Table 6.1: OCR Performance with image size 1024xwidth

Figure Name*	**Scan Resolution (dpi)**	**Number of Characters**	**Number of Errors**	**Accuracy Rate(%)**	**Time (sec)**
Test1_300.bmp	**300**	**297**	**8**	**97.31**	**30.39**
Test2_300.bmp	**300**	**142**	**2**	**98.59**	**13.39**
Test3_150.bmp	**150**	**25**	**2**	**92.00**	**4.38**

**see appendix*

The OCR was also performed on the same set of test images but resized to a different resolution of 800 x *width*. Here *width* is determined by the imresize()

automatically to keep the actual aspect ratio of the image. The table 6.2 shows the experimental results.

Table 6.2: OCR Performance with image size 800xwidth

Figure Name*	Scan Resolution (dpi)	Number of Characters	Number of Errors	Accuracy Rate(%)	Time (sec)
Test1_300.bmp	**300**	**297**	**27**	**90.91**	**31.51**
Test2_300.bmp	**300**	**142**	**9**	**93.66**	**13.18**
Test3_150.bmp	**150**	**25**	**3**	**88.00**	**4.06**

**see appendix*

Thus from the above experimental results it is observed that the time to process images with lower resolution is not much less then the images with high resolution but the accuracy of the OCR system degrades severely.

All most 98% accuracy in recognition was achieved when the font type of the input image was same as the trained data image and when the test page was scanned with higher resolution.

The experimental results have shown that the OCR system has an error rate from 5-10%. There are several reasons for this. Some of the reasons are:

- Some of the characters have very similar structural and statistical features. Such as গ and গ.
- Some of the features may be falsely detected by the feature extraction module because of distorted image or image with very low resolution. For example, there is no loop in গ. But sometimes the feature extraction module detects a loop in গ because sometimes only one pixel can connect two disconnected parts forming a loop.

- Rotation or translation may lead to recognize a character incorrectly. For example a rotated or translated character image is likely to have a very different aspect ratio.

6.2 System Configuration

We would like to argue that with more training images with distorted characters along with perfectly shaped once will improve the performance of the classifiers. All the experimental results where executed using MATLAB version 7.5 running on an Intel Pentium 4 – 2.4GHz, 384MB RAM, Windows XP platform.

Chapter 7

Discussion

This chapter presents conclusive discussion and some future improvements.

7.1 Conclusion

In this project, we have, implemented an OCR for Bangla characters. The OCR system has been quite successful in the recognition of input images. The experimental result shows us that with the test image with same font type the accuracy rate is 95 - 100%. The execution time for example for a 250 characters image of size 2500X2500 took around 40 seconds.

We have implemented the OCR system in MATLAB. The image processing and neural network toolboxes of MATLAB have been very helpful for this purpose. The OCR system has been designed according to the typical OCRs found with some improvements. The input images have been preprocessed before feature extraction and classification. The preprocessing step successfully removes noises from the image to some extent. The segmentation operation is performed to isolate the characters. As this operation also detects lines and words from the image the output text file that the OCR system generate has almost the same structure/formatting as the original image.

We have basically used three different classifier modules, all of which are neural based classifier which is trained with 13 statistical and structural features that were extracted from the preprocessed image of the characters. At the recognition stage after the preprocessing and the segmentation stages, the images resized as 20x20 are fed as input to the feature extraction module and the output of this module is then fed into the neural network based three classifiers. All the modules work separately, with a different set of characters. The results of the three classifiers are combined to gain more accuracy. Since the recognition time for an image, with for example 250 characters of size 2500x2500 pixels, is around 40 seconds, the modules can be executed parallely, thus reducing the execution time.

To ensure that an artificial neural network (ANN) recognizes all the characters effectively it is required that it works only with a small set of characters. Again when only a single ANN is used to recognize all the characters, it is almost impossible to recognize a character in case of a failure of that ANN. To alleviate these serious drawbacks, we have designed seven artificial neural networks rather than using just a single one to recognize a character. To recognize a single character three out of seven ANN is simulated and their results are then

compared to derive the final result. The seven ANN is grouped into three classes. This classification is based on some structural features of the characters which can uniquely separate characters into separate classes.

The experimental results have shown that the OCR system has an error rate from 5-10%. There are several reasons for this. Some of the characters have very similar structural and statistical features. Again, some of the features may be falsely detected by the feature extraction module because of distorted, rotated, translated image or image with very low resolution.

The output of the OCR is allowed to be saved as text or in MSWord for further processing. Spacing between the words in the original image is not retained in our system. It has been very difficult to accurately measure this spacing because of the difference between the widths of the words in the input image.

7.2 Future Work

Based on character recognition system that we have implemented, it should be easy to detect all the individual bangla letters. We will just need to train the classifiers accordingly. But recognition of bangla compound characters (songjukkto okkhor) would require more pre-processing work. We have achieved some progress in this respect by isolating the letters from such constructs that are mentioned above. But to recognize the compound characters will require more analysis and experimentation.

We have not used any error detection and correction module in our recognition systems. Also, we have not included any feedback mechanism for the rejected characters for finer analysis. By fine tuning the classifiers and by using an error correction module, we hope to achieve about 99% of recognition accuracy on average.

In text/word OCR a post processing phase can be implemented to correct some errors. Dictionary based methods have proven to be most effective for error detection and correction. Once all the characters of a word have been recognized, the word is looked up in a dictionary. If it is present, we can assume the characters are correct (although this is not absolutely sure), if not, an error has been detected. The word can be changed to the most similar word in the

dictionary. If there is more than one candidate, the system has to ask the user unless it has some higher level knowledge about the language it has to recognize: its grammatical structure, for example.

The proposed method needs further development and testing before being fitted into a production OCR. The future steps are to make the system robust so that, it can read documents in any style and several popular fonts as well as to ensure that the system performance does not degrade appreciably on documents of lower grade.

REFERENCES

[1] J.C. Rabinow. Wither OCR. *Datamation*, pages 38042, Jul. 1969.

[2] S Impedovo, L. Ottaviano, and S. Occhinegro. *Optical character recognition – a survey*. International Journal of Pattern Recognition and Artificial Intelligence, 5(1 & 2):1-24, 1991.

[3] C. C. Tappert, C. Y. Suen, and T. Wakahara. *State of the art in online hand-writing recognition*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 12(8), Aug. 1990.

[4] T. Pavdilis and Z. Jiangying. *Page segmentation and classification.* CVGIP: Graphical models and image processing, 54(6), 1992.

[5] Mohammad Anwar Hossain, Mohammad Kabir Hossainand Muhammad Khalid Adnan. *Design and Testing of a Set of Bengali OCR Algorithms.* B.Sc. Thesis Submitted to Dept. of Computer Science & Engineering, University of Dhaka, March 2006

[6] Project Gutenberg Official Home Site. [http://www.promo.net/pg/] (01.07.08)

[7] Jalal Uddin Mahmud, Mohammed Feroz Raihan and Chowdhury Mofizur Rahman. *Development of a Complete OCR by Feature Oriented Decision Tree Based Classifier.* Proceedings of ICCIT, 2003.

[8] A. K. Roy, B. Chatterjee. *Design of Nearest Neighbor Classifier for Bengali Character Recogniton.* J. IEEE 30, 1984.

[9] Abhijit Dutta, Santanu Chaudhury. *Bengali Alpha Numeric Character Recognition Using Curvature Features.* Pattern Recognition vol. @6, pp. 1707-1720, 1993.

[10] B. B. Chaudhuri, U. Pal. *A Complete Printed Bangla OCR System.* Pattern Recognition Vol 31, pp. 531-549,1997.

[11] Arnab Chakraborty. *An Attempt to Perform Bengali Character Recongnition.* PHD Thesis, Submitted to Dept. of Statistics, Stanford University, 2003.

[12] *BOCRA* (Bangla Optical Character Recognition Application) by Deepayan Sarkar. [www.stat.wisc.edu/~deepayan/bocra/] (01.07.08)

[13] *"Chitrankon",* an OCR for the Devnagari script released by C-DAC India. [http://www.cdac.in/html/gist/products.asp#1] (01.07.08)

[14] Lijun Zhou, Yue Lu, Chew Lim Tan. *Bangla/English Script Identification Based on Analysis of Connected Component Profiles.* DAS, 2006.

[15] Wikipedia Entry for "Character (computing)". [http://en.wikipedia.org/wiki/Character_%28computing%29] (01.07.08)

[16] R. Karturi and L. O'Gorman. *Document image analysis: a bibliography.* Machine Vision and Application, pages 231--243, 1992.

[17] W. K. Pratt. *Digital Image Processing.* John Wiley and Sons Inc., second edition, 1991. ISBN: 0-471-85766-1.

[18] Amin Ahsan Ali and Syed Monowar Hossain. *Optical Bangla Digit Recognition using Backpropagation Neural Networks.* B.Sc. Thesis Submitted to Dept. of Computer Science & Engineering, University of Dhaka, April 2003

[19] C. J. Hilditch. *Linrar Skeletons from Square Cupboards.* In B. Meltzer and D. Mitchie, editors, Machine Intelligence IV. Edinburgh: University Press, 1969.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[21] U. Pal and B. B. Chaudhuri. *OCR in Bangla: an Indo-Bangladeshi Language.* IEEE, 1994.

[22] Jalal Uddin Mahmud, Mohammed Feroz Raihan and Chowdhury Mofizur Rahman. *A Complete OCR System for Continuous Bengali Characters.* IEEE, 2003.

[23] U. Pal and B. B. Chaudhuri. *An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari (Hindi).* IEEE, 1997.

[24] G. F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Fourth Edition, Addison Wesley, 2002.

[25] Veena Bansal, R. M. K. Sinha. *A Complete OCR for Printed Hindi Text in Devanagari Script.* IEEE, 2001.

[26] A. Mucciardi and E. E. Gose. *A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties.* IEEE Trans. on Computers, C-20(9):1023--1031, Sep. 1971.

[27] G S Lehal and Chandan Singh. *A Gurumukhi Script Recognition System.* IEEE, 2000.

[28] C. Y Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. *Computer recognition of unconstrained handwritten numerals.* Proceedings of the IEEE, 80(7):1162--1180, Jul. 1992.

[29] Robert Hetch-Nielsen. *Neurocomputing*. Addison-Wesley Publishing Company, January, 1991.

[30] Albert Nigrin. *Neural Networks for Pattern Recognition*. The MIT Press, 1993.

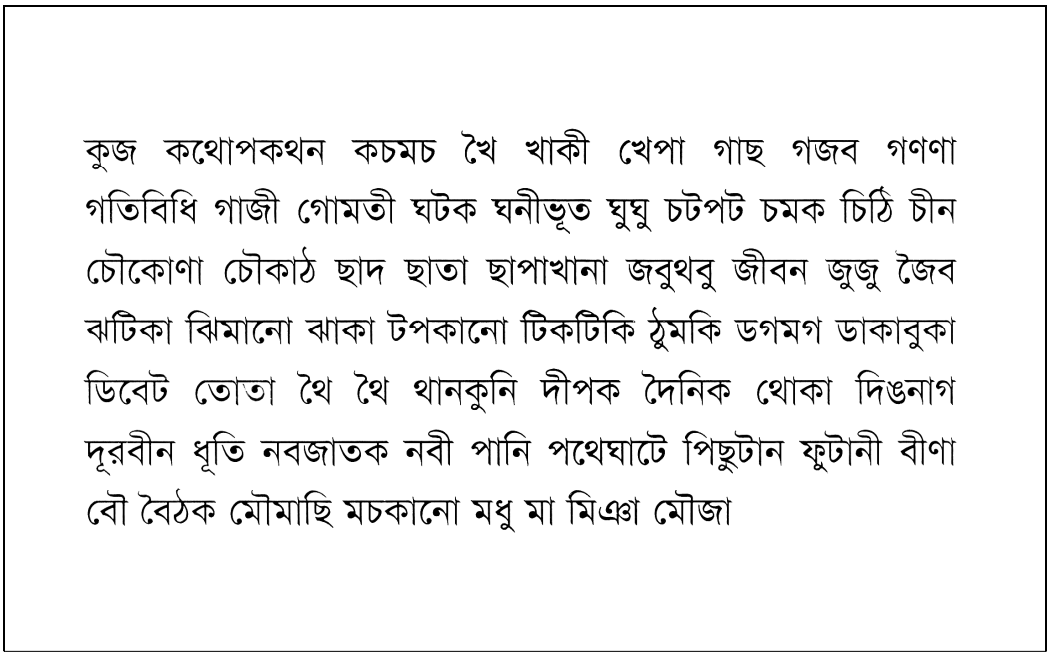
[31] Granino A. Korn. Neural Network Experiments on Personal Computers And Workstations. The MIT Press.

[32] Anbumani Subramanian and Bhadri Kukbendran. *Optical Character Recognition of Printed Tamil Characters.* [http://www.ee.vt.edu/~anbumani/ocr/]

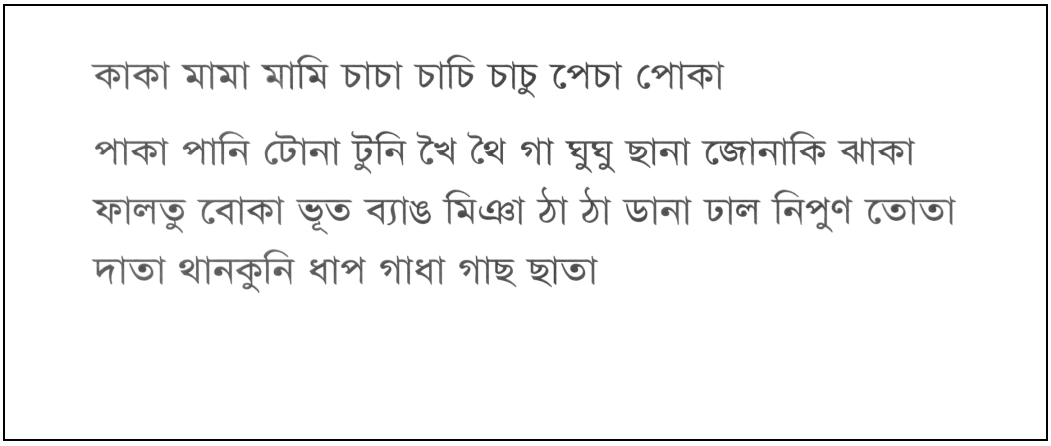
[33] Gonzalez R. C. and Woods R. E.. *Digital Image Processing.* Addision-Wesley Publishing Company-1992.

APPENDIX

Test1.bmp



Test2.jpg



Test3.jpg