

**Bangla Optical Character Recognition using Kohonen Network**

**A Thesis**

**Submitted to the Department of Computer Science and Engineering  
of  
BRAC University**

**by**

**Adnan Md. Shoeb Shatil**

**ID: 01201070**

**CSE**

**In Partial Fulfillment of the Requirements for the Degree**

**of**

**Bachelor of Science in Computer Science**

**May 2006**

### **Declaration**

I hereby declare that this thesis is based on the results found by myself.  
Materials of work found by other researcher are mentioned by reference.

Signature of Supervisor

Signature of Author

## **Acknowledgement**

It was my great pleasure to be involved with the thesis study on “Bangla Character Recognition”.

I would like to extend my sincere thank to Dr. Mumit Khan, Associate Professor, CSE Dept. BRAC University, Dhaka for his valuable support to fulfill my academic assignment.

Also I honor his immense contribution throughout the process of study. His contribution in the preparation of the concept paper, literature, methodology and writing this report is highly acknowledged.

I am also thankful to my co-researcher and senior friend Mr. Abul Hasnat for his helpful and appreciable assistance at the beginning of the study and clearing the subject matters.

My sincere thanks also go to seniors Mr. Nawsher Ahmed Noor and Mr. Mortuza Habib for providing me a large collection of documents and papers and also BRAC University Research Group for their technical help.

Finally, I would like to extend my appreciation to all BRAC University friends, course mates and study partners for their support and co-operation.

Adnan Md. Shoeb Shatil

02101078

CSE Dept.

BRAC University.

## **Abstract**

This report is based on Optical Character Recognition (OCR) specially Bangla (National Language of Bangladesh) Optical Character Recognition process and its steps. The whole idea is converting text images into editable texts. In this report the raw data are offline printed characters. These characters are collected from computer images and also scanned documents. As no skew conversion and correction is considered in processing or preprocessing stages, so the documents are scanned with great care. After collecting the raw data, they are pre-processed with gray scale conversion and then black and white conversion. When we get the black and white image, there are only two types of data on that image. So we can consider it as a binary file. The character area is represented by 1 (one) and the rest of image area is represented with 0 (zero). On the next step we grab the whole image, take the binary data and placed it on to 25 X 25 pixels by pixel mapping procedures. Then we collect 625 bit long vector for each word or character. This vector is then trained with Kohonen neural network is considered as classification stage. So an optical character is recognized with Kohonen Network.

## Contents

| Topics                                                           | Pages |
|------------------------------------------------------------------|-------|
| Declaration                                                      |       |
| Acknowledgement                                                  |       |
| Abstract                                                         |       |
| Chapter 1: Introduction                                          | 01-01 |
| Chapter 2: Character Recognition procedure                       | 02-03 |
| 2.1 Bangla Character Recognition Procedures with Kohonen Network | 03    |
| Chapter 3: Data Collection                                       | 04-04 |
| Chapter 4: Image processing                                      | 05-07 |
| 4.1 RGB to Grayscale Image conversion                            | 05    |
| 4.2 Grayscale to Binary Image conversion                         | 06    |
| Chapter 5: Feature Extraction                                    | 07-11 |
| 5.1 Pixel Grabbing from Image                                    | 08    |
| 5.2 Finding probability of making square                         | 08    |
| 5.3 Mapped to sampled area                                       | 09    |
| 5.4 Creating Vector                                              | 10    |
| 5.5: Representing a character with a model number                | 11    |
| Chapter 6: Kohonen Neural Network                                | 11-19 |
| 6.1 Introduction to Kohonen Network                              | 11    |
| 6.2 The structure of Kohonen Network                             | 13    |
| 6.3 Sample Input to Kohonen Network                              | 14    |
| 6.3.1 Normalizing the input                                      | 14    |
| 6.3.2 Calculating each neuron's output                           | 15    |
| 6.3.3 Mapping to bipolar                                         | 16    |
| 6.3.4 Choosing a winner                                          | 16    |
| 6.4 Kohonen Network learning procedure                           | 17    |
| 6.5 Learning algorithm flowchart                                 | 17    |
| 6.5.1 Learning rate                                              | 18    |
| 6.5.2 Adjusting weight                                           | 18    |
| 6.5.3 Calculating the errors                                     | 19    |
| 6.6 Recognition with Kohonen network                             | 19    |

|                                       |       |
|---------------------------------------|-------|
| Chapter 7: Functional Output          | 20-20 |
| 7.1 Accuracy Rates                    | 20    |
| 7.2 Drawbacks                         | 20    |
| Chapter 8: Conclusion and future work | 21-21 |
| 8.1 Conclusion                        | 21    |
| 8.2 Future work                       | 21    |
| Glossary                              | 22-25 |
| References                            | 26    |

## List of Tables

| Name of the Tables                                               | Page |
|------------------------------------------------------------------|------|
| Table 1: List of Bangla Characters                               | 01   |
| Table 2: Initial pixel data from image                           | 08   |
| Table 3: Separating the pixels                                   | 08   |
| Table 4: Separating the pixels                                   | 09   |
| Table 5: Making squares                                          | 09   |
| Table 6: Sample Inputs to a Kohonen Neural Network               | 14   |
| Table 7: Connection weights in the sample Kohonen neural network | 14   |
| Table 8: Accuracy rates corresponding to different sectors       | 20   |

## List of Figures

| Name of the Figures                                                          | Page |
|------------------------------------------------------------------------------|------|
| Figure 1: General Character Recognition process                              | 03   |
| Figure 2: Bangla Character Recognition Procedure                             | 03   |
| Figure 3: Scanned Image                                                      | 05   |
| Figure 4: Computer Image                                                     | 05   |
| Figure 5: RGB (Scanned) image                                                | 06   |
| Figure 6: Grayscale image                                                    | 06   |
| Figure 7: Grayscale image with histogram                                     | 07   |
| Figure 8: Binary image with histogram                                        | 07   |
| Figure 9: A Bangla character after sampled                                   | 09   |
| Figure 10: Sampled character representation                                  | 10   |
| Figure 11: vector representation                                             | 10   |
| Figure 12: Kohonen Neural Network design for Bangla Character Recognition    | 12   |
| Figure 13: A Simple Kohonen Neural Network with 2 input and 2 output neurons | 13   |
| Figure 14: Learning Algorithm Flowchart                                      | 17   |

## Chapter 1: Introduction

Optical Character Recognition abbreviated as OCR means that converting some text image into computer editable text format. For example we can say about ASCII code. But in this thesis Unicode is considered as converted text. Lots of recognition systems are available in computer science and also OCR plays a prominent role in computer science. Recognition system works well for simple language like English. It has only 26 character sets. And for standard text there are 52 numbers of characters including capital and small letters. But a complex but organized language like Bangla, OCR system is still in preliminary level. The reasons of its complexities are its character shapes, its top bars and end bars more over it has some modified, voweled and compound characters.

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| Bangla digits       | ০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯                                                        |
| Bangla vowels       | অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ                                                      |
| Modified vowels     | া ি ী ু ূ ্ ে ৈ ো ৌ                                                        |
| Voweled characters  | কা কি কী কু কূ ক্ কৈ কো কৌ                                                 |
| Bangla consonants   | ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র ল শ ষ স হ ঞ্জ ঙ্জ য় |
| Compound characters | ঋ ত দ দ্ব ঞ K e ÿ ã "Q ' "Q ¶ ¼ ¶ ¾ ² Á K Ä É ¼                            |

Table 1: List of Bangla Characters

In this report a new approach is described for Bangla character and some word recognition. Kohonen Neural Network is used for training and recognition procedure which means classification stage. At the beginning grayscale and then BW image conversion takes place for producing binary data. These pre-processing steps are described in chapter 4. After that the image containing Bangla character(s) need to be converted into trainable form by means of processing steps. Processing steps are described in chapter 5. And about Kohonen network and its procedures are described in chapter 6.

## Chapter 2: Character Recognition Procedure

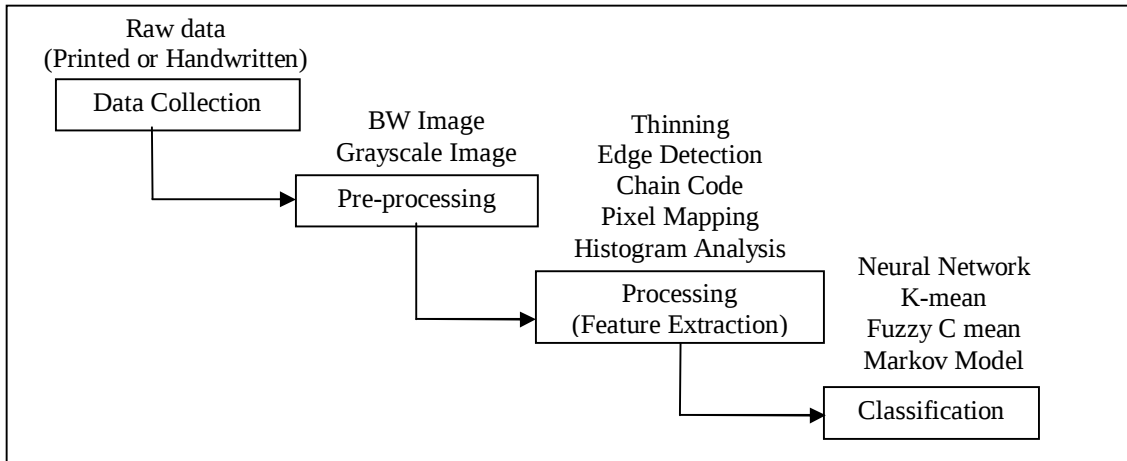
As like all other recognition procedure character recognition is nothing but a recognition process. Several steps are present for a recognition specially character recognition system. Here a simple and general character recognition procedure (figure 1) is described below.

First of all we need a large number of raw data or collected data which will be processed and later trained with the system. It is very important to collect a specific data. Later on we need to compare with similar kind of data. And also we have to think the complexity level of collected data because next steps will be dependent on my data type. It can be scanned documents or hand written documents.

Secondly we have to consider pre-processing stage. Here mainly image processing procedures take place. Like gray scale image conversion, binary image conversion, skew correction. Our processing stage depends on pre-processing stages. So we need to design our pre-processing steps with great care.

Thirdly the processing steps are occurred. Thinning, Edge Detection, Chain Code, Pixel Mapping, Histogram Analysis are some feature of processing stages. This stage basically converts raw data into trainable components.

Finally the training and recognition in short classification stage take place. The pre-processed and processed data is trained by means of taught the system about the incoming data. So later on it can easily recognize an input data.

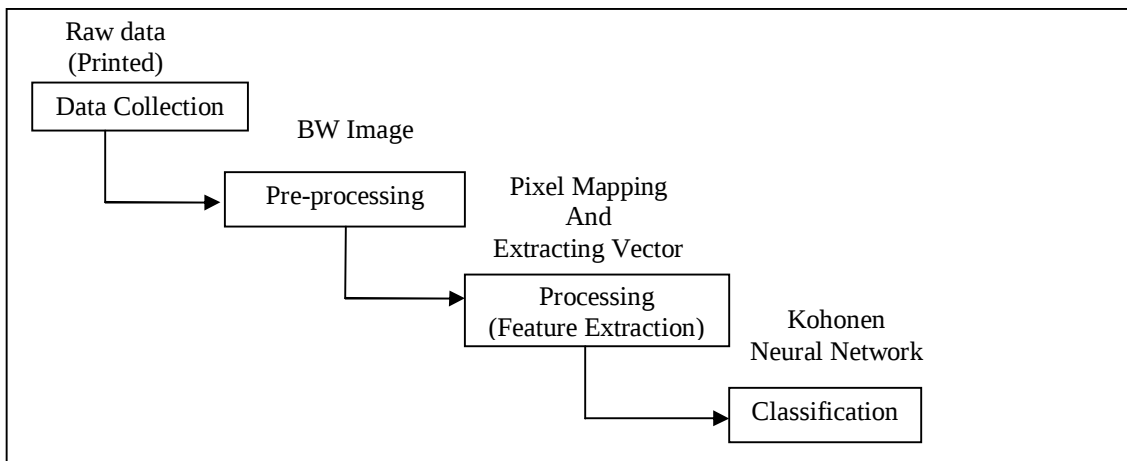


**Figure 1: General Character Recognition process**

## 2.1 Bangla Character Recognition Procedures with Kohonen Network

So far I have described about the character recognition procedure. Now I am describing the procedure used in my character recognition system (figure 2). Steps are described below:

- Printed Bangla character is taken for raw data.
- Printed Bangla character is gray scaled and then converted into BW image in pre-processing stage.
- Pixels are grabbed and mapped into specific area and vector is extracted from the image containing Bangla word or character. This part is considered as processing stage.
- Lastly Kohonen Neural Network is taken as classification stage.



**Figure 2: Bangla Character Recognition Procedure**

### **Chapter 3: Data Collection**

As I said we have to choose our data type with great care. Because we have to develop our system according to our raw data or collected data. Here I am talking about Bangla character recognition so obviously I need Bangla character. But I am also considering some Bangla word as well. But no word level or character level segmentation is considered here rather whole single character image or single word image is taken as raw input. And before entering into the system the image is resized into 250 X 250 pixels to satisfy the procedure. No matter whether it's a character or word contained image. I took both computer image and scanned image for my system to be trained. But no skew correction took place here. So when we need scanned image we have to be careful about the image size and shape.

Here the whole word is taken and trained. Because there are lots of features, irregular shapes and curvatures in Bangla characters as we have seen above. And still no general formula is generated for feature extraction from Bangla character. So rather than extracting characters from a word, the whole word is taken as input data in the first phase of Bangla character recognition system.

One thing is really important here. That is the character size of the image. The character shouldn't be partially present on the image and it shouldn't be too small in size. In my system I am taking above 36 font size for each individual character or word.

## Chapter 4: Image processing

A digital text image that is containing Bangla character is generally an RGB image. The figures below showing two types of image containing digital Bangla character. The character on figure 3 is scanned and resized into 250 X 250 pixels. The same thing for the figure 4 except scanning.



**Figure 3: Scanned Image**



**Figure 4: Computer Image**

### 4.1 RGB to Grayscale Image conversion

In the pre-processing 1<sup>st</sup> stage I am converting the input RGB image into gray scale image. Here I am considering the Othu's algorithm for RGB to gray scale conversion. The algorithm is given below:

1. Count the number of pixel according to color (256 colors) and save it to matrix count.
2. Calculate probability matrix  $P$  of each color,  $P_i = \text{count}_i / \text{sum of count}$ ,  
where  $i = 1, 2, \dots, 256$ .
3. Find matrix omega,  $\text{omega}_i = \text{cumulative sum of } P_i$ ,  
where  $i = 1, 2, \dots, 256$ .

4. Find matrix  $\mu$ ,  $\mu_i = \text{cumulative sum of } P_i * i$ , where  $i = 1, 2, \dots, 256$  and  $\mu_t = \text{cumulative sum of } P_{256} * 256$

5. Calculate matrix  $\sigma_b^2$  where,  
 $\sigma_b^2_i = (\mu_t \times \omega_i - \mu_i)^2 / \omega_i - (1 - \omega_i)$

6. Find the location,  $idx$ , of the maximum value of  $\sigma_b^2$ .  
The maximum may extend over several bins, so average together the locations.

7. If maximum is not a number, meaning that  $\sigma_b^2$  is all not a number, and then threshold is 0.

8. If maximum is a finite number,  $\text{threshold} = (idx - 1) / (256 - 1)$ ;

The figure 5 below is showing an RGB image and figure 6 is showing and grayscale converted image.



Figure 5: RGB (Scanned) image

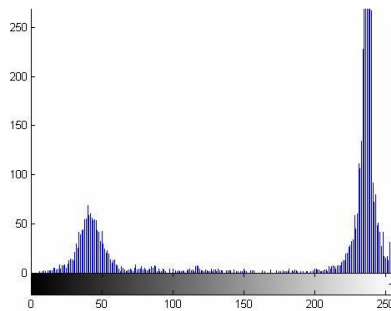


Figure 6: Grayscale image

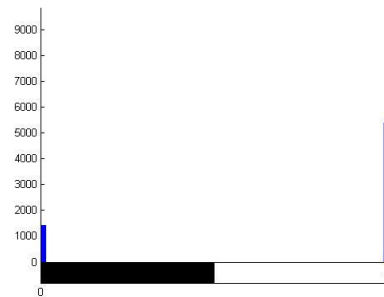
## 4.2 Grayscale to Binary Image conversion

In the pre-processing 2<sup>nd</sup> stage I am converting the gray scale image into binary image. In a grayscale image there are 256 combinations of black and white colors where 0 means pure black and 255 means pure white. This image is converted to binary image by checking whether or not each pixel value is greater than  $255 \cdot \text{level}$  (level, found by Otsu's

Method). If the pixel value is greater than or equal to 255•level then the value is set to 1 i.e. white otherwise 0 i.e. black. The figure 7 is showing a grayscale image with 0-255 level of histogram and figure 8 is showing a BW or binary image with two level of histogram.



**Figure 7: Grayscale image with histogram**



**Figure 8: Binary image with histogram**

## Chapter 5: Feature Extraction

Next and the most important feature of Bangla character recognition is feature extraction. In this system I am considering a few steps for extracting a vector. Our main target is finding a vector from the image. So image is processed and then binary image is created. So we have only 2 types of data on the image. Those are 1 for the white space and 0 for the black space. Now we have to pass the following steps for creating 625 length vector for a particular character or image. Those are:

- a. Pixel grabbing
- b. Finding probability of making square

- c. Mapped to sampled area
- d. Creating vector
- e. Representing character with a model no.

## 5.1 Pixel Grabbing from Image

As we are considering binary image and we also fixed the image size, so we can easily get 250 X 250 pixels from a particular image containing Bangla character or word. One thing is clear that we can grab and separate only character portion from the digital image. In specific, we took a Bangla character contained image. And obviously it's a binary image. As we specified that the pixel containing value 1 is a white spot and 0 for a black one, so naturally the 0 portioned spots are the original character.

## 5.2 Finding probability of making square

Now we are going to sample the entire image into a specified portion so that we can get the vector easily. We specified an area of 25 X 25 pixels. For this we need to convert the 250 X 250 image into the 25 X 25 area. So for each sampled area we need to take 10 X 10 pixels from binary image.

We can give a short example for that. Table 2 is the original binary image of 25 X 15 pixels. We need to sampled it 5 X 3 pixels area. So, for each area we will consider 5 X 5 pixel from the binary image. Table 3 will show how pixels are classified for finding the probability of making square.

|                              |       |       |       |       |       |
|------------------------------|-------|-------|-------|-------|-------|
| 00000110000010000000000000   | 00000 | 11000 | 00100 | 00000 | 00000 |
| 0000000000000000000111000010 | 00000 | 00000 | 00000 | 01110 | 00010 |
| 0100000111001000111000100    | 01000 | 00111 | 00100 | 01110 | 00100 |
| 0010011111000100111000000    | 00100 | 11111 | 00010 | 01110 | 00000 |
| 00000001110000100000000000   | 00000 | 00111 | 00001 | 00000 | 00000 |
| 0111000000110001100100110    |       |       |       |       |       |

|                           |       |       |       |       |       |
|---------------------------|-------|-------|-------|-------|-------|
| 010000000011111001100000  | 01110 | 00000 | 11000 | 11001 | 00110 |
| 0011100000110001100100000 | 01000 | 00000 | 01111 | 10011 | 00000 |
| 0111100000001111001101110 | 00111 | 00000 | 11000 | 11001 | 00000 |
| 0001100000001111100100001 | 01111 | 00000 | 00111 | 10011 | 01110 |
| 0000011000000000000011000 | 00011 | 00000 | 00111 | 11001 | 00001 |
| 0000000011000000011100000 |       |       |       |       |       |
| 0000000000000000000000110 | 00000 | 11000 | 00000 | 00000 | 11000 |
| 0000011111000000011100000 | 00000 | 00011 | 00000 | 00111 | 00000 |
| 0000011111000000000000001 | 00000 | 00000 | 00000 | 00000 | 00110 |
|                           | 00000 | 11111 | 00000 | 00111 | 00000 |
|                           | 00000 | 11111 | 00000 | 00000 | 00001 |

**Table 2: Initial pixel data from image**

**Table 3: Separating the pixels**

### 5.3 Mapped to sampled area

We can recall the previous example from table 4 Now the same sample pixel from binary image after separating is showing in table 5. Now we will find out for each 5 X 5 pixel from the separated pixel portion and give an unique number for each separated pixel class. And this number will be equal to the 5 X 3 sampled areas. Now we need no consider whether 5 X 5 pixels will make a black area or square or a white area or square. We will take the priority of 0s or 1s from 5 X 5 pixels. And from there we can say, if the 0s get the priority from 5X5 in  $i^{\text{th}}$  location then we will make a black square on  $i^{\text{th}}$  position of sample area. Table 4 is having a unique number of 5 X 5 separated pixels and table 5 in covering black or white depending on the probabilistic manner.

|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| 00000 | 11000 | 00100 | 00000 | 00000 |
| 00000 | 00000 | 00000 | 01110 | 00010 |
| 01000 | 00111 | 00100 | 01110 | 00100 |
| 00100 | 11111 | 00010 | 01110 | 00000 |
| 00000 | 00111 | 00001 | 00000 | 00000 |
| 1     | 2     | 3     | 4     | 5     |
| 01110 | 00000 | 11000 | 11001 | 00110 |
| 01000 | 00000 | 01111 | 10011 | 00000 |
| 00111 | 00000 | 11000 | 11001 | 00000 |
| 01111 | 00000 | 00111 | 10011 | 01110 |
| 00011 | 00000 | 00111 | 11001 | 00001 |
| 6     | 7     | 8     | 9     | 10    |
| 00000 | 11000 | 00000 | 00000 | 11000 |
| 00000 | 00011 | 00000 | 00111 | 00000 |
| 00000 | 00000 | 00000 | 00000 | 00110 |
| 00000 | 11111 | 00000 | 00111 | 00000 |

|    |    |    |    |    |
|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  |
| 6  | 7  | 8  | 9  | 10 |
| 11 | 12 | 13 | 14 | 15 |



Now we will collect each row, combine together and it will make a vector. Vector for figure 10 is given in the figure 11.

[illegible]

**Figure 11: vector representation**

## 5.5: Representing a character with a model number

One thing we need to mention here. That is, we gave a 4 digit numerical number as a model for each vector and also the corresponding Bangla word or character from that particular model. This is because Bangla character is unique length. But we are also considering Bangla word which has irregular length. When we need to train, we will train it with its model no. and model number knows its corresponding Bangla character. So in shorts we can say that, a particular model has a unique vector with the length of 625 characters of 1s and 0s and a unique Bangla character.

We did lots of activities in pre-processing stages and as well as in processing stage. The main idea is to make it simple and acceptable for Kohonen Neural Network. The Kohonen neural network contains no hidden layer. This network architecture is named after its creator, Tuevo Kohonen. The Kohonen neural network differs from the feedforward back propagation neural network in several important ways. In this chapter we will examine the Kohonen neural network and see how it is implemented.

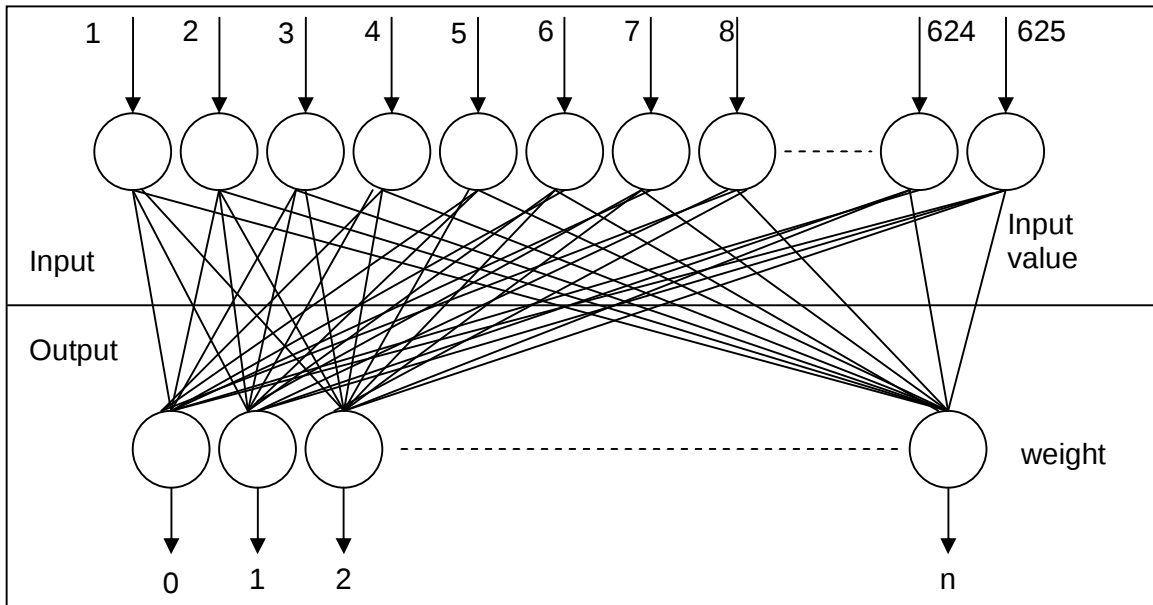
## **6.1 Introduction to Kohonen Network**

The Kohonen neural network differs considerably from the feedforward back propagation neural network. The Kohonen neural network differs both in how it is trained and how it recalls a pattern. The Kohonen neural network does not use any sort of activation function. Further, the Kohonen neural network does not use any sort of a bias weight.

Output from the Kohonen neural network does not consist of the output of several neurons. When a pattern is presented to a Kohonen network one of the output neurons is selected as a "winner". This "winning" neuron is the output from the Kohonen network. Often these "winning" neurons represent groups in the data that is presented to the Kohonen network. For example, in our system we consider 10 Bangla digits, 11 vowels, 36 consonants, and 42 words in total 100 models. The most significant difference between the Kohonen neural network and the feed forward back propagation neural network is that the Kohonen network trained in an unsupervised mode. This means that the Kohonen network is presented with data, but the correct output that corresponds to that data is not specified. Using the Kohonen network this data can be classified into groups. We will begin our review of the Kohonen network by examining the training process.

As our vector length is 625 so our input layer has 625 neurons. But in our output layer the number of neuron depends on the number of character trained with the network. As we

take 625 as input and  $n$  for output character, we can draw our suitable Kohonen Neural Network as figure 12.



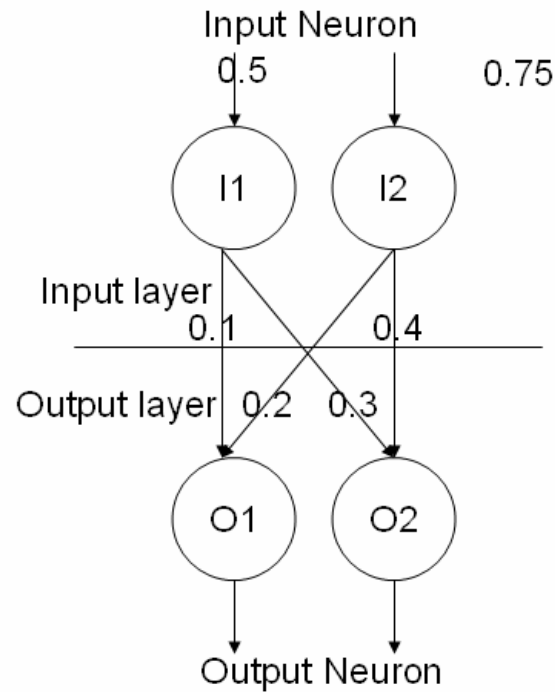
**Figure 12: Kohonen Neural Network design for Bangla Character Recognition**

## 6.2 The structure of Kohonen Network

The Kohonen neural network contains only an input and output layer of neurons. There is no hidden layer in a Kohonen neural network. First we will examine the input and output to a Kohonen neural network.

The input to a Kohonen neural network is given to the neural network using the input neurons. These input neurons are each given the floating point numbers that make up the input pattern to the network. A Kohonen neural network requires that these inputs be normalized to the range between -1 and 1. Presenting an input pattern to the network will cause a reaction from the output neurons. In a Kohonen neural network only one of the output neurons actually produces a value. Additionally, this single value is either true or false. When the pattern is presented to the Kohonen neural network, one single output neuron is chosen as the output neuron. Therefore, the output from the Kohonen neural

network is usually the index of the neuron that fired. The structure of a typical Kohonen neural network is shown in Figure 13.



**Figure 13: A Simple Kohonen Neural Network with 2 input and 2 output neurons**

### 6.3 Sample Input to Kohonen Network

As we understand the structure of the Kohonen neural network we will examine how the network processes information. To examine this process we will step through the calculation process. For this example we will consider a very simple Kohonen neural network. This network will have only two input and two output neurons. The input given to the two input neurons is shown in Table 6.

|                     |      |
|---------------------|------|
| Input Neuron 1 (I1) | 0.5  |
| Input Neuron 2 (I2) | 0.75 |

**Table 6: Sample Inputs to a Kohonen Neural Network**

We must also know the connection weights between the neurons. These connection weights are given in Table 7.

|        |     |
|--------|-----|
| I1->O1 | 0.1 |
| I2->O1 | 0.2 |
| I1->O2 | 0.3 |
| I2->O2 | 0.4 |

**Table 7: Connection weights in the sample Kohonen neural network**

Using these values we will now examine which neuron would win and produce output. We will begin by normalizing the input.

### **6.3.1 Normalizing the input**

The requirements that the Kohonen neural network places on its input data are one of the most severe limitations of the Kohonen neural network. Input to the Kohonen neural network should be between the values -1 and 1. In addition, each of the inputs should fully use the range. If one, or more, of the input neurons were to use only the numbers between 0 and 1, the performance of the neural network would suffer.

To normalize the input we must first calculate the "vector length" of the input data, or vector. This is done by summing the squares of the input vector. In this case it would be.  $(0.5 * 0.5) + (0.75 * 0.75)$ . This would result in a "vector length" of 0.8125. If the length becomes too small, say less than the length is set to that same arbitrarily small value. In this case the "vector length" is a sufficiently large number. Using this length we can now determine the normalization factor. The normalization factor is the reciprocal of the square root of the length. For our value the normalization factor is calculated as follows,

$\frac{1}{\sqrt{0.8125}}$  and this results in a normalization factor of 1.1094. This normalization process

will be used in the next step where the output layer is calculated.

### 6.3.2 Calculating each neuron's output

To calculate the output the input vector and neuron connection weights must both be considered. First the "dot product" of the input neurons and their connection weights must be calculated. To calculate the dot product between two vectors you must multiply each of the elements in the two vectors. We will now examine how this is done.

The Kohonen algorithm specifies that we must take the dot product of the input vector and the weights between the input neurons and the output neurons. The result of this is as follows.

$$\begin{bmatrix} 0.5 & 0.75 \end{bmatrix} \bullet \begin{bmatrix} 0.1 & 0.2 \end{bmatrix} = (0.5 * 0.75) + (0.1 * 0.2) .$$

As we can see from the above calculation the dot product would be 0.395. This calculation will be performed for the first output neuron. This calculation will have to be done for each of the output neurons. Through this example we will only examine the calculations for the first output neuron. The calculations necessary for the second output neuron are calculated in the same way.

This output must now be normalized by multiplying it by the normalization factor that was determined in the previous step. We must now multiply the dot product of 0.395 by the normalization factor of 1.1094. This results in an output of 0.438213. Now that the output has been calculated and normalized it must be mapped to a bipolar number.

### 6.3.3 Mapping to bipolar

In the bipolar system the binary zero maps to -1 and the binary remains a 1. Because the input to the neural network normalized to this range we must perform a similar normalization to the output of the neurons. To make this mapping we add one and divide the result in half. For the output of 0.438213 this would result in a final output of 0.7191065.

The value 0.7191065 is the output of the first neuron. This value will be compared with the outputs of the other neuron. By comparing these values we can determine a "winning" neuron.

#### **6.3.4 Choosing a winner**

We have seen how to calculate the value for the first output neuron. If we are to determine a winning output neuron we must also calculate the value for the second output neuron. We will now quickly review the process to calculate the second neuron. For a more detailed description you should refer to the previous section.

The second output neuron will use exactly the same normalization factor as was used to calculate the first output neuron. As you recall from the previous section the normalization factor is 1.1094. If we apply the dot product for the weights of the second output neuron and the input vector we get a value of 0.45. This value is multiplied by the normalization factor of 1.1094 to give the value of 0.0465948. We can now calculate the final output for neuron 2 by converting the output of 0.0465948 to bipolar yields 0.49923.

As we can see we now have an output value for each of the neurons. The first neuron has an output value of 0.7191065 and the second neuron has an output value of 0.49923. To choose the winning neuron we choose the output that has the largest output value. In this case the winning neuron is the first output neuron with an output of 0.7191065, which beats neuron two's output of 0.49923.

We have now seen how the output of the Kohonen neural network was derived. As we can see the weights between the input and output neurons determine this output.

#### **6.4 Kohonen Network learning procedure**

The training process for the Kohonen neural network is competitive. For each training set one neuron will "win". This winning neuron will have its weight adjusted so that it will react even more strongly to the input the next time. As different neurons win for different patterns, their ability to recognize that particular pattern will be increased.

We will first examine the overall process involving training the Kohonen neural network.

## 6.5 Learning algorithm flowchart

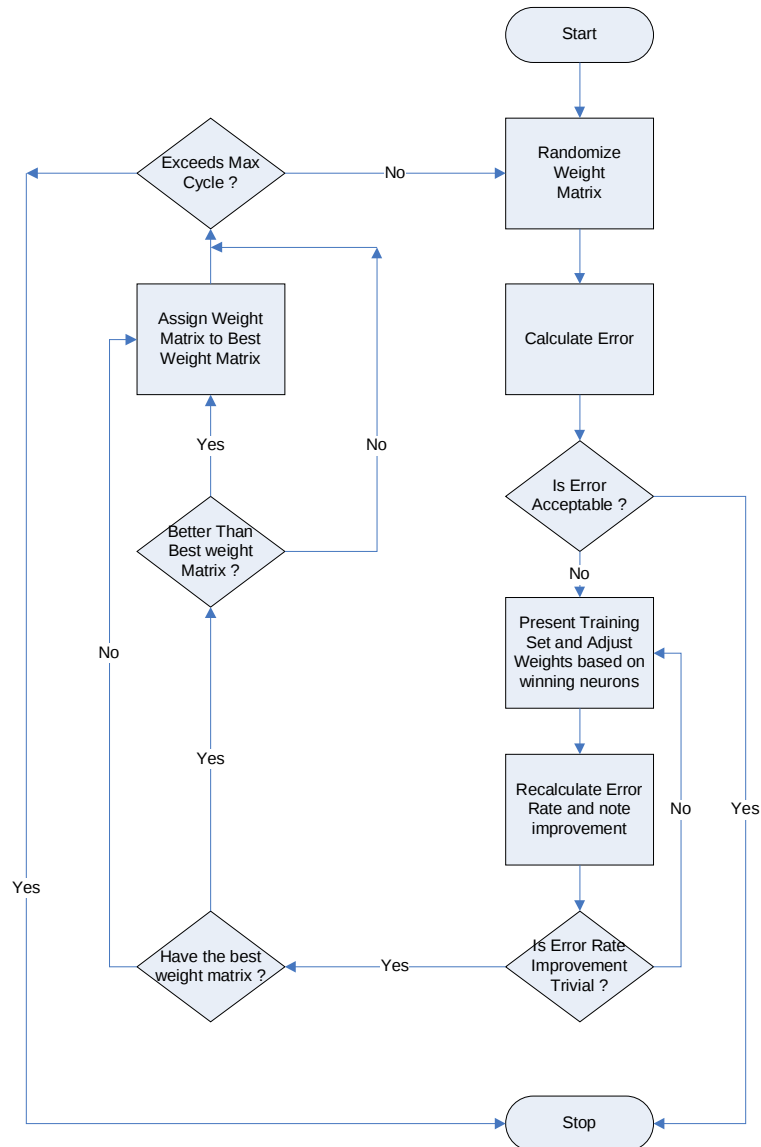


Figure 14: Learning Algorithm Flowchart

### 6.5.1 Learning rate

The learning rate is a constant that will be used by the learning algorithm. The learning rate must be a positive number less than 1. Typically the learning rate is a number such as .4 or .5. In the following section the learning rate will be specified by the symbol alpha.

Generally setting the learning rate to a larger value will cause the training to progress faster. Though setting the learning rate to too large a number could cause the network to never converge. This is because the oscillations of the weight vectors will be too great for the classification patterns to ever emerge. Another technique is to start with a relatively high learning rate and decrease this rate as training progresses. This allows initial rapid training of the neural network that will be "fine tuned" as training progresses. The learning rate is just a variable that is used as part of the algorithm used to adjust the weights of the neurons.

### 6.5.2 Adjusting weight

The entire memory of the Kohonen neural network is stored inside of the weighted connections between the input and output layer. The weights are adjusted in each epoch. An epoch occurs when training data is presented to the Kohonen neural network and the weights are adjusted based on the results of this item of training data. The adjustments to the weights should produce a network that will yield more favorable results the next time the same training data is presented. Epochs continue as more and more data is presented to the network and the weights are adjusted.

Eventually the return on these weight adjustments will diminish to the point that it is no longer valuable to continue with this particular set of weights. When this happens the entire weight matrix is reset to new random values. This forms a new cycle. The final weight matrix that will be used will be the best weight matrix determined from each of the cycles. We will now examine how these weights are transformed.

The original method for calculating the changes to weights, which was proposed by Kohonen, is often called the additive method. This method uses the following equation,

$w^{t+1} = \frac{w^t + ax}{\|w^t + ax\|}$ . The variable x is the training vector that was presented to the network.

The variable  $w^t$  is the weight of the winning neuron, and the variable  $w^{t+1}$  is the new weight. The double vertical bars represent the vector length.

The additive method generally works well for Kohonen neural networks. Though in cases where the additive method shows excessive instability, and fails to converge, an alternate method can be used. This method is called the subtractive method. The subtractive method uses the following equations.  $e = x - w^t$  and  $w^{t+1} = w^t + ae$ . These two equations show you the basic transformation that will occur on the weights of the network.

### **6.5.3 Calculating the errors**

Before we can understand how to calculate the error for Kohonen neural network must first understand what the error means. The Kohonen neural network is trained in an unsupervised fashion so the definition of the error is somewhat different involving the normally think of as an error.

The purpose of the Kohonen neural network is to classify the input into several sets. The error for the Kohonen neural network, therefore, must be able to measure how well the network is classifying these items.

### **6.6 Recognition with Kohonen network**

So for a given pattern we can easily find out the vector and can send it through the Kohonen Neural Network. And for that particular pattern any one of the neuron will be fired. As for all input pattern the weight is normalized so the input pattern will be calculated with the normalized weight. As a result the fired neuron is the best answer for that particular input pattern.

## Chapter 7: Functional Output

According to my system stated above, I study some accuracy level for different input data. It is important because it is an identifier of feasibility and efficiency. I consider both accuracy rates and also some drawbacks respected to the described Bangla Character Recognition process.

### 7.1 Accuracy Rates

Table 8 shows some accuracy level for different input data. The data is considered as trained characters or words, untrained similar fonts (solaiman lipi is the trained font and similar font is dhanshiri), scanned documents are of solaiman lipi fonts and lastly irregular shape fonts (some big or small fonts).

| Character/words         | Accuracy rate |
|-------------------------|---------------|
| Trained                 | 100%          |
| Untrained similar fonts | 99%           |
| Scanned docs.           | 99%           |
| Irregular font size     | 98%           |

Table 8: Accuracy rates corresponding to different sectors

### 7.2 Drawbacks

As I said we are at the preliminary level of the Bangla Character Recognition so the main drawback we can consider is we need to modify and make it more accurate. Again like all other Neural Network training time increase with the increase in number of characters or words in Kohonen Neural Network. Besides I defined fixed picture size 250 X 250. So it will not work for different image documents. So it needs to be more generalized. Finally the system can't work for small fonts. This is because; we need to grab the pixels at first from the original image documents. Then we need to map it. But in the case of small

Bangla fonted image it can't grab the pixel from original documents. So it creates problem for recognizing small fonts.

## **Chapter 8: Conclusion and future work**

### **8.1 Conclusion**

This report tries to emphasis on a way or method of Bangla character recognition in a simplest manner. But there are lots of ways to implement it. And that could be more efficient than Kohonen Neural Network. But successfully the research comes to an end. And at the conclusion I can quote that there is a huge area to research on Bangla Character and its recognition procedure.

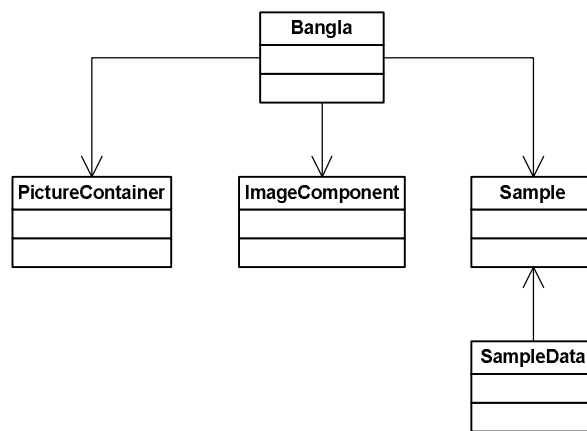
### **8.2 Future work**

I stated above that there is a vast area of research on Bangla Character Recognition. Complex character based language like Bangla needs deep research to meet its goal. Number of input neuron in my system is quite high. It can be reduced by segmenting character. But inaccuracy is palpable in segmentation of Bangla character. So efficient system is still far away. We can think about the basic characteristics of Bangla character and those characters can be grouped and then segmented. Thus the number of input neuron can be reduced. And also Kohonen Network is used based on the feature of each Bangla character.

## Glossary

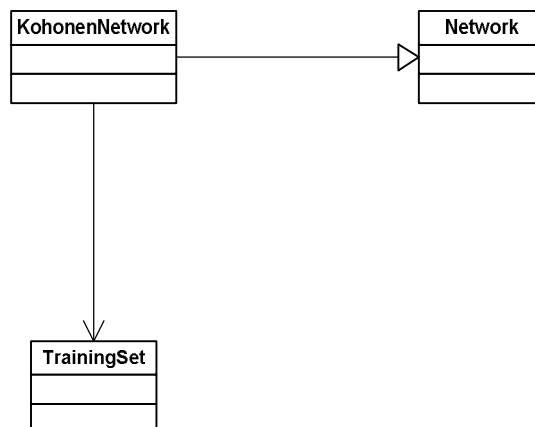
### UML Diagram for Feature Extraction

Here the main class is Bangla which contains GUI of the system. It usually helps to take image as input. Picture container actually contains the image and image component helps to draw into Bangla GUI pixel by pixel. Again the sample class used to sample the input image. The class SampleData split the input image pixel by pixel, normalize and modify it and returns the sampled data.



### UML Diagram for Kohonen Network

This is a simple diagram from Kohonen Network. Network is the actual design of the Kohonen Network. Kohonen Network actually uses the attributes of Network class. And the training methodology is stands on the TrainingSet class. Winning selection, normalizing, error detection, analyzing is the part of TrainingSet class.



## **References:**

1. Martin T. Hagan, Howard B. Demuth and Mark Beale, "Neural Network Design", PWS Publishing Company, 1995.
2. Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", second edition, Pearson Education, 2003
3. Jeff Heaton, "Introduction to Neural Network in Java", HR publication, 2002
4. N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, 1979.
5. Anil K. Jain and Torfinn Taxt, "Feature Extraction Methods for Character Recognition- a survey", Michigan State University, 1995
6. Nawsher Ahmed Noor and S. M. Murtoza Habib, ""Bangla Optical Character Recognition", BRAC University, 2005
7. Juan-Carlos Perez, Enrique Vidal and Lourdes Sanchez, "Simple and Effective Feature Extraction for Optical Character Recognition"
8. Zhindong Lu, Issam Bazzi and Richard Schwartz, "A- Robust, Language Independent OCR System"
9. Jalal Uddin Mahmud, Md. Feroz Raihan and Chowdhury Mofizur Rahman, "A Complete OCR System for continuous Bengali Character"