NeurOCR: A Neural Network based Approach to Optical Character Recognition (OCR) Systems

Harsh Thakkar

Computer Engineering Department, S. V. National Institute of Technology,

Surat – 395 007, Gujarat, India

E-mail: harsh9t@gmail.com

ABSTRACT

The recognition of optical characters is known to be one of the earliest applications of Artificial Neural Networks, which partially emulate human thinking in the domain of Artificial Intelligence. The current paper focuses on the use of neural network in order to mitigate the problems of digital handwriting recognition by using Self-Organizing Map model for fast processing and less processing power consumption keeping its deployment on PDA in mind. The document is expected to serve as a resource for learners and amateur investigators in pattern recognition, neural networking and related disciplines. Technology Used is C#, .Net 3.5 Framework and Heaton Research Neural Network API has been used exhaustively for deploying the calibrated Self-Organising Model (SOM) approach. Characters are input when a user draws on a high-resolution box. Unfortunately, this resolution is too high to be directly presented to the neural network. To resolve this problem, we use the techniques of cropping and down sampling to transform the image into a second image that has a much lower resolution. It was observed that the use of Neural Networks for Recognising characters proves out to be more efficient and robust compared to other hard computing techniques. The Self-Organising Model network is proved to be the most prominent competitor for such an application providing precise outputs for recognised characters.

Index Terms: Neural Networks, Digital Handwriting Recognition, Artificial Intelligencer, Pattern Recognition, Self-Organizing Map.

1. INTRODUCTION

Inspired by human brain, and its very way of thinking, the domain of Artificial intelligence and Neural Networks gains its motivation. In this modern era, Artificial Intelligence and associated realms are perceived to be of prodigious prominence[1]. Computation is an un-evitable part of living, and so is called synonym to existence today. Internet, a place where majority time share of life is spent consists of very large volume information that one wishes to access in printed or machine readable format[2]. Computation leads to programing, which till last one or two decades was majorly consisted of single approach i.e. Hard Computing. But Soft Computing[2] approach

has also been developed based on the Neural Networks. The forest of computer science has indeed enormous potential to offer. Soft Computing, unlike the previous conventional Hard Computing[1] approach, is called the new way of thinking. It offers features like tolerance of imprecision, vagueness in input data, and fractional truth. Soft Computing consists of Fuzzy Logic, Genetic Algorithms, Swarm Intelligence and Neural Networks.

The field of Optical Character Recognition has its roots from early 1930s. Originated in Germany as a patent by Gustav Tauschek, an Austrian pioneer of Information technology. Abbreviated to OCR, Optical Character Recognition is a technique of translating handwritten, typewritten or printed text characters to a machine-encoded text[2]. It simple means to read a handwritten character from a piece of paper and recognize what is written. It also parallel got recognition as the subdomain of the pattern recognition of the Image Processing Field. Human brain generally tries to find some kind of relation, predominantly in graphic form, in order to be able to remember it and recognize the same later. Thus, it tends to produce or find patterns in handwritten characters. This led as the key inspiration in the development of the OCR systems and the dominant field of Pattern Recognition. Based on the lines and curves of the characters of various languages, a particular OCR can be designed to recognize them.

This paper discusses about the use of Soft Computing approach instead of Hard Computing approach that is being used for the development of OCR systems, which leads to erroneous outputs in presence of uncertainty in input patterns. Moreover these systems are very less tolerant to faults and could learn from previous inputs. At any particular given moment of time there are various factors affecting the characters being written by a human being, which causes vagueness in input leading to dissatisfying results[3]. Thus, applying neural networks based approach eradicates these drawbacks. OCR systems have been commercialized for example Vehicle Number Plate Identification[7], OmniPage, WordScan, ABBYY FineReader, TypeReader, etc[2]. Here a neural network based computing approach is chosen to develop a better Optical Character recognizer[1,2].

2. SYSTEM OVERVIEW AND CONSIDERATIONS

2.1 SYSTEM OVERVIEW

The deployed systems possesses the capability recognize characters based n the previous training and tuning, which also includes feature of training neural network. The GUI is simple enough for user to utilize it and not to get confused. By drawing character in the given image area, user can train the network or user can recognize the drawn character using trained network. User can also edit the neural network training by deleting or adding, described later in this document. The trained network can also be saved to a SAMPLE.DAT file for future use. Currently only one by one character is recognized, later giving scope to expansion to the work.

2.2 DESIGN CONSIDERATIONS

The application has been developed on .NET framework. The primary supported operating system will be Microsoft Windows 7. As well as it will also perform on Microsoft Vista/XP SP2. The minimum system requirements other than the software platform are minimal. The API used in development of the application is Heaton-Research SOM API[4,5] and various neural Network libraries for the .net Framework.

3. NEURAL NETWORK ARCHITECTURE

There are various kinds of models of Neural Networks that can be chosen to approach a particular problem. The selection of the type of neural network to be deployed depends on the time constraint of responsiveness required from the program. In OCR systems as numerous characters have to be recognized in less possible time, the model selected here is Self Organizing Model or also known as Self Organizing Feature Model (SOFM). SOM follows an unsupervised[6] learning method to produce a low-dimensional (basically 2D) discrete representation of the input space called Maps. These maps are distinct from other artificial neural networks in the way they parameterize a neighboring function to preserve the topological properties of the input space.

This feature is responsible for making SOM be able to visualize low dimensional view of high dimensional data, similar to multi-dimensional scaling. SOM is also named the Kohonen

network after its inventor Professor Teuvo Kohonen, and the corresponding maps are called Kohonen maps. The schematic diagram of a SOM Neural Network is detailed in the Fig.1.

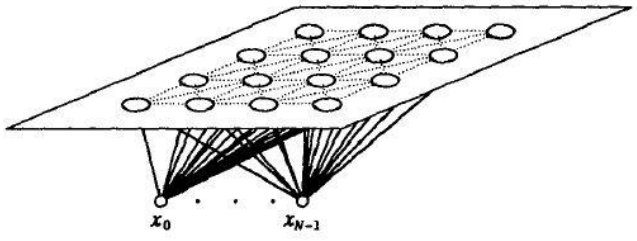

Fig. 1- A Typical Kohonen network, in which the output knits are ordered in a two-dimensional array[4, 5].

The main advantages of the SOM Neural Networks are that they are very easy to build and can be trained very rapidly, thus suits OCR application[2]. The output of the SOM networks is not composed of the result of several neurons, instead is decided by the concept of "Winning Neuron". When an input pattern is presented to the SOM network, one of the output neuron is selected as the output neuron, i.e. "winning neuron". A twenty six (26) neuron network has been used to individually recognize all the twenty six characters of the English alphabet in this study.

The Kohonen Network is distinct from other networks in the way it is trained and the way it recalls the patterns. It doesn't use an activation function instead and weight update vector Wv(t). The equation (3.1)and (3.2) represents a Vector update formula, both additive and subtractive, used in the Neural network.

Eq. (3.1)
$$W^{t+1} = \frac{W^t + \alpha x}{length(W^t + \alpha x)}$$

Eq.(3.2)
$$e = x - w^t$$
$$w^{t+1} = w^t + \alpha e$$

x = is the training vector that was presented to the network; $\mathbf{W^t}$ = weight of the winning neuron.

The additive vector update formula works very well for most of the SOM maps, but however if it shows excessive instability, then a subtractive version of the vector update formula(Eq.3.2) can also be used. The self-organizing map places strict limitations on the input it receives. Input to the self-organizing map must be between the values of –1 and 1. In addition, each of the input neurons must use the full range. If one or more of the input neurons were to only accept the numbers between 0 and 1, the performance of the neural network would suffer.

- Multiplicative[5] Normalization[7]
- Z-Axis Normalization[5]

a) Multiplicative Normalization

To perform multiplicative normalization, we must first calculate the vector length of the input data, or vector. This is done by summing the squares of the input vector and then taking the square root of this number, as shown in Eq. (3.3)

Eq. (3.3) $$F = \frac{1}{\sqrt{\sum_{i=0}^{n-1} X_i^2}}$$

b) Z-Axis Normalization

Unlike the multiplicative algorithm for normalization, the z-axis normalization algorithm does not depend upon the actual data itself; instead the raw data is multiplied by a constant. To calculate the normalization factor using z-axis normalization, Eq. (3.4) is used.

Eq. (3.4) $$F = \frac{1}{\sqrt{n}}$$

As can be seen in the above equation, the normalization factor is only dependent upon the size of the input, denoted by the variable **n**. This preserves absolute magnitude information. However,

we do not want to disregard the actual inputs completely. Thus, a synthetic input is created, based on the input values. The synthetic input is calculated using (Eq.3.5)[5]

Eq. (3.5) $$s = f\sqrt{n - l^2}$$

Where, variable **n** represents the input size. The variable **f** is the normalization factor. The variable **l** is the vector length. The synthetic input will be added to the input vector that was presented to the neural network.

The Fig. 2 shows the control flow diagram of the layers in the SOM neural Network. It consists of the normalization of the user input, and the structure of control flow between the two layers of the neurons, followed by the final output decided by the pattern matched on basis of a winning neutron.

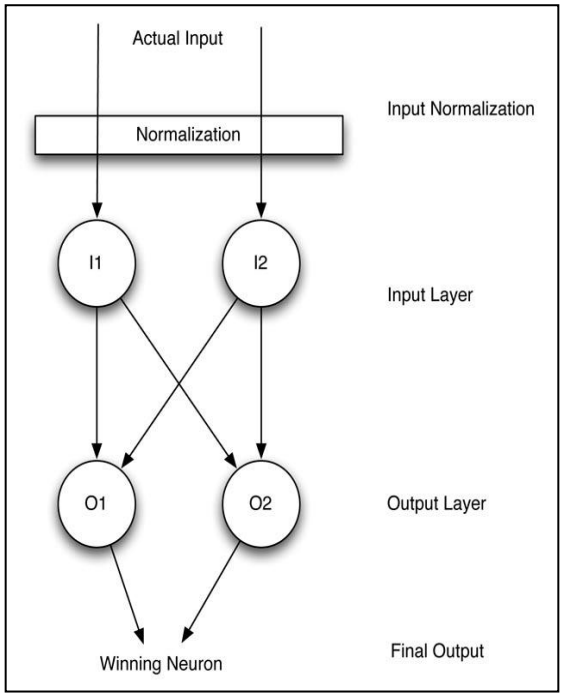

Fig.2- SOM Neural Network Control Flow diagram

Control flow in the SOM Neural Network

Neuron in its corresponding layer

4. SYSTEM WORKING AND USER INTERACTION

4.1 SYSTEM WORKING

The NeurOCR application system works on these four basic modules

- Character Fetching
- Cropping & Downsampling
- Network Loading & Training
- Recognizing encouraged

The flowchart below shown in the fig.3 depicts the system working.

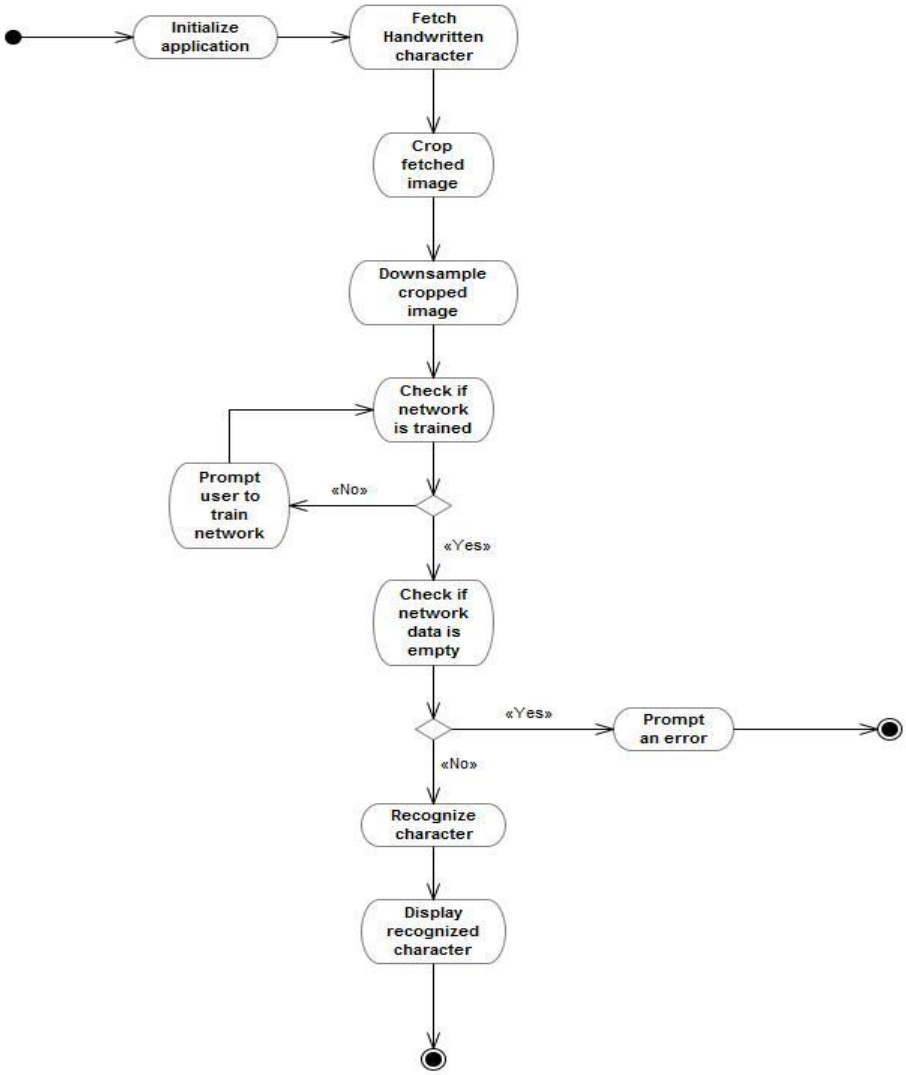

Fig. 3- NeurOCR System Workflow

In the Character Fetching phase, the handwritten is stored on to a bitmap image format[7], in he form of an array of pixels. Since the user is more likely to draw on a random area of the given input space, and as the drawn character is of high resolution, it has to be first cropped, wherein the white space is removed and the input is fitted to the boundaries, and the it is sampled down to a low resolution format for further processing. The Neural Network has to be trained first in order for it to be able to recognize a character. Therefore, a sample data file is loaded in to the network, that consists of some basic input patters of the English alphabet. Now, the based on the patterns trained by the network it is has to be tuned. Tuning refers to adjusting the parameters while iteratively recognizing a given character to rectify the errors. This error range lies from 0.006 to worst of 0.003 percent depending on the number of character patterns trained to the neural network. More training leads to less errors, and has gone down to a level of 0.0001 percent also. The training data provided in the "sample.dat" file can be loaded and as well as other patterns can be added to the sample file for future use. Hence, the NeurOCR application has proved results upto average error performance as shown in (Eq. 3.3),

(Eq. 4.1) $\Delta x \leq 0.003\%$

4.2 USER INTERACTION

The screenshots of the NeurOCR are shown (Fig.5.1 to 5.2)

These depict the NeurOCR Application in action, taking the user inputted written characters and thereby being trained and tuned first, thereafter recognizing the given handwritten inputs to the English characters.

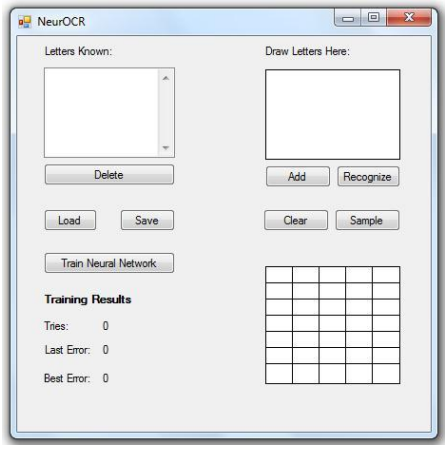

Fig. 4.1: NeurOCR Main Frame GUI

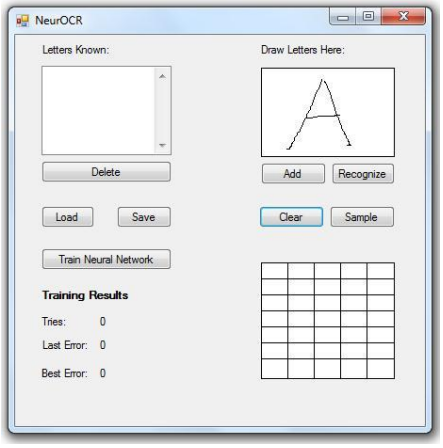

Fig. 4.2: Handwritten Input character by user

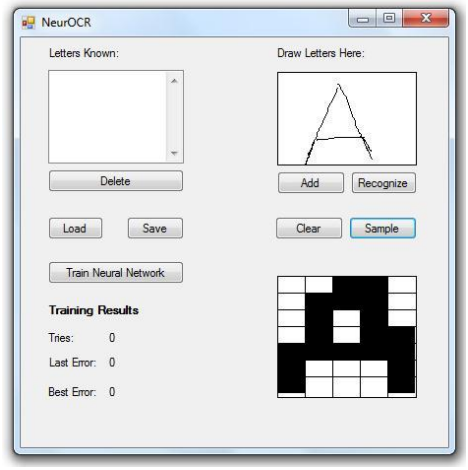

Fig. 4.3 Downsampling of the character inputted by user after white space removal (cropping + scaling) from the image

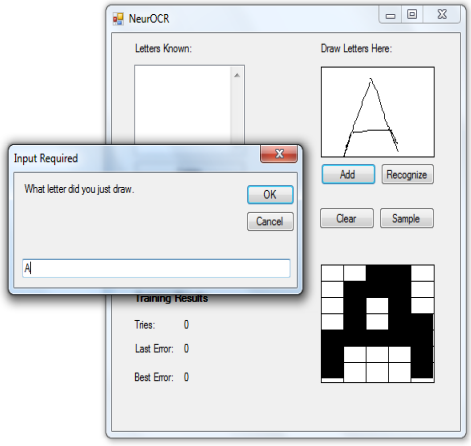

Fig. 4.4 Training the NeurOCR system by providing initial data sets; i.e. adding characters to the training set

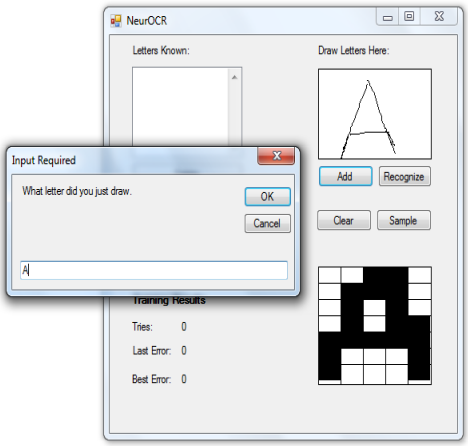

Fig. 4.5 Loading of the initial Training set, i.e. SAMPLE.DAT file complete (System is trained & ready for recognition).

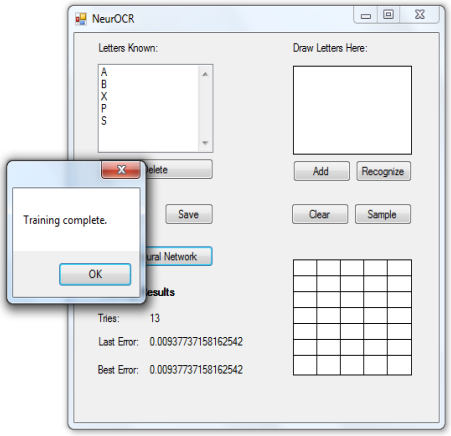

Fig 4.6 Training with sample data complete.

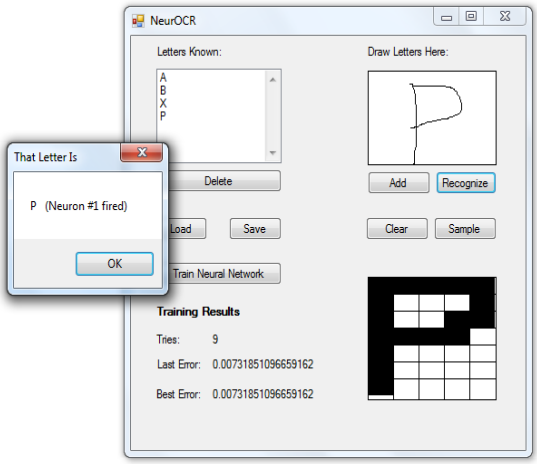

Fig. 4.7 Handwritten character of the user recognized.

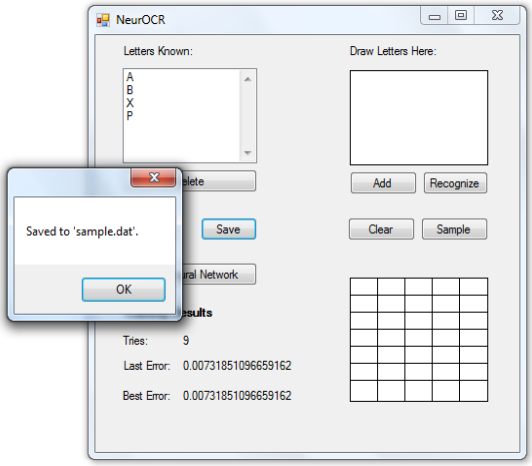

Fig 4.8 saving the newly structured training set to the SAMPLE.DAT file

5. ADVANTAGES AND LIMITATIONS

5.1 ADVANTAGES

The neural system has some direct advantages, over the traditional or conventional hard computing based approach, that become apparent at this stage:

1. The method is highly adaptive; recognition is tolerant to minor errors and changes in patterns.

2. The knowledge base of the system can be modified by teaching it newer characters or teaching different variants of earlier characters. (Can be easily trained & tuned).

3. The system is highly general and is invariant to size and aspect ratio (i.e. Robust performance; gracefull degradation).

4. In future, the system can be made user specific: User-profiles of characters can be maintained, and the system can be made to recognize them as per the orientation of the user.

5.2 LIMTATIONS

The program presented here is only capable of recognizing individual letters, one at a time. In addition, the sample data provided only includes support for the uppercase letters of the Latin alphabet. There is nothing in this program that would prevent you from using both upper and lowercase characters, as well as digits. If you train the program for two sets of 26 letters each and 10 digits, the program will require 62 training sets.

You can quickly run into problems with such a scenario. The program will have a very hard time differentiating between a lowercase “o”, an uppercase “O, and the digit zero (0). The problem of discerning between them cannot be handled by the neural network. Instead, you will have to examine the context in which the letters and digits appear.

Many layers of complexity will be added if the program is expanded to process an entire page of writing at one time. Even if the page is only text, it will be necessary for the program to

determine where each line begins and ends. Additionally, spaces between letters will need to be located so that the individual characters can be fed to the self-organizing map for processing.

If the image being scanned is not pure text, then the job becomes even more complex. It will be necessary for the program to scan around the borders of the text and graphics. Some lines may be in different fonts, and thus be of different sizes. All of these issues will need to be considered to extend this program to a commercial grade OCR application.

Another limitation of this sample program is that only one drawing can be defined per character. You might want to use three different handwriting samples for a letter, rather than just one. The underlying neural network classes will easily support this feature. This change can be implemented by adding a few more classes to the user interface. To do so, you will have to modify the program to accept more training data than the number of output neurons.

6. EXPERIMENTAL RESULTS

The **Table 1** shows the experimental results on the NeurOCR based on the number of characters provided for training and the percent error (%).

The graph Fig. 6 shows the relative decrease in error rate of the character recognition.

Table 1

Sr. No.	Number of Alphabets fed to the Neural Network	Percentage (%) Error in Recognition
1	4	0.0093773
2	10	0.0061459
3	26	0.00327464

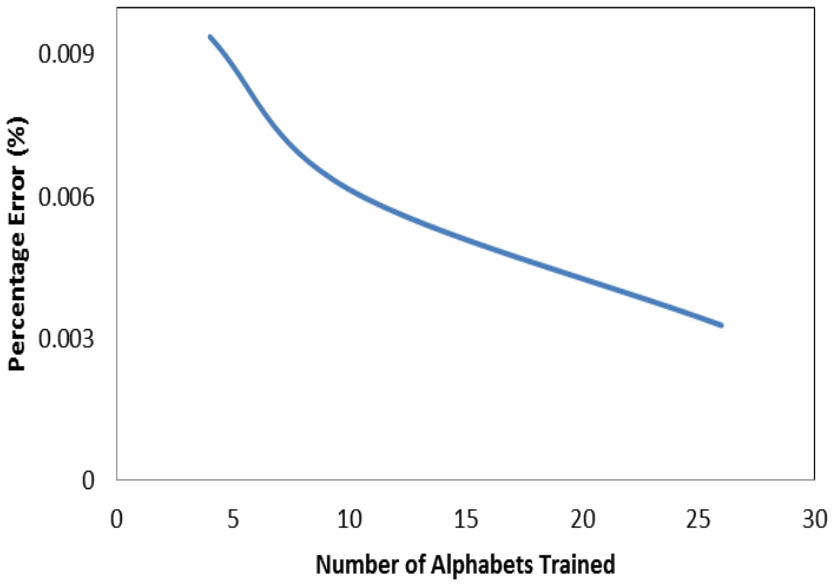

Fig. 6 Relative decrease in error rate in %

Hence, a substantial drop in the error rate can be noticed by increasing the number of character patterns fed to the SOM network.

7. CONCLUSION

A simplistic Soft Computing approach for recognition of visual characters using artificial neural networks has been described in this paper. The advantages of neural computing over classical methods have been outlined. Despite the computational complexity involved, artificial neural networks offer several advantages in pattern recognition and classification in the sense of emulating adaptive human intelligence to a small extent. Thus, implementation of Neural Networks has proven to save a lot of time[1] with respect to other Hard Computing techniques and methodologies. Neural Networks is a vast and rapidly growing domain, that is gaining its momentum. Its applications are not just limited to OCR systems some of the applications of Artificial Intelligence and Neural Networks have been deployed in military systems, stock market analysis (SmartMoney©), Robotics, Thermal Temperature controlling devices (Nuclear Reactor Heat Control System[8]), and etc.

REFERENCES

1. J.Pradeep, E.Srinivasan and S.Himavathi; “Neural Network based Handwritten Character Recognition system without feature extraction” on International Conference on Computer, Communication and Electrical Technology – ICCCET 2011.

2. N. Mani and P. Voumard; “An Optical Character Recognition using Arificial Neural Network”; IEEE International Conference on Systems, Man and Cybernetics, 1996

3. J. Summers and F. Catarro; *“*Assessment of handwriting speed and factors influencing written output of university students in examinations*”*; 2003.

4. Heaton Research® Neural Network API; Homepage: www.heatonresearch.com

5. J. Heaton 2005; “Introduction to Neural Networks using C#, 2nd Edition”

6. L. Fedorovici and F. Dragan, "A comparison between a neural network and a SVM and Zernike moments based blob recognition modules", Applied Computational Intelligence and Informatics (SACI), 2011 6th IEEE International Symposium, May 2011.

7. P. Shah, S. Karamchandani Nadkar, T. Gulechha, and K. Koli; "OCR-based chassis-number recognition using artificial neural networks" Vehicular Electronics and Safety (ICVES), 2009 IEEE International Conference on Nov. 2009.

8. C. Ku, K. Lee, and R. Edwards; “Improved Nuclear Reactor Temperature Control Using Diagonal Recurrent Neural Networks” on IEEE Transaction on Nuclear Science, December 1992.