Handwritten Script Recognition

Sangita Panmand[1] Dhanashree Shinde[2] Kirti Khair[3] Yugandhara Thumbre[4]

[1, 2, 3, 4]Computer Engineering Department

[1, 2, 3, 4] Padmabhushan Vasantdada Patil Pratishthan's College of Engineering (Mumbai University)

Abstract—OCR is abbreviated as Optical Character Recognition. Optical Character recognition is a process of recognition of different characters (printed or handwritten) from a digital image of documents. In OCR technique, characters can be recognized through optical mechanism. Various combinations of lines & curves make the characters. Characters recognition ability of human beings is very high. They can recognize all characters accurately. But same task is very difficult by OCR system. The wide usage of touch-screen based mobile devices has led to a large volume of the users preferring touch-based interaction with the machine, as opposed to traditional input via keyboards/mice. To exploit this, we focus on the Android platform to design a personalized handwriting recognition system that is acceptably fast, light-weight, possessing a user-friendly interface with minimally-intrusive correction and auto-personalization mechanisms.

Key words: OCR, Handwriting Recognition, Kohonen Neural Network, artificial neural network

I. INTRODUCTION

Handwritten character recognition, irrespective of the script, finds potential application areas for automation in various fields like postal automation, bank automation, form filling etc. Handwritten character recognition for Indian scripts is quite a challenging task for the researchers. This is due to the various characteristics of these scripts like their large character set, complex shape, presence of modifiers, presence of compound characters and similarity between characters. Various languages use specific script to write. One of them is Devnagari which is most widely used for many major languages such as Marathi, Hindi, etc. Hindi & Marathi are most commonly used languages by several thousand people. Devnagari Marathi Character contains complicated curves & various shapes. So recognition of Marathi characters is difficult & complicated task. All these considerations make Optical Character recognition (OCR) with Devnagari script very challenging. The ultimate goal of designing a character recognition system with an accuracy rate of 100 % is quite difficult because handwritten characters are non-uniform; they can be written in many different styles. Also they can be written in various sizes by different writers. Even there is variation in characters written by the same writer at different time.

II. HISTORICAL EVOLUTION

This paper focuses only on the offline handwritten recognition systems developed in the past few years. [4], [5] generally there are six major steps in the character recognition system which has been shown in fig.1.

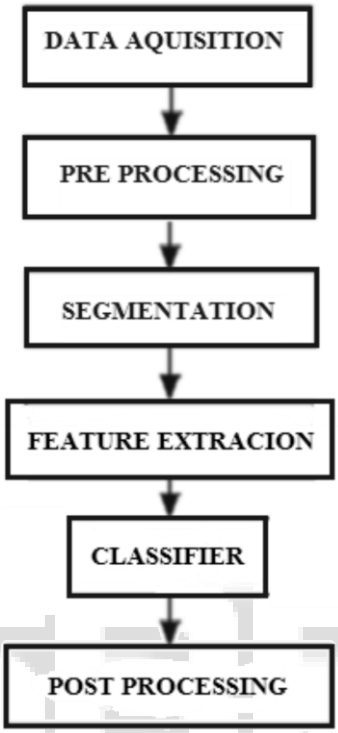

Fig. 1: Typical flowchart for CR methodology

A. *Data acquisition:*

Data acquisition, as the name implies is the process used to collect information to document or analyze some phenomenon. In the simplest form, a technician logging the temperature of an oven on a piece of paper is performing data acquisition.

B. *Pre-Processing*

Pre-processing is the major step in handwriting recognition System. In this, the algorithms described have been tested on binary word images. The black pixels in the binary images represent the handwriting whilst white pixels are used to denote the background. Fig 2 shows the block diagram of the pre-processing steps.

C. *Segmentation*

Character segmentation has long been one of the most critical areas of optical character recognition process. [7].Through this operation, a sequence of characters, which may be connected in some cases, is decomposed into individual alphabetic symbols. [6]

D. *Feature Extraction*

It incorporates many characteristics of handwritten characters based on structural, directional and zoning information and combines them to create a single global

feature vector. It is independent to character size and it can extract features from the raw data without resizing. [9]

E. Classifier

This step identifies the character that it represents and to assign them the corresponding ASCII code, after extracting character features.

F. Post Processing

Post-processing step is the final stage of the character recognition system. It prints the corresponding recognized characters in the structured text form.

III. PROPOSED SYSTEM DESIGN

The aim of this paper is to recognize handwritten character using artificial neural network. An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information.

One of the primary means by which computers are endowed with humanlike abilities, is through the use of a neural network. The human brain is the ultimate example of a neural network. The human brain consists of a network of over a hundred billion interconnected neurons. Neurons are individual cells that can process small amounts of information and then activate other neurons to continue the process. Neuron cell is the basic building block of human brain. It accepts signals from dendrites. When a neuron accepts a signal, that neuron may fire. When a neuron accepts a signal, that neuron may fire. When a neuron fires, a signal is transmitted over the neurons axon. Ultimately the signal will leave the neuron as it travels to axon terminals. The signal is then transmitted to other neurons or nerves. This signal, transmitted by the neuron is an analog signal. Most modern computers are digital machines, and thus require a digital signal. A digital computer processes information as either on or off. This is the basis of the binary digits zero and one. The presence of an electric signal represents a value of one, whereas the absence of an electrical signal represents a value of zero. Neuron accepts input from other neurons or user program. Activation function is used to activate particular neuron. If a particular neuron satisfies activation function then that neuron is said to be activated. Connections between neurons are assigned weights. These weights are adjusted to make neural network recognize different pattern.

A. Kohonen Neural Network

The Kohonen neural network differs considerably from the feed forward 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, we will examine an OCR program that uses 26 output neurons. These 26 output neurons map the input patterns into the 26 letters of the Latin alphabet. 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. [10] It is also important to understand the limitations of the Kohonen neural network. You will recall from the previous chapter that neural networks with only two layers can only be applied to linearly separable problems. This is the case with the Kohonen neural network. Kohonen neural networks are used because they are a relatively simple network to construct that can be trained very rapidly.

B. Principal of working

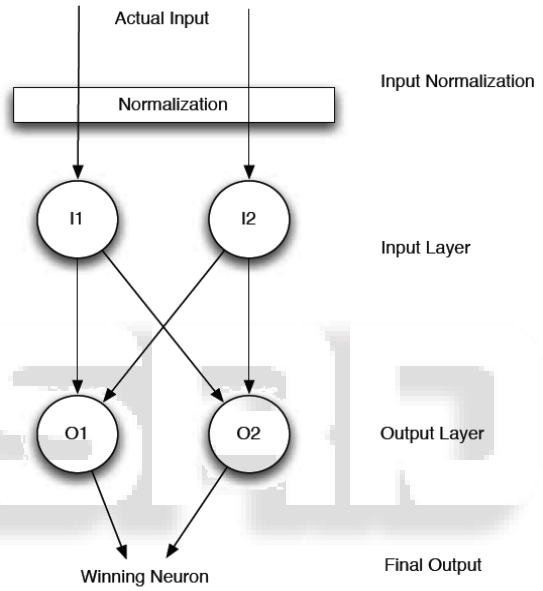

Fig. 2: Self-Organizing Map / Kohonen Neural Network

1) Sample Inputs to a Self-Organizing Map

This network will have only two input neurons and two output neurons. The input to be given to the two input neurons is shown in Table 1.

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

Table. 1: Inputs to a Self-Organizing Map

2) Connection Weights in the Sample Self-Organizing Map

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

Table. 2: Connection Weights

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

3) Normalizing the Input

The self-organizing map requires that its input be normalized. Thus, some texts refer to the normalization as a third layer. However, in this book, the self-organizing map is considered to be a two-layer network, because there are

only two actual neuron layers at work. 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. Input for a self-organizing map is generally normalized using one of two common methods, multiplicative normalization and z-axis normalization. Multiplicative normalization is the simpler of the two methods; however z-axis normalization can sometimes provide a better scaling factor. The algorithms for these two methods will be discussed in the next two sections. We will begin with multiplicative normalization.

4) 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 Equation (3.1) of Multiplicative Normalization

$$f = \frac{1}{\sqrt{\sum_{i=0}^{n-1} x_i{}^2}} \quad (3.1)$$

The above equation produces the normalization factor ha each input is multiplied by to properly scale them. Using the sample data provided in Tables 1 and 2, the normalization factor is calculated as follows*:*

$$1.0/\,Math.sqrt\,((0.5 * 0.5) + (0.75 * 0.75))$$

This produces a normalization factor of 1.1094.

5) 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, we use Equation (3. 2).

$$f = \frac{1}{\sqrt{n}} \quad (3.2)$$

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 Equation (3.3).

$$s = f \times \sqrt{n - l^2} \quad (3.3)$$

The 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. You might be wondering when you should use the multiplicative algorithm and when you should use the z-axis algorithm. In general, you will want to use the z-axis algorithm, since the z-axis algorithm preserves absolute magnitude. However, if most of the training values are near zero, the z-axis algorithm may not be the best choice. This is because the synthetic component of the input will dominate the other near-zero values.

6) 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 as shown in Equation (3.4).

$$[0.5 \quad 0.75] \times [0.1 \quad 0.2]$$
$$= (0.5 \times 0.75) + (0.1 \times 0.2)$$
$$= 0.395$$

As you can see from the above calculation, the dot product is 0.395. This calculation will have to be performed for each of the output neurons. In this example, we will only examine the calculations for the first output neuron. The calculations necessary for the second output neuron are carried out in the same way. The output must now be normalized by multiplying it by the normalization factor that was determined in the previous step. You must multiply the dot product of 0.395 by the normalization factor of 1.1094. The result is an output of 0.438213. Now that the output has been calculated and normalized, it must be mapped to a bipolar number.

7) Mapping to a Bipolar Ranged Number

As you may recall from chapter 2, a bipolar number is an alternate way of representing binary numbers. In the bipolar system, the binary zero maps to –1 and the binary one remains a 1. Because the input to the neural network has been normalized to this range, we must perform a similar normalization on the output of the neurons. To make this mapping, we multiply by two and subtract one. For the output of 0.438213, the result is a final output of –0.123574. The value –0.123574 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.

8) Choosing the Winner

We have seen how to calculate the value for the first output neuron. If we are to determine a winning neuron, we must also calculate the value for the second output neuron. We will now quickly review the process to calculate the second neuron. The same normalization factor is used to calculate the second output neuron 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, resulting in a value of 0.0465948. We can now calculate the final output for neuron 2 by converting the output of 0.0465948 to bipolar, which yields –0.9068104. As you can see, we now have an output value for each of the neurons. The first neuron has an output value of –0.123574 and the second neuron has an output value of –0.9068104. To choose the winning neuron, we select the neuron that produces the largest output value. In this case, the winning neuron is the second output neuron with an output of –0.9068104, which beats the first neuron's output of –0.123574. You have now seen how the output of the self-organizing map was derived. As you can see, the weights between the input and output neurons determine this output.

IV. EXECUTION OF SYSTEM

Following flowchart shows the stepwise execution of above handwritten character recognition system.

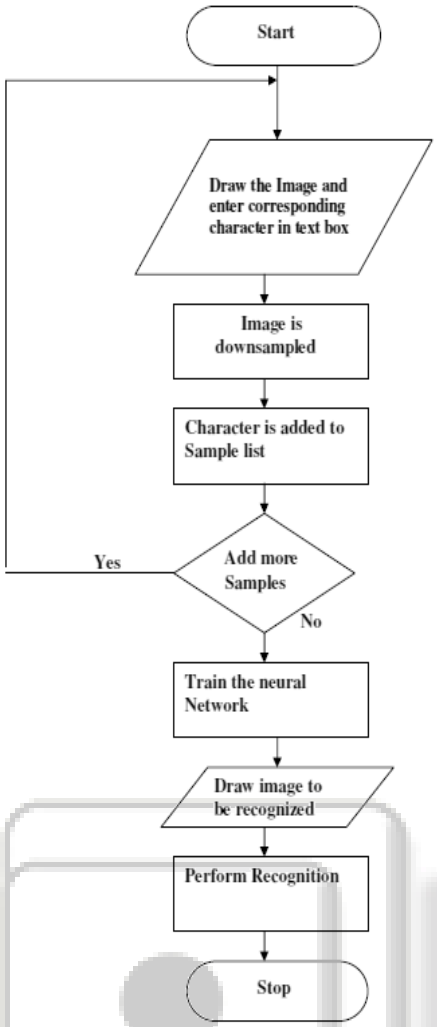

Fig. 3: Flowchart of Execution

1) Draw the Image

Use the mouse as an input to draw the image of characters. Enter the corresponding character with which you want the drawn character to be recognized in the text box.

2) Down Sampling

The images have to be cropped sharp to the border of the character in order to standardize the images. The image standardization is done by finding the maximum row and column with 1s and with the peak point, increase and decrease the counter until meeting the white space, or the line with all 0s. This technique is shown in figure below where a character "S" is being cropped and resize.

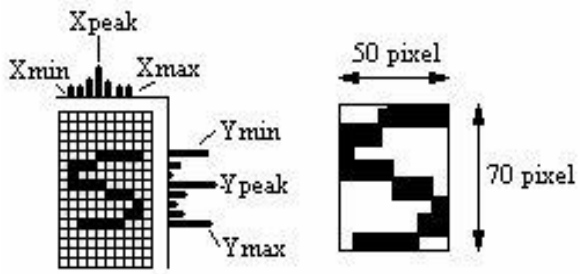

Fig. 4: Down sampling

The image pre-processing is then followed by the image resize again to meet the network input requirement, 5 by 7 matrices, where the value of 1 will be assign to all pixel where all 10 by 10 box are filled with 1s, as shown below:

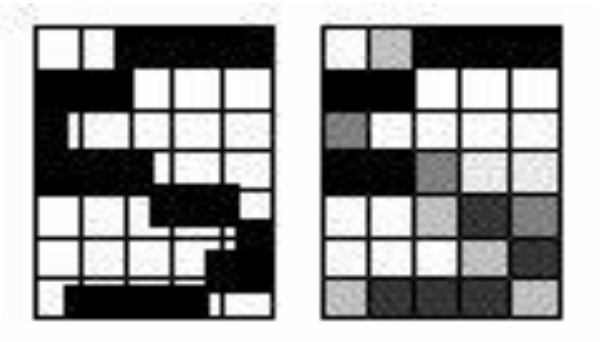

Fig. 5: Pre-processing of image

Finally, the 5 by 7 matrices are concatenated into a stream so that it can be feed into network 35 input neurons.

3) Train the Neural Network

In training the neural network we actually make the neural network learn how to recognize image. The process of training is adjusting the individual weights between each of the individual neurons until we achieve close to the desired output.

4) Recognition

Draw the image to be recognized in drawing area and click recognize button. The best neuron will be fired as output and recognition will be performed.

V. CONCLUSION

This paper has presented a review about Handwritten Character Recognition is one of the main branches of character recognition. Three stages of CR are presented namely, pre-processing, feature extraction and classification. In the future, we will use this knowledge to develop the characters recognition system. The task of human expert is to select or invent features that allow effective and efficient recognition of character. It is important to get the better result in recognizing a character. Many types of classifier are applicable to the OCR system. Recognition of a character can be done using a template matching, statistical, syntactic (structural) and neural network approach. Neural network approach is selected in this paper because it gives a better recognition result than compared to other.

REFERENCES

[1] H. Yamada, K. Yamamoto, T. Saito: "A Non-linear Normalization Method for Hand printed Kanji Character Recognition – Line Density Equalization," Pattern Recognition, vol.23, no.9, pp.1023-1029, (1990).

[2] N. R. Pal and S. A Pal, "Review on image segmentation techniques," Pattern Recognition, Vol. 26, pp. 1277–1294, 1993.

[3] Singh, S. and Amin, A. (1999). Neural network Recognition of Hand-printed Characters. Neural Compute Applications. Springer-Verlag London. 8: 67-76.

[4] Avi-Itzhak H.I., Diep T.A. and Garland H., ― High Accuracy Optical Character Recognition Using Neural Network with Centroid Dithering‖, IEEE Transaction on

Pattern Analysis and Machine Intelligence, vol.17, no.2,pp.218-223, 1995.

[5] M. Y. Chen, A. Kundu, and J. Zhou, ―Off-line handwritten word recognition using a hidden Markov model type stochastic network, ‖ IEEE Transaction on. Pattern Analysis and Machine Intelligence. vol. 16, pp. 481–496, May 1994.

[6] Simone Marinai, Marco Gori and Giovanni Soda, ― Artificial Neural Network for Document Analysis and Recognition‖, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.27, no.27, pp. 23-35,January 2005.

[7] Vassilis Papavassiliou, Themos Stafylakis, Vassilis Katsouros and words, ―Handwritten document image segmentation into text lines and words‖, Pattern recognition, vol.43, pp.369-377, 2010.

[8] Ø. D. Trier, A. K. Jain, and T. Taxt, ―Feature extraction method for character recognition—A survey,‖ Pattern Recognition., vol. 29, no. 4, pp. 641–662, 1996.

[9] I.S. Oh, J. S. Lee, and C. Y. Suen, ―Analysis of class separation and combination of class-dependent features for handwriting recognition,‖ IEEE Transaction on Pattern Analysis and Machine Intelligence., vol. 21, pp. 1089–1094, Oct.1999.

[10] Kohonen, Teuvo (1982)."Self-Organized Formation of Topologically Correct Feature Maps". Biological Cybernetics 43 (1): 59–69.

[11] Ultsch, Alfred; Siemon, H. Peter (1990). "Kohonen's Self-Organizing Feature Maps for Exploratory Data Analysis". In Widrow, Bernard; Angeniol, Bernard. Proceedings of the International Neural Network Conference (INNC-90), Paris, France, July 9–13, 19901. Dordrecht, Netherlands: Kluwer. pp. 305–308. ISBN 978-0-7923-0831-7.

