Appendix D:

Copies of two of the Papers Published in International Journals

Hand Written English Character Recognition using Column-wise Segmentation of Image Matrix (CSIM)

RAKESH KUMAR MANDAL
N R MANNA
Department of Computer Science & Application
University of North Bengal
PO: NBU, Distt: Darjeeling
West Bengal-734013
INDIA
rakesh_it2002@yahoo.com, nrmanna@sify.com http://www.nbu.ac.in

Abstract: - Research is going on to develop both hardware and software to recognize handwritten characters easily and accurately. Artificial Neural Network (ANN) is a very efficient method for recognizing handwritten characters. Attempts have already been made to recognize English alphabets using similar type of methods. A new method has been tried, in this paper, to improve the performance of the previously applied methods. The input image matrix is compressed into a lower dimension matrix in order to reduce non significant elements of the image matrix. The compressed matrix is segmented column-wise. Each column of a particular image matrix is mapped to identical patterns for recognizing a particular character. Majority of a known pattern decides the existence of a particular character.

Key-Words: - ANN, CSIM, Compression, Perceptron, Segmentation, Learning Rule

1 Introduction

Expert systems can be developed using Artificial Neural Network (ANN) which can recognize hand written English alphabets easily and accurately. The handwriting styles of different individuals vary infinitely which makes the development of expert systems to recognize handwritten characters very difficult. Common features extracted out from different handwriting styles have proved to be a very successful method. Generalized problems can be solved using Multiscale Training Technique (MST), [1, 2, 3].

One such method has been applied in Devnagri Script where some feature extraction techniques like intersection, shadow features, chain code histogram and straight line fitting were used, [4].

One feature extraction method is finding out twelve directional feature inputs which depend upon the gradients. The features of the characters are directions of the pixels w.r.t. neighbouring pixels, [5].

Hand written English character recognition using Row-wise Segmentation Technique (RST) is an approach to find out common features of same characters written in different hand writing styles by segmenting the input pattern matrix into separate rows and trying to find out common rows among different hand writing styles, [6].

In order to find out common features among the characters written by different individuals some methods are added to the already developed ANNs like perceptron learning method, [7, 8].

In this paper an attempt has been made through the compressed Column-wise Segmentation of Image Matrix (CSIM) to recognize hand written characters using Neural Network.

The overall program is divided into three parts, compression of the image matrix, segmenting the compressed matrix column-wise and training the net and finally testing the net by providing characters taken from different individuals.

2 Methodology

A scanner is used to scan the handwritten English alphabets written on a piece of A4 size paper. Each character is enclosed in a box of size 80 x 80 in jpeg

format. The image is converted into a binary matrix of size 80 x 80. Each matrix contains a large number of elements which conveys no information about the pattern. In order to eliminate such elements 80 x 80 matrix is compressed into a matrix of size 10 x 10. The 10 x 10 matrix is segmented column-wise into 10 segments of size 10 each. All the columns of a particular character are mapped into identical patterns used to recognize that particular character. In this way standard weight matrix is obtained for each character. Test characters are presented to the trained net. A counter is used to find out the number of identical patterns produced by the test character. Majority of the identical pattern present in a test character decides the existence of a particular character.

2.1 Compression of 80 x 80 matrix into 10 x 10 matrix

A matrix can be compressed into a matrix of lower dimension in order to reduce the non significant elements of the matrix. Matrix A can be compressed into matrix A_COM by using the function Φ.

Matrix A can be split into n uniform blocks. The dimension of each block A_BLOCK_i is m x m.

2.1.1 Algorithm Compression

Step1. Input Matrix A.

Step2. Split A into n uniform blocks, where A_BLOCK_i is the i^{th} block of A.

Step3. for i=1 to n,

$A\_COM_i = \Phi(A\_BLOCK_i)$

End,

where A_COM_i is the i^{th} element of the compressed matrix A_COM and

$\Phi(A\_BLOCK_i) = 1$, if there exists at least one element in A_BLOCK_i, 1

$\Phi(A\_BLOCK_i) = 0$, if all the elements of A_BLOCK_i are 0

Step 4. Stop

Example 1. Given matrix A of size 4 x 4

$$A = \begin{matrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

A can be split into 4 uniform blocks of dimension 2 x 2 each.

$$A\_BLOCK_1 = \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$$

$$A\_BLOCK_2 = \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix}$$

$$A\_BLOCK_3 = \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}$$

$$A\_BLOCK_4 = \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}$$

Using algorithm Compression

$$A\_COM = \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}$$

In the same way 80 x 80 binary matrix is compressed into a 10 x 10 binary matrix considering 100 uniform blocks of dimension 8 x 8 each.

2.2 Column-wise Segmentation of 10 x 10 matrix

The 10 x 10 matrix is segmented column-wise into 10 columns of size 10 each. Each column is mapped into 10 identical patterns used to recognize a character, (Figure 1). Weight matrix is initialized to zero. Perceptron learning rule is used to train the net.

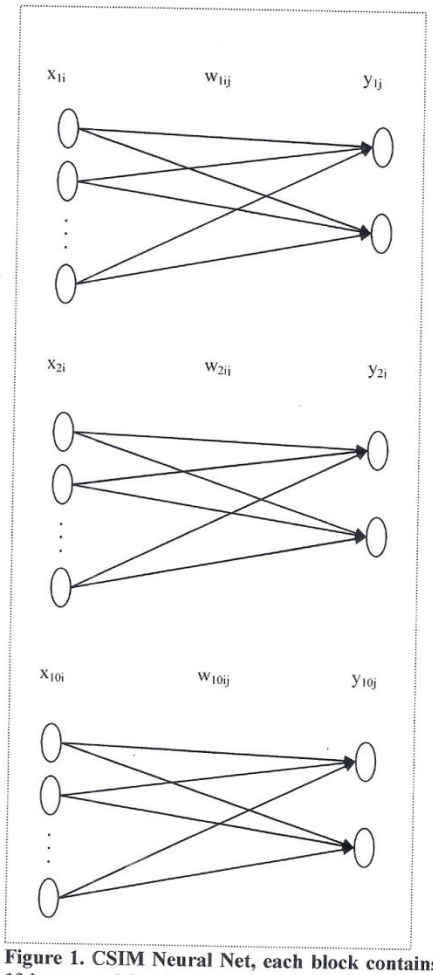


Figure 1. CSIM Neural Net, each block contains 10 inputs and 2 outputs

2.2.1 Perceptron Learning Rule

In a simple peceptron one or more than one neurons are connected to a single neuron with the help of weights, (Figure 2). Weights may be initialized to 0 or very small values. An iterative procedure is used to find out the standard weights [7, 8]. At standard weights the net generates correct outputs. The output of the j^{th} perceptron is y_j. Output is calculated by applying the activation function f, which is a function of the net output, y_out. Thus, $y_j=f(y\_out_j)$

and

$$y\_out_j = \sum_i x_i w_{ij} \qquad (1)$$

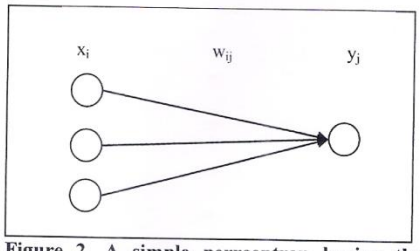


Figure 2. A simple perrceptron having three inputs and one output

Where, x_i is the i^{th} element of input vector and w_{ij} is the weight between the i^{th} element of the input vector and j^{th} element of the output vector. The function $f(y\_out_j)$ takes the following values depending upon the values of y_out_j.

$$f(y\_out_j) = \begin{cases} 1 \text{ if } y\_out_j > \theta \\ 0 \text{ if } -\theta <= y\_out_j <= \theta \\ -1 \text{ if } y\_out_j < -\theta \end{cases}$$

Here, θ is the threshold value, taken at random. For each training input, the net would calculate the response of the output unit.

The net would determine whether an error occurred for this pattern by comparing the calculated output with the target value. If an error occurs for a particular training input pattern, the weights would be changed according to the formula:

$$w_{ij}(new) = w_{ij}(old) + \alpha t_j x_i \qquad (2)$$

α is the learning rate, the value of which is taken at random, t_j is the target value, i.e. the output expected from the net. The output y_j produced by the net is compared with t_j and the difference leads to the modification in weight given by equation (2). The process continues until y_j becomes equal to t_j. Weights obtained at that point are the final and standard weight.

2.3 Training of the net to obtain the weight matrix

The training started with an input vector of size [80 x 80 = 6400], compressed into a vector of size [10 x 10 = 100], (Figure 3).

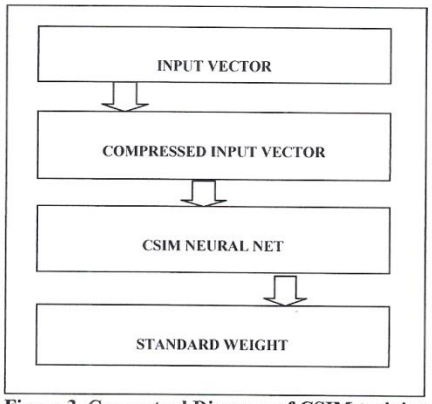


Figure 3. Conceptual Diagram of CSIM training

First four characters of the English alphabets are considered for training, (Figure 4).The weights are initially set to zero. The net, (Figure 1) is trained using Perceptron Learning Rule. After training the net for few epochs, the final weight that generates the correct output is obtained. The final weight obtained remains the same for the next few epochs and is considered as the standard weight for testing.

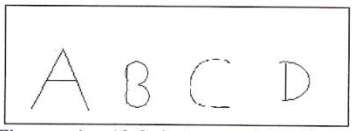


Figure 4. Alphabets considered for CSIM training

2.4 Testing of the net

The net is tested with first four characters of English alphabets, deviated to the extent, at which the deviated character can be identified, (Figure 5). The testing starts with a [80 x 80 = 6400] vector compressed into a [10 x 10 =100] vector on the net as shown in Figure 1.

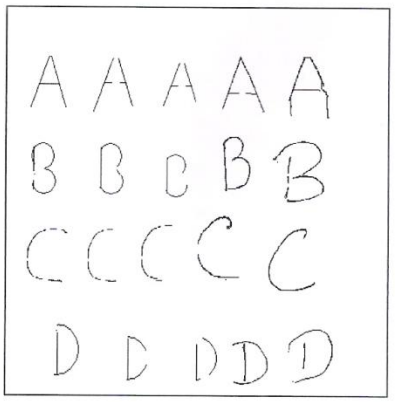


Figure 5. Alphabets considered for CSIM testing

The compressed vector is presented to the net having the standard weight matrix. Since, there are two outputs, input patterns are mapped as either [-1, -1], [-1, 1], [1, -1] or [1,1] to identify the characters A, B, C or D respectively. A counter is used to count the number of identical pattern outputs from the ten different segments for confirmation. A value greater than or equal to six is considered for the identification of a particular character, (Figure 6).

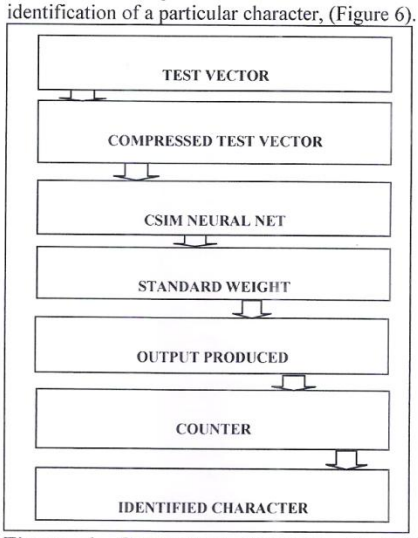


Figure 6. Conceptual Diagram of CSIM testing

3 Algorithm CSIM

Step 1. Scan, N number of characters to be trained and convert each into a binary matrix of dimension m x m.

Step 2. Apply algorithm compression on each m x m matrix and compress into n x n matrix, where m>n.

Step 3. Store each compressed matrix in column major form and assign one by one to the input vector x.

Step 4. Divide x into M number of segments, where each segment contains n number of elements.

Step 5. Consider M number of groups in the output layer, y. Each group contains p number of elements, where

p=2, if 1<N<=4

p=3, if 4<N<=8

p=4, if 8<N<=16 and so on.

Step 6. Completely interconnect all the neurons of each M segment in the input layer to all the neurons of each corresponding M group in the output layer.

Step 7. For 1 to N number of alphabets to be trained, consider the target vector in the following way:

For example,

A=00

B=01

C=10

D=11 and so on.

Step 8. Initialize weight matrix

For k=1 to M

For i=1 to n

For j=1 to p

$W_{kij}=0$

End p

End n

End M

Step 9. Apply Perceptron Learning Algorithm to each segment and find out the standard weights.

Step 10. Scan the sample character to be tested and convert into binary matrix of dimension m x m.

Step 11. Apply algorithm Compression on the sample m x m matrix and compress into n x n matrix, where m>n.

Step 12. Store the compressed matrix in column major form and assign one by one to the input vector x.

Step 13. Present the test vector x into the CSIM net.

Step 14. Apply counter to the output produced, to identify the character.

The following example demonstrates the applicability of the CSIM algorithm. It can be observed that the characters when compressed using the compression algorithm holds the pattern identified by the naked eye.

Example 2. Let us consider four characters of English alphabet A, B, C and D.

A can be represented by a 10 x 10 matrix shown below.

					*				
				*		*			
			*				*		
		*						*	
	*								*
*									*
*	*	*	*	*	*	*	*	*	*
*									*
*									*
*									*

In the above matrix the blank space can be considered as -1 and * can be considered as 1. So, the vector representation of the matrix A in column major form is represented as:

A={-1,-1,-1,-1,-1,1,1,1,1,1,-1,-1,-1,-1,1,-1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,1,1,1,1,-1,-1,1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,1,1,1,1,1}

A can be compressed to form a matrix of dimension 5 x 5 by applying compression algorithm.

		*	*	
	*		*	*
*	*	*	*	*
*				*
*				*

A_COM={-1,-1,1,1,1,-1,1,1,-1,-1,1,-1,1,-1,-1,1,1,1,-1,-1,-1,1,1,1,1}

Similarly, compression algorithm is applied on B, C and D to form B_COM, C_COM and D_COM.

Matrix representation of character B is given below.

*	*	*	*	*	*	*	*	*	
*									*
*									*
*									*
*									*
*	*	*	*	*	*	*	*	*	
*								*	
*									*
*									*
*	*	*	*	*	*	*	*	*	

B={1,1,1,1,1,1,1,1,1,1, 1,-1,-1,-1,-1,1,-1,-1,-1,1, 1,-1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,1, 1,-1,-1,-1,-1,1,1,-1,-1,1, -1,1,1,1,1,-1,-1,1,1,-1}

Matrix representation of character B_COM is given below.

*	*	*	*	*
*				*
*	*	*	*	*
*				*
*	*	*	*	*

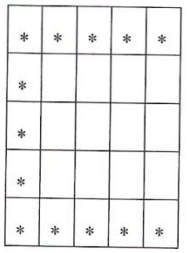
B_COM={1,1,1,1,1, 1,-1,1,-1,1, 1,-1,1,-1,1, 1,-1,1,-1,1, 1,1,1,1,1}

Matrix representation of character C is given below.

	*	*	*	*	*	*	*	*	*
*									
*									
*									
*									
*									
*									
*									
*									
	*	*	*	*	*	*	*	*	*

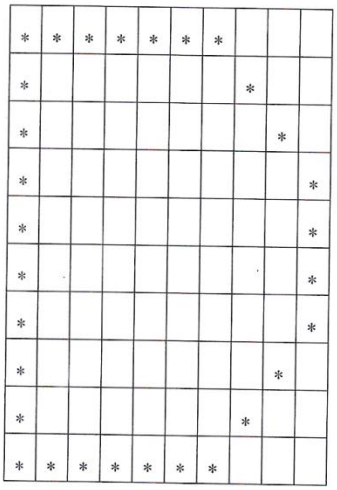
C={-1,1,1,1,1,1,1,1,1,-1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,1}

Matrix representation of character C_COM is given below.



C_COM={1,1,1,1,1,1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,1, 1,-1,-1,-1,1, 1,-1,-1,-1,1}

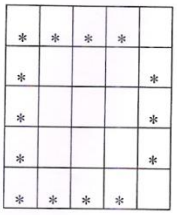
Matrix representation of character D is given below.



D={1,1,1,1,1,1,1,1,1,1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1, 1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1, -1,1,-1,-1,-1,-1,-1,-1,-1,1,-1, -1,-1,1,-1,-1,-1,-1,-1,-1,-1,1, -1,-1,-1,1,1,1,1,1,-1,-1,-1}

Matrix representation of character C_COM is given below.

D_COM={1,1,1,1,1,1, 1,-1,-1,-1,1, 1,-1,-1,-1,1, 1,-1,-1,-1,1, -1,1,1,1,-1}



D_COM={1,1,1,1,1, 1,-1,-1,-1,1, 1,-1,-1,-1,1, 1,-1,-1,-1,1, -1,1,1,1,-1}

The training of the net will be compleneted in the following epochs.

Epoch 1

Table 2. Segmented inputs of four characters

	x1	x2	x3	x4	x5
A_C	-1, -1, 1, 1, 1	-1, 1, 1, -1, -1	1, -1, 1, -1, 1	1, 1, 1, -1, -1	-1, 1, 1, 1, 1
B_C	1, 1, 1, 1, 1	1, -1, 1, -1, 1	1, -1, 1, -1, 1	1, -1, 1, -1, 1	1, 1, 1, 1, 1
C_C	1, 1, 1, 1, 1	1 ,-1, -1, -1, 1	1, -1, -1, -1, 1	1, -1, -1, -1, 1	1, -1 ,-1, -1, 1
D_C	1, 1, 1, 1, 1	1, -1 ,-1, -1, 1	1, -1, -1, -1, 1	1, -1, -1, -1, 1	-1, 1, 1, 1, -1

The segmented inputs of four characters are presented to the net one by one for training. The initial weight matrix is set to zero as the weights are not known in advance. The target matrix is known in advance. The learning used here is supervised in nature. The target matrix depicts the expected result. Taking w_{ij}=0, initially the net output as well as the activations are calculated. The net output and the activations calculated after first epoch are given in the following tables. It can be observed that the expected results are not obtained after the first epoch.

Table 3. Initial Weight Matrix

	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg1	0	0	0	0	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg2	0	0	0	0	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg3	0	0	0	0	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg4	0	0	0	0	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg5	0	0	0	0	0	0	0	0	0	0

Table 4. Net Output after epoch1

	Y_IN11	Y_IN 21	Y_IN 31	Y_IN 41	Y_IN 51
A_C	0	0	0	0	0
B_C	-3	1	-3	-1	-3
C_C	-3	2	-1	1	3
D_C	4	2	6	3	-1
	Y_IN12	Y_IN 22	Y_IN 32	Y_IN 42	Y_IN 52
A_C	0	0	0	0	0
B_C	-3	1	-3	-1	-3
C_C	0	3	0	2	0
D_C	-1	0	-1	1	5

The epochs are repeated until the expected results are obtained. In this example it is found that the correct results are obtained in three epochs. The following condition is satisfied at epoch 3.

$y_j = t_j$

Table 5. Activation after epoch1

	Y11	Y 21	Y 31	Y 41	Y 51
A_C	0	0	0	0	0
B_C	-1	1	-1	-1	-1
C_C	-1	1	-1	1	1
D_C	1	1	1	3	-1
	Y12	Y 22	Y 32	Y 42	Y 52
A_C	0	0	0	0	0
B_C	-1	1	-1	-1	-1
C_C	0	1	0	1	0
D_C	-1	0	-1	1	1

Epoch 2

The activation obtained after epoch1 are not equal to the target. So, the net is trained for the next epoch.

Table 6. Weight matrix obtained after epoch2

	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg1	0	0	2	2	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg2	0	1	0	-1	-2	-1	2	1	0	1
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg3	0	0	0	0	-2	0	0	0	2	2
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg4	-1	-1	-1	-1	-1	1	1	1	1	1
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg5	0	1	0	1	0	1	0	1	-2	-1

Table 7. Net Output after epoch2

	Y_IN11	Y_IN 21	Y_IN 31	Y_IN 41	Y_IN 51
A_C	-4	-6	-6	-7	-6
B_C	0	-6	-2	-1	-2
C_C	1	2	6	1	-2
D_C	-1	2	6	3	1
	Y_IN12	Y_IN 22	Y_IN 32	Y_IN 42	Y_IN 52
A_C	-6	-9	-6	-5	-1
B_C	6	3	6	5	7
C_C	-2	1	-2	-5	-7
D_C	6	2	6	1	5

Table 8. Activation after epoch2

	Y11	Y 21	Y 31	Y 41	Y 51
A_C	-1	-1	-1	-1	-1
B_C	0	-1	-1	-1	-1
C_C	1	1	1	1	-1
D_C	-1	1	1	1	1
	Y12	Y 22	Y 32	Y 42	Y 52
A_C	-1	-1	-1	-1	-1
B_C	1	1	1	1	1
C_C	-1	1	-1	-1	-1
D_C	1	1	1	1	1

Epoch3 produces the final standard weight matrix for training.

Epoch3

Table 9. Weight matrix obtained after epoch3

	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg1	0	0	2	2	0	0	0	0	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg2	0	0	0	0	-2	0	2	2	0	0
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg3	0	0	0	0	-2	0	0	0	2	2
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg4	-1	-1	-1	-1	-1	1	1	1	1	1
	w11	w12	w21	w22	w31	w32	w41	w42	w51	w52
Seg5	1	1	-1	1	-1	1	-1	1	-1	-1

Table 10. Net Output after epoch3

	Y_IN11	Y_IN 21	Y_IN 31	Y_IN 41	Y_IN 51
A_C	-6	-6	-6	-7	-9
B_C	-2	-6	-2	-1	-5
C_C	6	2	6	1	5
D_C	6	2	6	3	1
	Y_IN12	Y_IN 22	Y_IN 32	Y_IN 42	Y_IN 52
A_C	-6	-6	-6	-5	-1
B_C	6	2	6	5	7
C_C	-2	-6	-2	-5	-7
D_C	6	2	6	1	5

Table 11. Activation after epoch3

	Y11	Y 21	Y 31	Y 41	Y 51
A_C	-1	-1	-1	-1	-1
B_C	-1	-1	-1	-1	-1
C_C	1	1	1	1	1
D_C	1	1	1	1	1
	Y12	Y 22	Y 32	Y 42	Y 52
A_C	-1	-1	-1	-1	-1
B_C	1	1	1	1	1
C_C	-1	-1	-1	-1	-1
D_C	1	1	1	1	1

4 Result Analysis

Scilab was used to test the accuracy of the net. Four characters A, B, C and D are taken initially for testing. The response of the experiment shows very good results for the first four alphabets. It was found that the neural network identifies each sample up to 40% distortion of the test sample from the standard character.

Neural Network parameters used in the experiment

Number of input neurons =6400

Number compressed input neurons=100

Number of neurons in each input pattern segment=10

Number of input pattern segments=10

Number of Neurons in each output group = 2

Dimension of weight matrix for each segment = 10 x 2

Training Algorithm used = Perceptron

Number of Epoch = 3, $\theta = 0.2$, $\alpha = 1$

$t_j = \{A(-1,-1), B(-1,1), C(1,-1), D(1,1)\}$

Table 1, shows the results produced by the neural net when characters were presented to the net with different levels of distortions from the standard ones. NI stands for not identified.

Table 12. Identification of the deviated characters

	Characters presented with different levels of distortion				
	10%	**20%**	**30%**	**40%**	**50%**
A	Identified	Identified	Identified	Identified	Identified
B	Identified	Identified	Identified	Identified	Identified
C	Identified	Identified	Identified	Identified	Identified
D	Identified	Identified	Identified	Identified	NI

5 Discussion

The results show that CSIM training, allows very fast convergence. The number of epochs required for the training is very less. Accuracy of identifying the characters is also very good. Number of samples identified with maximum distortion from the standard ones is satisfactory. The compression of the input matrix helps to improve the performance of CSIM net to a great extent. It was also found that the method of compression applied is able to hold the information in the compressed matrix with great accuracy. However, the method is tested for a few sample characters.

6 Conclusion

CSIM is a better approach as compared to the previously tried methods of segmentation. Compression of the input matrix helped a lot to get rid of the unused elements of the input matrix. Removal of unused elements makes the net small, simple and fast. Column-wise segmentation is a better approach than the row-wise segmentation as

segmentation of the matrix column-wise produces more variation in the patterns and helps to obtain a sharp eye in the identification of the pattern. Experiments will be carried out to refine the already developed methods in order to enhance the performance and accuracy of the net.

7 Acknowledgements

Thanks to all my colleagues who contributed their moral support in the development of this paper.

References:

[1] Robinson, G. (1995), The Multiscale Technique, Available: http://www.netlib.org/utk/Isi/pcwLSI/text/node123.html.

[2] Handwritten Character Recognition, Available: http://tcts.fpms.ac.be/rdf/hcrinuk.htm.

[3] Velappa Ganapathy, and kok Leong Liew, Handwritten Character Recognition Using Multiscale Neural Network Training Technique, World Academy of Science, Engineering and Technology 39 2008.

[4] Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, Dipak kumar Basu and Mahantapas Kundu, Combining Multiple Feature Extraction Techniques for Handwritten Devnagri Character Recognition, Available: http://arxiv.org/ftp/arxiv/papers/1005/1005.4032.pdf, (Accessed: 26, July, 2010).

[5] Dayashankar Singh, Sanjay Kr. Singh and Dr. (Mrs.) Maitreyee Dutta, Hand Written Character Recognition Using Twelve Directional Feature Input and Neural Network, Available: http://www.ijcaonline.org/journal/number3/pxc387173.pdf, (Accessed: 26, July, 2010).

[6] Rakesh Kumar Mandal and N R Manna, Hand Written English Character Recognition using Row-wise Segmentation Technique (RST), International Symposium on Devices MEMS, Intelligent Systems & Communication (ISDMISC) 2011, Proceedings published by International Journal of Computer Applications (IJCA).

[7] Neural Networks, G.N. Swamy, G. Vijay Kumar, SCITECH.

[8] Fundamentals of Neural Networks, Architectures, Algorithms and Applications, Laurene Fausett, Pearson Education.

IJCSE International Journal of Computer Science and Engineering Open Access
Research Paper Volume-2, Issue-3 E-ISSN: 2347-2693

Handwritten English Character Recognition using Pixel Density Gradient Method

Rakesh Kumar Mandal[1*] and N R Manna[2]

[1*] *Department of Computer Science & Application, University of North Bengal, India, rakesh_it2002@yahoo.com*
[2] *Department of Computer Science & Application, University of North Bengal, India, nrmanna12@gmail.com*

www.ijcaonline.org

Received: 28 Feb 2014 Revised: 10 March 2014 Accepted: 22 March 2014 Published: 30 March 2014

***Abstract*—** Handwritten character recognition is a subject of importance in these days. Artificial Neural Networks (ANNs) are very much in demand in order to accomplish the task and that is why mass research is also going on in this field. This paper is an approach to identify handwritten characters by observing the gradient of the pixel densities at different segments of the handwritten characters. Different segments of the characters are observed carefully with the help of generated computer programs and rigorous experiments. It is found that the pixel densities at various segments of the character image matrix of different alphabets vary. The gradient of the pixel densities in these segments are used to form unique codes for different alphabets, which are found standard for different variations of same alphabet. Generation of unique codes actually extracts out common features of a particular alphabet written by one or more individuals at different instants of time. The unique codes formed for different alphabets are used to recognize different test alphabets. The method developed in this paper is a feature extraction technique which uses self organizing neural network, where supervised learning is not required.

***Keywords*—**Artificial Neural Networks; Pixel Density Gradient; Segments; Handwritten Character

I. INTRODUCTION

Handwritten character recognition using neural networks is very interesting and challenging task for many researchers. Handwriting character identification is not an easy task as different individuals have different handwriting styles and a single individual never repeats identical patterns. So, it is not worth to go for the pattern matching approach to identify characters. Many techniques have already been developed in this field [1], [4], [5], [6], [7]. Extracting out common features from varying patterns is a good approach. Feature extraction techniques are used to recognize handwritten characters, where generalization is done by using Multi scale Training Technique (MST) [1], [2], [3]. Feature extraction technique is also found good for identifying characters written in different non English scripts like Devnagri script, where common features like intersection, shadow, chain code histogram and straight line fitting are analyzed [4]. 'Twelve Directional' Method is a feature extraction method, which depends upon the gradients of the pixels, representing the character in the image matrix, where the features of the characters are directions of the pixels w.r.t. neighbouring pixels [5]. Row-wise Segmentation Technique (RST) [6] is a feature extraction method where the two dimensional image matrix of the character is segmented into rows to analyze the common features of same characters written in different handwriting styles among the corresponding rows. Column-wise Segmentation of Image Matrix (CSIM) [7] is a feature extraction method where the two dimensional image matrix segments to measure pixel density, formation of 5-bit binary codes for each character and testing the net are discussed in Section-3. Section 4 describes the result analysis. Section 5 of the character is segmented into columns to analyze the common features of same characters written in different handwriting styles among the corresponding columns. Methods used to extract the features can be appended to the already developed ANNs like 'Perceptron Learning Method' in order to achieve better convergence while training [8], [9], [10], [11].

In this paper, an attempt has been made to map infinite pattern variations of a particular character into an unique pattern and hence identification of the character through a net called PDG net, capable of finding variation(gradient) of the pixel densities, present in the character, in its different segments. This Pixel Density Gradient (PDG) net is nothing but a self organized neural net which extracts out common features of a particular character written at different instants of time by generating unique codes for the alphabets. A combination of pixel densities of different segments of the characters is used to find out unique binary codes of 5 bits, which are ultimately decoded into 26 lines representing the 26 uppercase English alphabets. The test alphabets will specify which output line of 26 lines, representing which alphabet.

An overview of the paper is given as follows: Section 2 describes the architecture of the PDG-net. Section 3, describes the overall methodology used in the PDG-net. Preparation of input pattern, identifying row and column

Corresponding Author: Rakesh Kumar Mandal

describes the discussion and Section 6 describes the conclusion.

II. ARCHITECTURE OF THE PDG-NET

A self organized neural net is an unsupervised type of neural net where neural net tends to converge towards the desired results using fixed weights, [12].

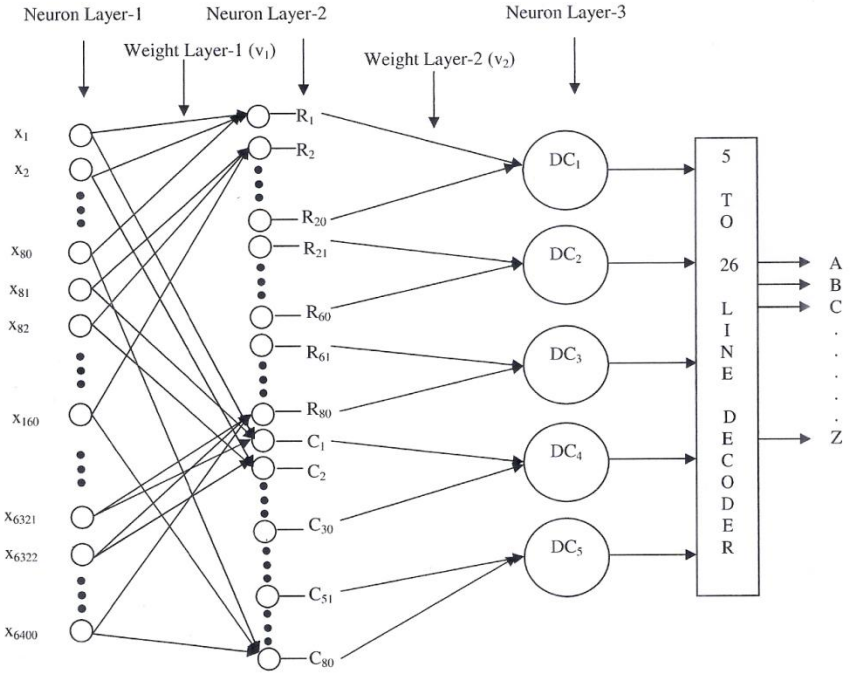


Fig. 1. Neural Network representing pixel density gradient method

Fig. 2. Alphabets used to form clusters

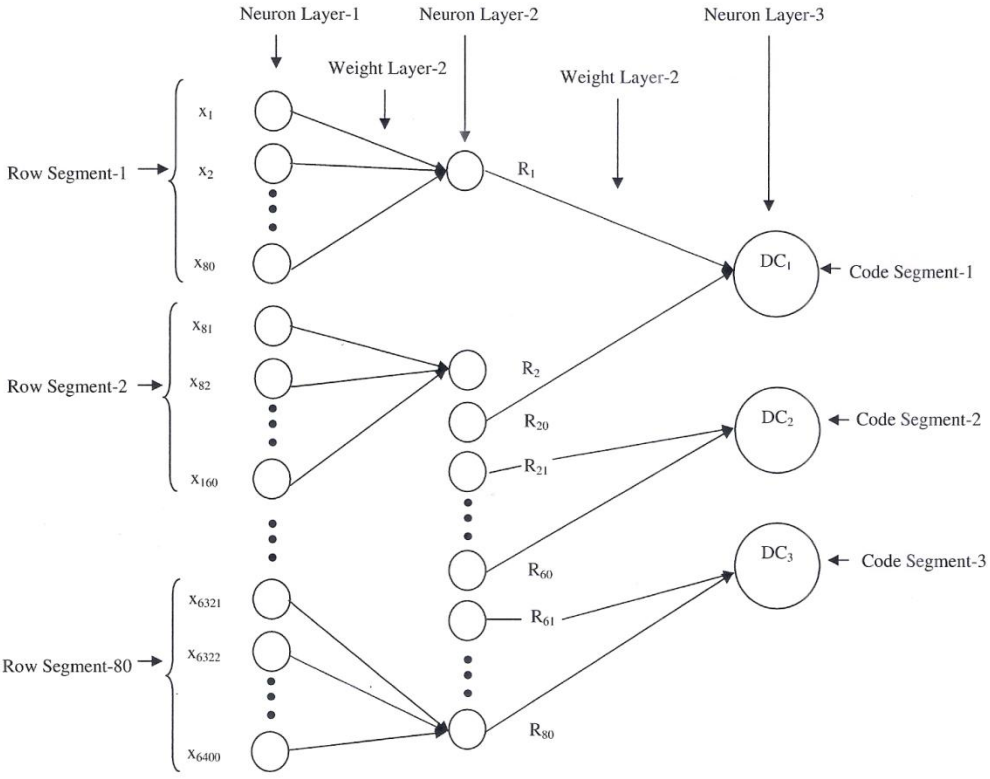


Fig. 3. Forming codes from the row segments

In case of an unsupervised type of neural net supervised learning of the net is not required. This method shows a far better convergence than the earlier used methods. A PDG-net, capable of identifying five characters, consists of three layers of neurons and two layers of fixed weights and one binary to decimal code converter, (Fig. 1). The neuron layers are represented by the vectors x, R_C and D_C. Vector x represents Neuron Layer-1, which is the first layer of the PDG-net and it consists of 6400 neurons, where the actual character is presented in the form of binary vector input. Row-wise segmentation of the input vector produces rows in the form as shown by the vector $[\{x_1, x_2, \ldots x_{80}\}, \{x_{81}, x_{82}, \ldots x_{160}\}, \ldots\ldots, \{x_{6321}, x_{6322}, \ldots x_{6400}\}]$ and Column-wise segmentation of the input vector produces columns in the form as shown by the vector $[\{x_1, x_{81}, \ldots, x_{6321}\}, \{x_2, x_{82}, \ldots, x_{6322}\}, \ldots, \{x_{80}, x_{160}, \ldots x_{6400}\}]$.

Vector R_C represents Neuron Layer-2 and it consists of 160 neurons out of which first 80 neurons represent the row segments and next 80 neurons represent the column segments. Vector D_C represents Neuron Layer-3 and it consists of only five neurons which represents the 5-bit unique code, where each neuron represents one bit of the code.

Similarly, there are two fixed weight layers of PDG-net. Weight Layer-1 represented by vector v_1 is present between the Neuron Layer-1 and Neuron Layer-2 and is always fixed to a value which is 1. Vector v_1 consists of 12800 elements, out of which first 6400 weight elements represents weights on row segments and next 6400 weight elements represents weights on column segments. Weight Layer-2 represented by vector v_2 is present between the Neuron Layer-2 and Neuron Layer-3 and is always fixed to a value which is 1.

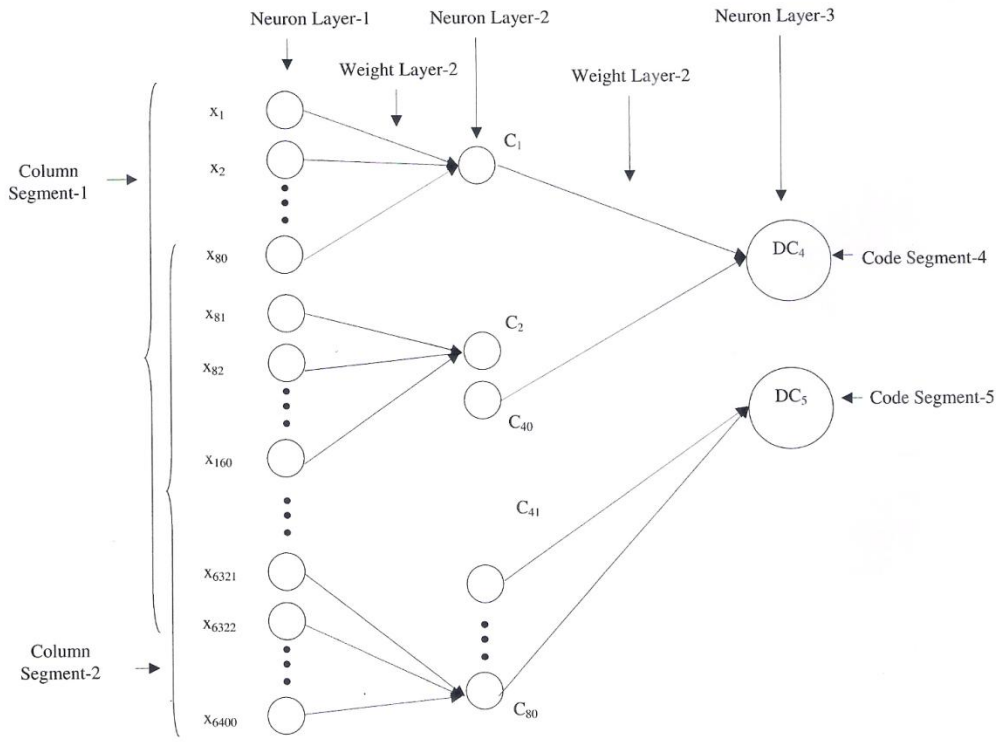


Fig. 4. Forming codes from the column segments

III. METHODOLOGY

A. *Preparation of Input Pattern*

First five English alphabets are written on a piece of paper and scanned with the help of a high definition scanner. Each scanned alphabet is saved in bmp b/w format and resized into a matrix of size 80 x 80 as shown in Fig. 2. The image matrix is converted into a two dimensional binary matrix, where the presence of the image on any pixel is represented be a binary 1 and absence of the image on any pixel is represented by a 0.

The two dimensional matrix is converted into a single dimensional input vector. The input vector is segmented row-wise first and then column-wise to form 80 rows and 80 columns of size. Each row and column is mapped into a layer containing 160 neurons out of which 80 neurons represent 80 rows and another 80 neurons represent 80 columns as shown in Neuron Layer-2 of Fig. 1.

B. *Code Generation Using Row Segments*

The binary character matrix of dimension 80 x 80 is segmented row-wise into 80 segments; each row segment consists of 80 neurons as shown in Fig. 3. There are three neuron layers. Neuron Layer-1 consists of 6400 neurons, which accepts the input character vector in row major form. Neuron Layer-2 consists of only 80 neurons, the activations of which are calculated as shown in Equation (1).

Several equations in the article numbers 3.2, 3.3 and 3.4, have been formulated through rigorous experiments and the results obtained thereof.

$$R_j = \{1 \text{ if } R\_tot_j >= \beta \text{ else } 0\} \quad (1)$$

R_j is the activation of the neuron j, produced at Neuron Layer-2 and β is the threshold value which is set to 20. R_j

calculates the density of dark pixels present in the row segments. β is set to 20 because it is found that average pixel density is approximately equal to 20 in a particular row segment, if a line or curve is present in that segment. R_tot_j is the total number of the dark pixels produced at that particular row, as shown in (2).

$$R\_tot_j = \sum_i v_1 * x_i \text{ , where } i = 1 \text{ to } 80 \quad (2)$$

Between Neuron Layer-1 and Neuron Layer-2 there is a Weight Layer-1 represented by vector v_1 which is fixed to 1. x_i is the i^{th} element of the Neuron Layer-1, which is the initial input pattern vector. Neuron Layer-3 consists of only 3 neurons, the activations of which are calculated as shown below.

$$DC_k = \{1 \text{ if } DC\_tot_K >= \alpha_r \text{ else } 0\} \quad (3)$$

Here, DC_k is the activation of the neuron k produced at Neuron Layer-3 and α_r is the threshold value. DC_k calculates the k^{th} bit of the code in the row segments. α_r is set to 4 as the presence of dense dark pixels consisting of positive value in four or more consecutive rows show the presence of a line or curve in that region. DC_tot_k is the total number of rows with dense pixels present in a particular code segment$_k$ as given below.

$$DC\_tot_k = \sum_j v_2 * R_j \quad (4)$$

Where j= 1 to m and m=30 if k=2 else 20

Fig. 3 demonstrates the generation of first three bits of the pattern generated. Between Neuron Layer-2 and Neuron Layer-3, there is Weight Layer-2 represented by vector v_2 with all its elements fixed to 1. The value of m has been set to 20 or 30 depending on the variation of gradient of pixel appearance in different row segments or portions of a character.

C. Code generation using column segments

The binary character matrix of dimension 80 x 80 is segmented column-wise into 80 segments as shown in Fig. 4, where each segment consists of 80 neurons. There are three neuron layers in the above figure. Neuron Layer-1 consists of 6400 neurons, which accepts the input character in column major form. Fig. 4 demonstrates the generation of last two bits of the pattern generated. Between Neuron Layer-1 and Neuron Layer-2 there is a Weight Layer-1 represented by vector v_1 which is fixed to 1. Neuron Layer-2 consists of only 80 neurons, the activations of which are calculated as shown in (5).

$$C_j = \{1 \text{ if } C\_tot_j >= \beta_c \text{ else } 0\} \quad (5)$$

C_j is the activation of the neuron j produced at Neuron Layer-2 and β_c is the column threshold value which is set to 20. C_j calculates the density of dark pixels present in the column segments. β_c is set to 20 because it was found that average pixel density is approximately equal to 20 in a particular column segment, if a line or curve is present in that segment. C_tot_j is the total density of the pixels produced at that particular column as shown in (6).

$$C\_tot_j = \sum_i v_1 * x_i \text{ , where } i = 1 \text{ to } 80 \quad (6)$$

Between Neuron Layer-2 and Neuron Layer-3 there is a Weight Layer-2 represented by vector v_2 which is fixed to 1. Neuron Layer-3 consists of only 2 neurons, the activations of which may be calculated as shown in (7).

$$DC_k = \{1 \text{ if } DC\_tot_K >= \alpha_c \text{ else } 0\} \quad (7)$$

DC_k is the activation of the neuron k produced at Neuron Layer-3 and α_c is the threshold value which is set to 4. α_c is set to 4 because it shows the presence of dense dark pixels in four or more consecutive columns, which shows the presence of a line or curve there. DC_k shows the k^{th} code bit of the column segemnts. DC_tot_k is the total number of columns with dense dark pixels present in a particular code segment as shown in (8)

$$DC\_tot_k = \sum_j v_2 * C_j \quad (8)$$

Where, j = n to m, n=1and m=40 if k=1, n=41 and m=80 if k=2

The bits obtained from the column segments of the characters forms the last two bits (i.e. 4^{th} and 5^{th}) of the five bit binary code segment.

The PDG-net produces unique binary codes for all the characters under consideration. The binary codes produced for each character is decoded by a 5-to-26 decoder, (Fig. 5). The output of the decoder indicates the identified alphabet as discussed earlier.

D. Testing the net

Initially, first five characters of English alphabet are used to test the net. Ten samples each of the characters are taken. The character samples taken from different individuals are presented to the PDG-net to test the performance of the net as shown in Fig. 6. For example, if the character sample 'C' is presented to the net, it generates an unique binary code, which is identical to the binary number generated by the character 'C', while unsupervised learning. Similarly the other alphabets are identified.

IV. RESULT ANALYSIS

'Matlab' software has been used to test the performance of the PDG-net. First five letters of English alphabet set are taken initially for testing. The response of the experiment shows excellent results.

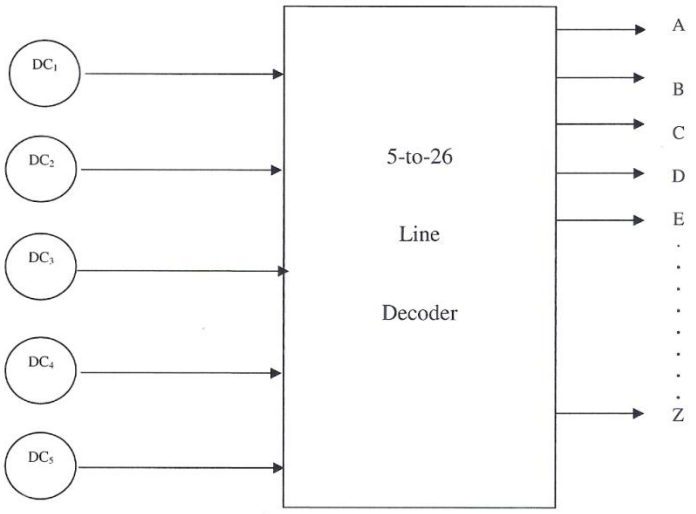


Fig. 5. 5-to-26 Line Decoder

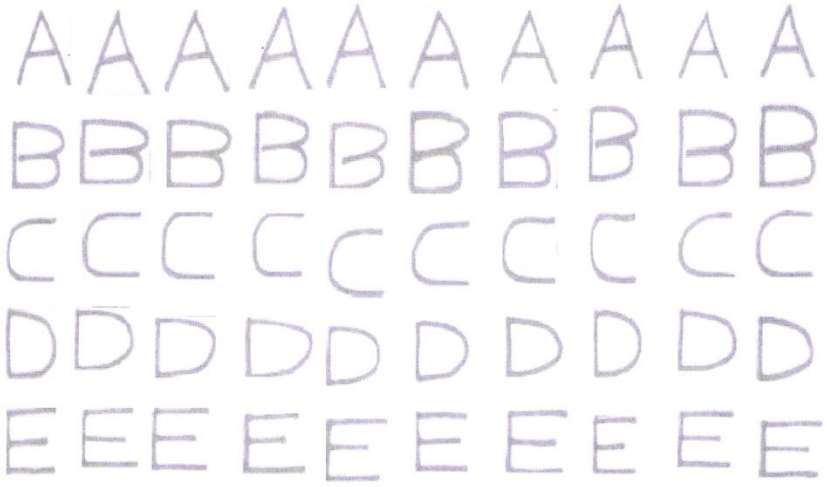


Fig. 6. Alphabet samples used to test the net

The different parameters of the PDG-net, under testing, are given as follows:

Number of neuron layers in PDG-net = 3

Number of weight layers in PDG-net = 2

Number of neurons in Neuron Layer-1 = 6400

Number of neurons in Neuron Layer-2 = 160

Number of neurons in Neuron Layer-3 = 5

Total Number of Neurons = 6565

One 5-to-26 decoder

Weight between Neuron Layer-1 & Neuron Layer-2 = 1

Weight between Neuron Layer-2 & Neuron Layer-3 = 1

Table 1, shows the results produced by the neural net when different sets of characters taken from different individuals.

Table 1. Accuracy of the net

S.No.	Alphabets	No. of Characters presented to the net	No. of characters identified	Percentage of identification
1	A	10	10	100%
2	B	10	10	100%
3	C	10	9	90%
4	D	10	9	90%
5	E	10	10	100%
Average				96%

V. Discussion

The result shows that PDG-net training, allows very fast convergence. Accuracy of identifying the characters is also very good and far better than the previously tried methods. Number of samples taken from different individuals identified is much better than the previously designed single layered nets [6], [7]. To enhance the accuracy of the net some more layers of neurons may be embedded in the net, which may increase the overhead of maintaining more number of neurons and also make the net little more complex. The overall performance of the PDG-net in terms of accuracy of identifying the character samples, it is found that PDG-net is a better choice over other nets used for the purpose. Total Number of Extra Neurons, if compared with earlier formulated nets is 155 and improvement in accuracy is 20%.

VI. Conclusion

PDG-net is a better approach as compared to the previously tried methods of recognition. Emphasis is on the presence of variable densities of the pixels at different segments of the characters, which leads to the presence of some common features among same alphabets. The gradient in the density of the pixels for different alphabets lead to the formation of unique codes for different alphabets which can be further trained to produce correct outputs. Segmentation of the matrix for finding out unique codes produces common features in the patterns and helps to obtain an eagle's eye in the identification of the pattern. This method has been tried for only five characters in the English alphabet set. The method can be refined in future to create a more generalized version.

Acknowledgment

Thanks to all my colleagues, students and my near ones for giving me support in completing this paper.

References

[1] G Robinson, "The multiscale technique", Available: http://www.netlib.org/utk/lsi/pcwLSI/text/node123.html, Page No (1), Mar 1995.

[2] TCTS Website, "Handwritten character recognition", Available: http://tcts.fpms.ac.bc/rdf/hcrinuk.htm, Page No (1), Accessed: 2010 (June).

[3] V Ganapathy and K L Liew, "Handwritten character recognition using multiscale neural network training technique", Proceedings of World Academy of Science, Engineering and Technology, Page No (29-37), May 2008.

[4] S Arora, D Bhattacharjee, M Nasipuri, D K Basu and M Kundu, "Combining multiple feature extraction techniques for handwritten Devnagri character recognition", IEEE Region 10 Colloquium and the Third ICIIS, Page No (1-6), Dec 2008.

[5] D Singh, S K Singh and M Dutta, "Handwritten character recognition using twelve directional feature input and neural network", International Journal of Computer Applications (0975 – 8887), Page No (82-85), 2010.

[6] R K Mandal and N R Manna, "Handwritten English character recognition using row-wise segmentation technique (RST)", International Symposium on Devices MEMS Intelligent Systems Communication (ISDMISC), Proceedings published by International Journal of Computer Applications® (IJCA), Page No (5-9), April 2011.

[7] R K Mandal and N R Manna, "Handwritten English character recognition using column-wise segmentation of image matrix (CSIM)", WSEAS Transactions on Computers, E-ISSN: 2224-2872, Volume 11, Issue 05, Page No (148-158), May 2012.

[8] G N Swamy, G Vijay Kumar, "Neural networks", Scitech, India Pvt Ltd, ISBN: 8183710557|9788183710558, 2007.

[9] L Fausett, "Fundamentals of neural networks, Architectures, Algorithms and Applications", Pearson Education, Fourth Edition, ISBN: 978813170053-2, Page No (19-114), 2009.

[10] A Roy and N R Manna, "Character recognition using competitive neural network with multi-scale training", UGC Sponsored National Symposium on Emerging Trends in Computer Science (ETCS 2012), Page No (17-20), Jan 2012.

[11] A Roy and N R Manna, "Competitive neural network as applied for character recognition ", International Journal of advanced research in Computer Science and Software Engineering, Volume 02, Issue 03, Page No (06-10), 2012.
[12] D S Rajput, R S Thakur and G S Thakur, "Clustering approach based on efficient coverage with minimum weight for document data", IJCSE, International Journal of Computer Sciences and Engineering, Volume 01, Issue 01, Page No (06-13), 2013.

AUTHORS PROFILE

Rakesh Kumar Mandal was born on February, 06, 1977 in Secunderabad, Andhra Pradesh. He completed his B.Sc (Non-Med) degree in the year 1997 from Guru Nanak Dev University, Amritsar, Punjab. He also completed D.C.C.T.M diploma in the year 1998 and Master of Computer Application (MCA) degree in the year 2001, from University of North Bengal. He cleared IBPS and joined a Nationalized Bank as an EDP Officer in the year 2005. He cleared the UGC-NET exam in Computer Science and Application in the year 2004 and joined University of North Bengal as a Lecturer in the year 2007. At present, He is an Assistant Professor in the Department of Computer Science & Application, University of North Bengal. He has already completed his Ph.D course work from the same University and preparing his Ph.D. thesis. He has already published four papers in International Journals, two papers in the proceedings of International Conferences, four papers in the proceedings of the National Conferences and one paper in a National Seminar. His area of interest is Handwriting recognition using Neural Networks.

Dr. N R Manna is a Professor of Department of Computer Science & Application, University of North Bengal, India. He was born on 31st day of May, 1954. He obtained his B.Tech and M. Tech degree in Radio Physics and Electronics from Calcutta University in 1975 and 1977 respectively. Prof Manna has about 30 years of teaching and research experience. The broad area of his research interest is Pattern Recognition, Expert System and Neural Network. He has published a number of research papers in National and International journals.