Chapter 5

Preprocessing of Handwritten Character Input Matrix

Preprocessing of input is necessary for the proper identification of the text data. Initially a paragraph of handwritten text has been taken as input and identified by the Artificial Neural Network (ANN) model after proper segmentation of characters.

Scanned image matrix of the text, obtained from a high definition scanner, is used as input for segmentation of characters. This image matrix is saved in a picture format (.bmp or .png). Proper preprocessing of this data is required before segmentation.

Input image of the text is converted into a binary matrix. The 1s present in the binary matrix are used to represent the part of the image whereas 0s are used to represent the blank spaces. After extraction, each character binary matrix is converted into one dimensional pattern vector and is presented to the ANN for training and testing purposes. The number of elements in the vector represents the number of neurons in the input layer of the ANN.

The binary matrix eventually formed from the character image is actually very large in size which makes the ANN very large and complicated. In order to simplify the circuit, the large two dimensional binary matrixes are compressed into smaller sized matrixes. A compression algorithm has also been designed to compress the large sized matrix into a smaller sized matrix. This compression algorithm has been designed in such a way that the originality of the image is preserved.

The main preprocessing of the data, after scanning the text, is noise removal. Other techniques like binarization, matrix compression, reference line estimation and image thinning [190] are applied at proper phases.

5.1 Noise Removal

Noise removal is a process where unwanted pixels in the texts are removed by applying some methods. In this work very little noise has been found in the text characters because a high definition scanner is used to scan the text samples. Meager

noise has been found in the individual characters after segmentation process, like unwanted curve or dots present in the characters. Noise has been removed by applying simple methods like removal of isolated and unwanted dots which are not found near the characters and reduction of unwanted curves as shown in Figure 5.1. In case of digitization of deteriorated old documents noise removal is very significant because deterioration marks creates lot of unwanted pixels on the documents.

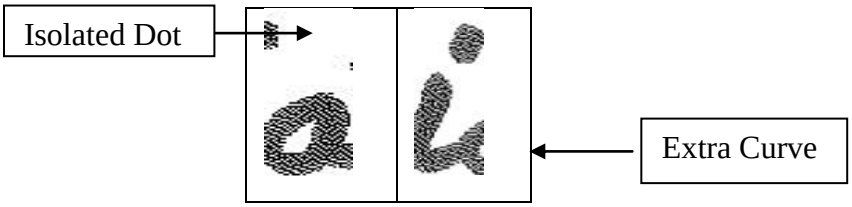

Figure 5.1: Removal of Isolated Dots and Extra Curve

5.2 Binarization

A text image has been scanned and saved in picture format. The image is converted into grayscale image, where the pixel density value lies between 0 and 255. Converting the grayscale image into a binary matrix makes the processing simple. The gray values can be converted into binary values by using thresholds. The binary value '0' is used for white pixel and '1' is used for the black pixel. The process of converting any image format into binary format is called binarization. Thresholds can be used locally or globally. In case of global threshold, one threshold value is used for the entire document image, based on an estimation of the background intensity level with that of the image using an intensity histogram [45]. Local or adaptive thresholds use different values for each pixel based on local area information [165]. Local thresholds are commonly used in images of varying intensities levels like satellite images. For the handwritten images global thresholds are used.

5.2.1 Converting the Character Image

In this work, the text is written on a piece of paper using black ball point pen/marker and scanned using a high definition scanner. The captured or scanned images

are in RGB scale, which are converted into grayscale format and later into binary for further processing. The binarization of image matrix has been carried out by the 'functions' provided by the MATLAB software. The 'functions' used to convert RGB to grayscale and grayscale into binary are rgb2gray() and dither() respectively. Using segmentation methods as developed and discussed in Chapter 6, single characters are extracted out of the text and enclosed within a rectangular boundary forming a two dimensional binary or bivalent matrix. The two dimensional matrix is further compressed to form a matrix of lower order using a compression algorithm developed in this work. Compression of the image reduces the number of elements in the matrix and makes the network smaller and simple. The two dimensional matrix is converted into a linear vector. All the 0s of the vectors are replaced by –1s for better learning. Using, 0s takes more epochs (iterations) and sometimes makes learning infinite because multiplying by 0 in the weight modification formula makes the weight stable.

5.3 Matrix Compression

The two-dimensional image matrixes, initially obtained, are very large in size. For a large vector it becomes very difficult to form an ANN, as numerous neurons are required. The developed compression algorithm compresses the large matrix into a smaller one without losing significant information. Character image matrix can also be compressed into smaller one by using the function imresize() provided by MATLAB.

5.3.1 Developed Compression Algorithm[♣]

A matrix can be compressed into a matrix of lower dimension in order to reduce the non significant elements i.e. unused spaces of the matrix. Character matrix 'A' has been compressed into a matrix 'A_COM' by using the Algorithm 5.1. Character Matrix 'A' has been split into 'n' uniform blocks. The dimension of each block 'A_BLOCK_i' is 'm x m'.

[♣] Based on author's Publication no 7 [Appendix B]

Algorithm 5.1 (Compression): Compression of a higher dimension matrix into a lower dimension matrix

STEP1. Read Character 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 of Step 3 Loop

[Where, ‘A_COM_i’ is the i^{th} element of the compressed matrix ‘A_COM’.]

$\Phi(A\_BLOCK_i) = 1$,

if there exists at least one element in ‘A_BLOCK_i’ which is 1

$\Phi(A\_BLOCK_i) = -1$,

if all the elements of ‘A_BLOCK_i‘are -1

STEP 4. STOP.

Example 1. Given matrix ‘A’ of size ‘4 x 4’

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

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

$$A\_BLOCK_1 = \begin{matrix} 1 & 1 \\ 1 & -1 \end{matrix} \qquad A\_BLOCK_2 = \begin{matrix} -1 & -1 \\ -1 & -1 \end{matrix}$$

$$A\_BLOCK_3 = \begin{matrix} -1 & 1 \\ -1 & -1 \end{matrix} \qquad A\_BLOCK_4 = \begin{matrix} 1 & -1 \\ -1 & -1 \end{matrix}$$

Applying Algorithm 5.1 following compressed matrix has been obtained (Figure 5.2).

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

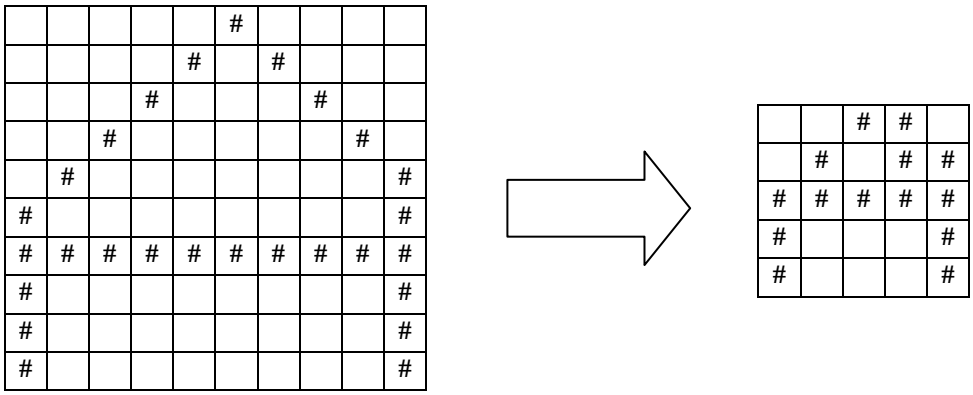

Figure 5.2: Conversion of Character ‘A’ from 10 x 10 into 5 x 5

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

Figure 5.3 represents character ‘A’ using a two dimensional matrix of order ‘10 x 10’ which is converted into a two dimensional matrix of order ‘5 x 5’.

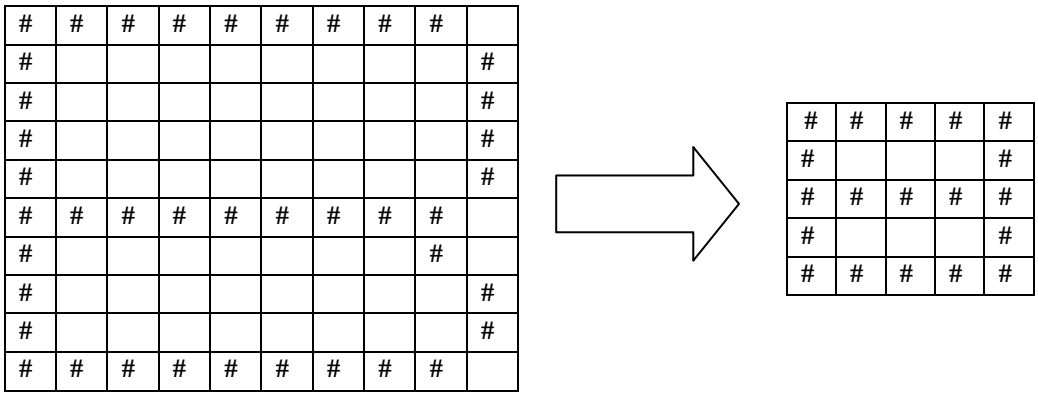

Figure 5.3: Conversion of Character ‘B’ from 10 x 10 into 5 x 5

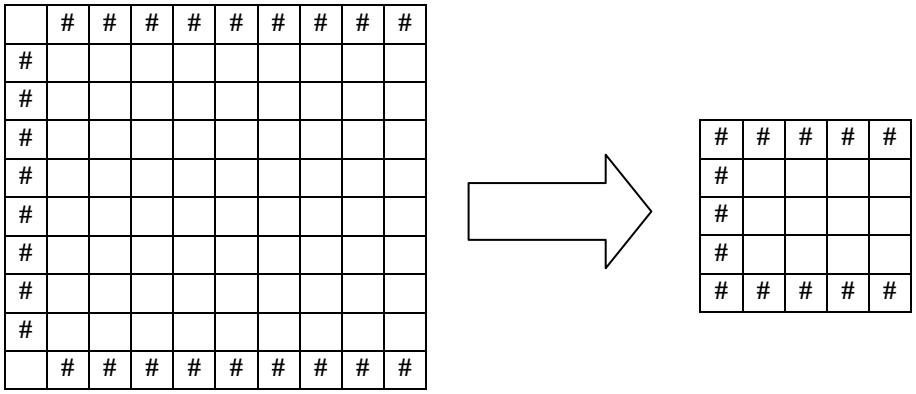

Figure 5.4: Conversion of Character ‘C’ from 10 x 10 into 5 x 5

It can be observed that the originality of the matrix has been retained to a great extent after using the compression algorithm. Similarly, the characters ‘B’, ‘C’, ‘D’ and ‘E’ are compressed using the same algorithm as shown in the Figure 5.3, 5.4, 5.5 and 5.6.

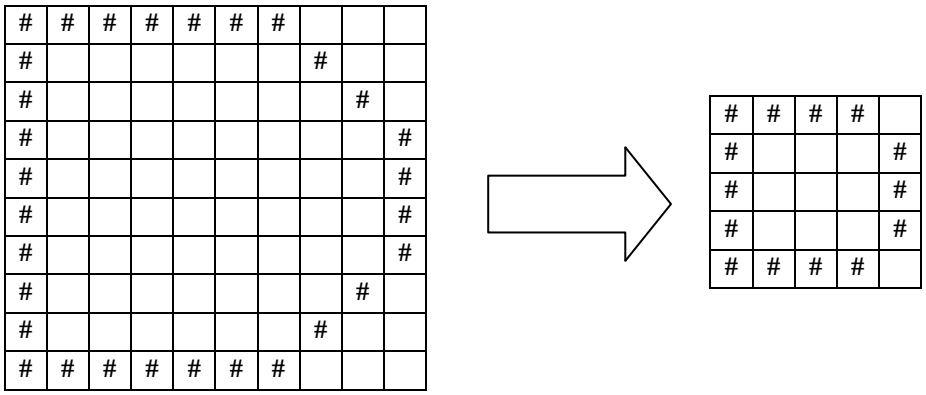

Figure 5.5: Conversion of Character ‘D’ from 10 x 10 into 5 x 5

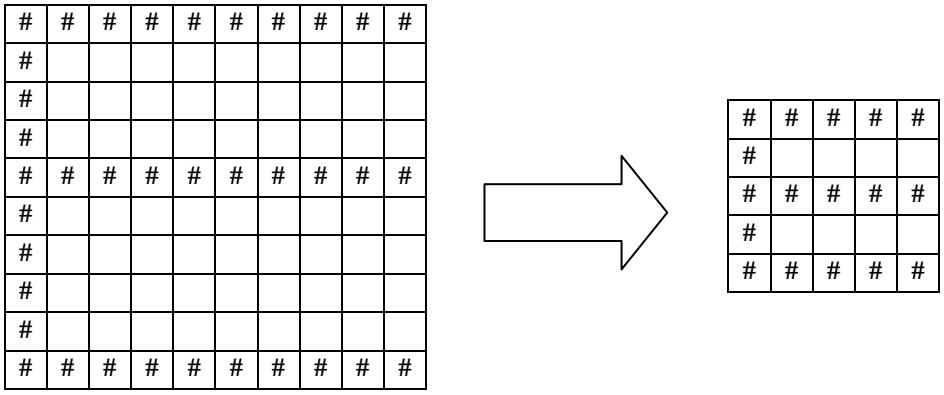

Figure 5.6: Conversion of Character ‘E’ from 10 x 10 into 5 x 5

5.4 Thinning

Image thinning is a process where the thick line of the character is reduced by removing pixels from the edges by maintaining the connectivity and keeping in mind that the thin shaped limbs of the character must not be shortened [190]. Here, a thinning process has been applied where the pixels are removed from the edges of the characters. Thinning process reduces the number of black pixels present in the character and makes the net simple.

5.5 Reference Line Estimation

This is another preprocessing technique which is helpful in determining the features in a text [190]. In this work lower baselines of the words are located to segment the characters by drifting upwards to find the first black pixel.

5.6 Conclusion

Preprocessing of input vector is significant in order to identify the text data. Preprocessing makes the input vector simple to process. It also makes the ANN simple. Various preprocessing techniques like noise removal, binarization, matrix compression, reference line estimation and image thinning are discussed. Noise removal removes the unwanted pixels from the image matrix. Binarization is the process of converting text image into binary vector. Matrix compression converts a matrix into lower order without losing major information. Reference line estimation is used to locate the baselines. Finaly, image thinning reduces the thickness of the line of the character image.