Chapter 9

Development of Multiple Layer ANNs

Sometimes it is necessary to add one or more additional hidden layers in single layer ANNs, in order to achieve better performance. Addition of hidden layers improves the performance of ANNs and helps them identify those test samples which are deviated from their original position to a larger extent. Actually, hidden layers are used to put more weights on those input segments that contribute the most in a character image. A vector segment having greater number of pixels than a vector segment which consists of comparatively lesser number of pixels contributes more in identifying a particular character. Hidden layers are also used to generate unique codes for different characters.

Multiple layer ANNs such as Arrow-Segmentation of Image Matrix (ASIM), Hoof Segmentation of Image Matrix (HSIM) and Pixel Density Gradient (PDG) Method are developed to identify handwritten characters

9.1 Character Recognition using Arrow Segmentation of Image Matrix (ASIM) ♣

A number of methods have already been developed and tried to recognize handwritten characters with ease and accuracy. The models developed using Artificial Neural Networks (ANNs) proved very efficient in doing so [161].

The input character image matrix has been compressed into a lower dimension matrix. This compressed matrix is segmented in such a way that takes a row as well as a column of the image matrix at a time, forming the shape of an arrow. This is called Arrow Segmentation of Image Matrix (ASIM). Several such arrows of different sizes are formed from the input image matrix and presented to the ANN for training.

The idea of forming arrow sized input segments, is to take the character pattern information from different portions of the character image matrix. There is a possibility of finding out common patterns among different alphabets written by different

♣ Based on author's Publication no. 3 [Appendix B]

individuals. An attempt has been made through this Arrow Segmentation of Image Matrix (ASIM) to recognize handwritten characters using ANN.

The overall program has been divided into three parts, compression of the input image matrix, segmenting the compressed matrix forming arrows of different sizes and training of the ANN. Finally, testing has been done by providing characters extracted out of the sample Paragraph Sets used to test the performances of different ANNs developed in this work.

9.1.1 Architecture of ASIM-net

ASIM-net consists of three layers of neurons and two layers of weights, as shown in Figure 9.1. Input neuron layer is represented by vector 'x', hidden neuron layer is represented by vector 'y' and the output neuron layer, also called the final decisive layer, is represented by vector $y\_f\_o_k$. Input neuron layer is divided into eight segments where each segment consists of varying number of neurons. The first segment consists of 19 neurons, the second segment consists of 17 neurons and so on, (Figure 9.2). Considering the last two segments that contains only three bits in the 9^{th} segment and one bit in the 10^{th} segment creates ambiguity problem as these two segments are almost identical for most of the character patterns. So, the last two segments are not considered in the ANN.

Hidden layer also consists of eight segments where each segment consists of ten neurons. Each segment in the hidden layer corresponds to a particular segment in the input layer. Length of the segments in the input layer depends upon the size of the arrow and length of the segment in the hidden layer is taken as ten in order to classify ten distinct alphabets. The output layer consists of ten neurons. The neurons present in each segment of the input layer are completely interconnected with the corresponding segment in the hidden layer.

The weight layer between input neuron layer and the hidden neuron layer has been represented by the vector 'w'. Weights taken between input layer and the hidden layer are initialized to zero.

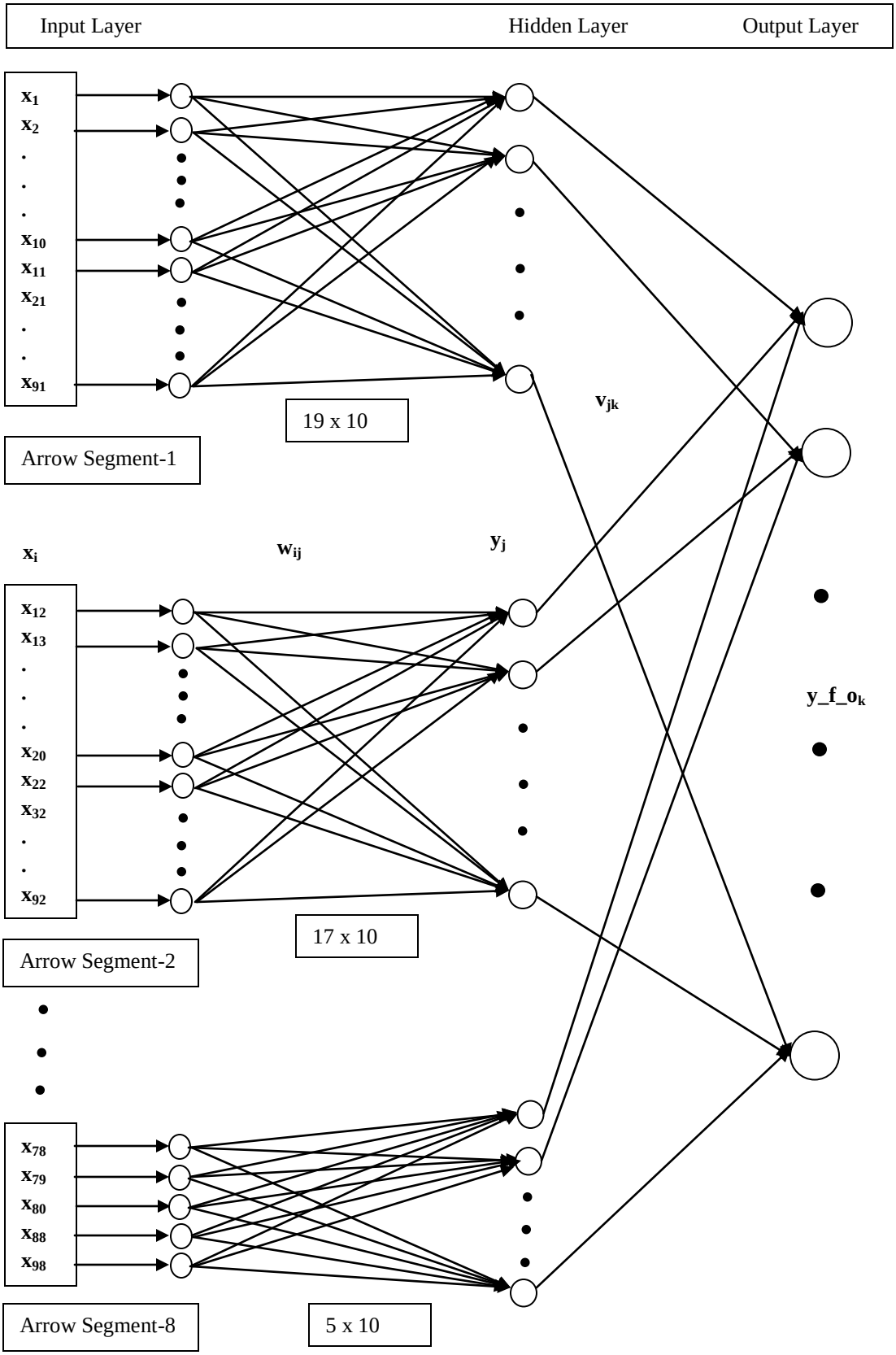

Figure 9.1: Neural Network Representing ASIM-net

The weight layer between hidden neuron layer and the output neuron layer is represented by the vector ‘v’. Weights taken between hidden neuron layer and the final decisive layer are fixed. First neuron of each segment in the hidden layer is connected to first neuron in the final decisive layer, second neuron connected to second neuron and so on. The neurons present in the hidden neuron layer of the first segment contributes with maximum weights to the final decisive layer as it is trained by maximum number of neurons in the input layer, second segment contributes with comparatively less weight and so on.

9.1.2 Methodology of ASIM-net

Each binary input character image matrix of dimension ‘80 x 80’ obtained after preprocessing is further compressed into a comparatively lower order matrix of dimension ‘10 x 10’.

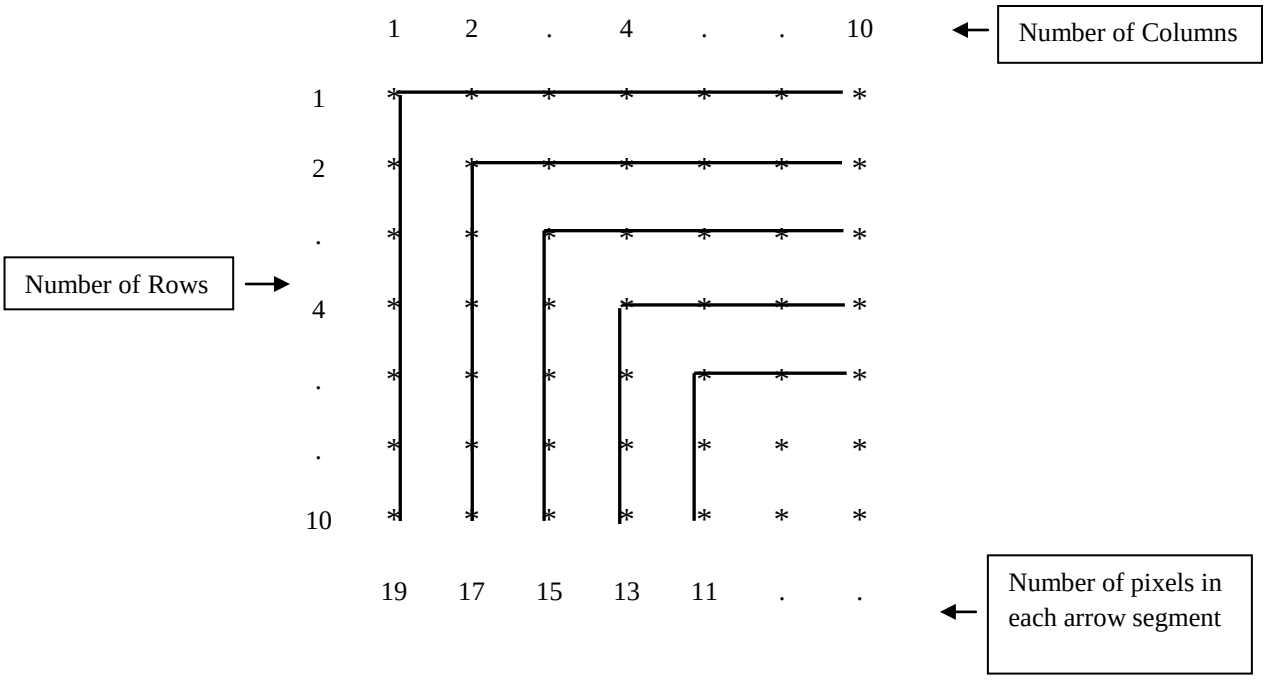

Figure 9.2: Conceptual diagram of ASIM-net

This matrix has been segmented in such a way that each segment takes one row from the top and one column from the extreme left (one pixel less than the row taken) from the top forming an arrow so that pixels contained in the first segment is 10 + 9 = 19 as shown in Figure 9.2. Remaining portion of the matrix vector is segmented in the same way. All these segments are presented to the ANN one by one for training purpose.

Once the standard weights are found, test characters are presented to the trained ANN. While, training the ANN, it is found in some characters that the arrow segments containing very less number of neurons produces an infinite loop indicating similar patterns for some alphabets. Training of two identical patterns for different targets never allows the net converge to produce correct results. If we just change one bit in the identical patterns, that does not make any huge difference in the originality of the pattern but removes the ambiguity. There are ten neurons for identifying ten distinct characters in each output segment. The neurons in the hidden layers are further connected to ten neurons in the output layer by different weights where each neuron identifies a particular character.

The segment consisting of maximum number of elements of the image matrix vector will contribute with maximum weight to identify the character. The final neuron with the maximum value, which is called final decisive output, will be the winner to identify the character.

9.1.2.1 Removing Ambiguity

It has been found that two identical patterns can’t be trained to identify two different output patterns and so it is necessary to remove identical patterns present in different characters. All the character patterns presented to the ANN are compared mutually to find any presence of identical pattern. This is done by comparing the first character pattern to be presented to the ANN with the remaining patterns bitwise. If any pattern is found identical to the first pattern, then the rightmost bit of that pattern is complimented to make it different from the first one. Same process is repeated with the second character pattern with the remaining patterns and so on. In this process two non identical patterns may become identical. To ensure this mutual comparisons are performed till all the character patterns are found non identical.

9.1.3 Training the ASIM-net

The sample characters used to train all the ANNs developed and discussed earlier, are used to train ASIM-net also. The weights between input and hidden layer are initially set to zero. Each segment of the ASIM-net, (Figure 9.1) has been trained using perceptron learning rule [63]. After training, the ANN for few epochs, the standard weights are obtained. These weights thus obtained remain same for the next few epochs and is considered as the final standard weights for testing the ANN.

9.1.3.1 Training using Modified Input Data Sets

Arrow segmentation (or scanning) is done in such a peculiar way, which may sometimes skip some important features present in those portions of the character matrix which lies outside the arrow. To solve this problem the ANN are trained using character matrixes with different orientations. This is permissible as character is no longer a character now, but may be thought of a pattern of pixels. It is found that this helps much in obtaining the features lying out of the scope of the arrow segments in the character matrix. So, instead of segmenting the character matrix from only one corner if considered from all other three corners also then possibility of obtaining common features increases. Thus, input character matrix A obtained after compression is transposed in the following way:

$$B = A^T$$
$$C = B^T$$
$$D = C^T$$

and, presented to the net for training or testing as the case may be. Weight vectors W_1, W_2, W_3 and W_4 are obtained by presenting matrixes A, B, C and D matrices to the ANN, separately in the training phase.

This type of modified input vectors are not at all significant in single layer ANNs because the segments are considered row-wise, column-wise or block-wise covering the whole character matrix leaving very less space to skip features in any portion and so omitted.

9.1.4 Testing the ASIM-net

An extra layer with fixed weights has been implemented here to test the character sets extracted out from the Paragraph Sets. The final output layer, called the final decisive layer, consists of ten neurons. The neuron having the maximum value, also called the decisive output factor, wins to choose a character. Each neuron recognizes a particular character, like neuron-1 identifies the first character; neuron-2 identifies the second character and so on. The neurons responsible for recognizing a character is connected to the neuron in the decisive layer with different weights. Maximum weight is given to the link between the segments which is having maximum number of neurons. A decisive output factor is calculated for each neuron in the decisive output layer. The neuron with the maximum decisive output factor ($y\_f\_o_k$) is the winner to identify the character.

$$y\_f\_o_k = 0.10*y_1(k) + 0.09*y_2(k) + 0.08*y_3(k) + 0.07*y_4(k) + 0.06*y_5(k) + 0.05*y_6(k) + 0.04*y_7(k) + 0.03*y_8(k) \quad \text{.....Equation 9.1}$$

$y\_f\_o_k$ is the decisive output factor for the k^{th} neuron in the output layer. $y_1(k)$, $y_2(k)$ …. $y_8(k)$ are outputs produced by the different segments of the middle layer by ten characters and 0.10, 0.09 …. 0.03 are the weights of different segments of the hidden layer connecting to the output layer as shown in equation 9.1. The weights are chosen depending upon the number of elements in a particular segment. For example, segment-1 having 19 elements, the weight chosen is 0.10, which is calculated as shown in equation 9.2.

$$v = \lceil \text{number of elements}/2 \rceil /100 \quad \text{.....Equation 9.2}$$

19/2 gives 9.5. In order to simplify the calculation ceiling of the number is taken. The result is further divided by 100 in order to reduce the weights. Low values of weights are taken in order to simplify the calculations. The weights are chosen in this way because more the number of neurons in a segment more is the information the segment is contributing to the ANN to identify a character.

Algorithm 9.1 (ASIM): Working of ASIM-net

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 8.2 'Compression' on each 'm x m' matrix and compress into 'n x n' matrix, where 'm>n'.

STEP 3. Form 'n-2' number of arrow segments from each compressed matrix in such a way, so that each segment consists of a row of length 'q' and a column of length 'q-1' starting from the topmost row and extreme left column for the input layer. [The value of 'q' is decremented by 1 for the next segment.]

STEP 4. Form 'n-2' number of segments each consisting of 10 neurons for the hidden neuron layer.

STEP 5. Completely interconnect all the neurons of each segment in the input layer to all the neurons of each corresponding segment in the hidden layer.

STEP 6. Initialize weight matrix 'w' between input layer and hidden layer to all 0s.

STEP 7. Apply perceptron learning rule to each segment and find out the standard weights.

STEP 8. Add an additional layer with fixed weights called the final decisive layer consisting of ten neurons. [From each segment in the hidden layer all the ten neurons are connected to the ten neurons in the output layer with fixed as shown in Figure 9.1.]

STEP 9. Present the compressed test characters to the ASIM-net and find out '$y\_f\_o_k$' for each 'k' as shown in Equation 9.1.

STEP 10. Depending upon the value of '$y\_f\_o_k$' identify the character.

STEP 11. STOP.

9.1.4.1 Testing using Modified Input Data Sets

It is observed experimentally that the chances of finding the maximum decisive output factor ($y\_f\_o_k$), responsible for identifying the character increase, if different ($y\_f\_o_k$)s are obtained by presenting transposed matrixes with corresponding standard weights to the ANN. The test samples are transposed thrice to obtain four input vectors. The standard weights already obtained while training is used to find out the maximum decisive output factors ($y\_f\_o_k$) for these four input vectors. The average value of the four '$y\_f\_o_k$' yields a better value for identification. It increases the efficiency of ANN to a great extent. Algorithm 9.2 depicts the new scheme of training and testing of ASIM-net.

Algorithm 9.2 (Modify_Input): Modification of Input Vectors

STEP 1. Let us consider the equivalent binary matrix 'A' of the character to be trained.

STEP 2. Find out the matrixes 'B', 'C' and 'D' in the following way: [Where each new matrix is the transpose of previous one]

$B = A^T$

$C = B^T$

$D = C^T$

and presented to the net for obtaining different sets of standard weights

STEP 3. Present the matrixes 'A', 'B', 'C' and 'D' to the ANN separately for training and generate corresponding weights 'W_1', 'W_2', 'W_3' and 'W_4'.

STEP 4. Consider the test character matrix 'A_TEST' and find out 'B_TEST', 'C_TEST' and 'D_TEST' by using transpose method.

STEP 5. Find the values of '$y\_f\_o_k$' for 'A_TEST', 'B_TEST', 'C_TEST' and 'D_TEST'

separately while testing by using the weights 'W_1', 'W_2', 'W_3' and 'W_4'.

STEP 6. Calculate the following:

$y\_f\_o_k(AVG)=[y\_f\_o_k(A\_TEST)+y\_f\_o_k(B\_TEST)+y\_f\_o_k(C\_TEST)+y\_f\_o_k(D\_TEST)]/4$

[Where, $y\_f\_o_k(AVG)$ is the average value of decisive output factor and $y\_f\_o_k(A\_TEST)$, $y\_f\_o_k(B\_TEST)$, $y\_f\_o_k(C\_TEST)$ and $y\_f\_o_k(D\_TEST)$ are the decisive output factors generated by matrixes obtained by transposing.]

STEP 7. STOP.

9.1.5 Performance Analysis of ASIM-net

MATLAB software has been used to test the accuracy of the ASIM-net. The characters extracted from the Paragraph Sets (Section 6.2.6) are taken for testing the performance of the ASIM-net also. The response of the experiment shows encouraging results. The Paragraph Sets are actually text paragraphs taken from different individuals for the testing purpose.

Neural Network Parameters used in the Experiment:

Number of neurons in the input vector = 6400
Number of neurons in the compressed vector = 100
Number of input pattern segments = 8
Number of pattern segments in the hidden layer= 8
Number of neurons in first input pattern segment = 19
Number of neurons in second input pattern segment = 17
Number of neurons in third input pattern segment = 15
Number of neurons in fourth input pattern segment = 13
Number of neurons in fifth input pattern segment = 11
Number of neurons in sixth input pattern segment = 9
Number of neurons in seventh input pattern segment = 7

Table 9.1: Identification of Characters of Paragraph Set-1 using ASIM-net

Alphabets	Number of Segmented characters	Correctly recognized	Percentage of accuracy
'C'	89	75	84.27%
'a'	95	84	88.42%
't'	80	70	87.50%
's'	92	87	94.57%
'A'	100	92	92.00%
'n'	80	71	88.75%
'd'	90	74	82.22%
'D'	92	61	66.30%
'o'	98	78	79.59%
'g'	70	64	91.43%
Total	**886**	**756**	**85.33%**

Number of neurons in eighth input pattern segment = 5

Number of Neurons in each segment of the hidden layer = 10

Dimension of weight matrix between the first input-hidden segment = 19 x 10

Dimension of weight matrix between the second input-hidden segment = 17 x 10

Dimension of weight matrix between the third input-hidden segment = 15 x 10

Dimension of weight matrix between the fourth input-hidden segment = 13 x 10

Dimension of weight matrix between the fifth input-hidden segment = 11 x 10

Dimension of weight matrix between the sixth input-hidden segment = 9 x 10

Dimension of weight matrix between the seventh input-hidden segment = 7 x 10

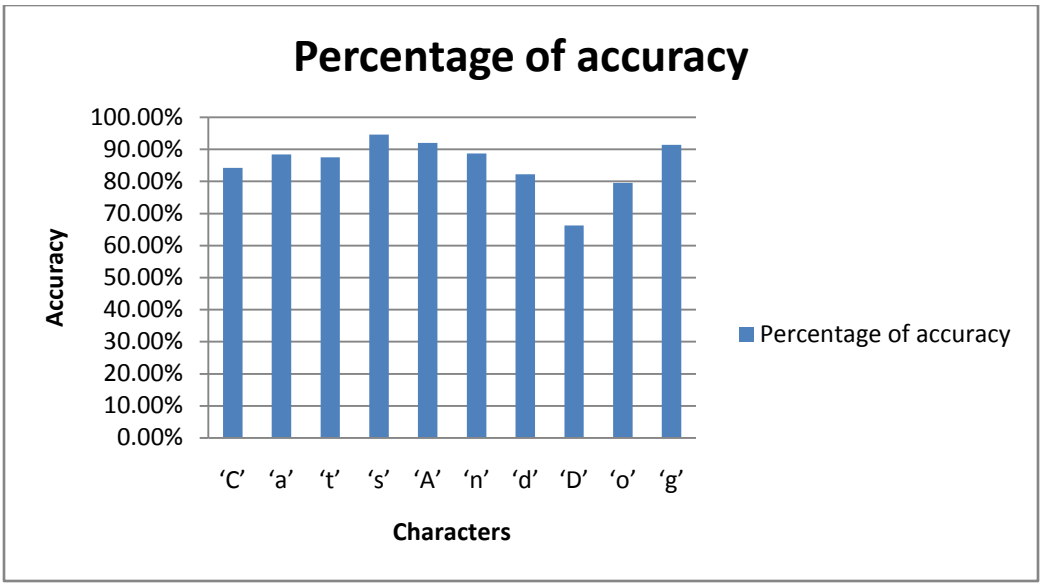

Figure 9.3: Percentage of Accuracy of Paragraph Set-1 by ASIM-net

Table 9.2: Identification of Characters of Paragraph Set-2 using ASIM-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'M'	100	84	84.00%
'y'	71	61	85.91%
'F'	91	85	93.41%
'i'	60	50	83.33%
'v'	73	61	83.56%
'e'	89	72	80.90%
'C'	90	80	88.89%
'o'	92	88	95.65%
'w'	71	51	71.83%
's'	83	69	83.13%
Total	**820**	**701**	**85.49%**

Dimension of weight matrix between the eighth input-hidden segment = 5 x 10

Number of neurons in output layer = 10

Dimension of weight matrix between the hidden-output layer = 8 x 10

Training Algorithm used = Perceptron

Number of Epochs = 3

Threshold (θ) = 0.2

Learning Rate (α) = 1

The accuracy of the system is shown in Table 9.1 and Table 9.2.

Accuracy generated by Paragraph Set 1 = 85.33%

Accuracy generated by Paragraph Set 2 = 85.49%

Average Accuracy of ASIM-net= 85.41%

Figure 9.3 displays the column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-1 by ASIM-net. Figure 9.4 displays column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-2 by ASIM-net.

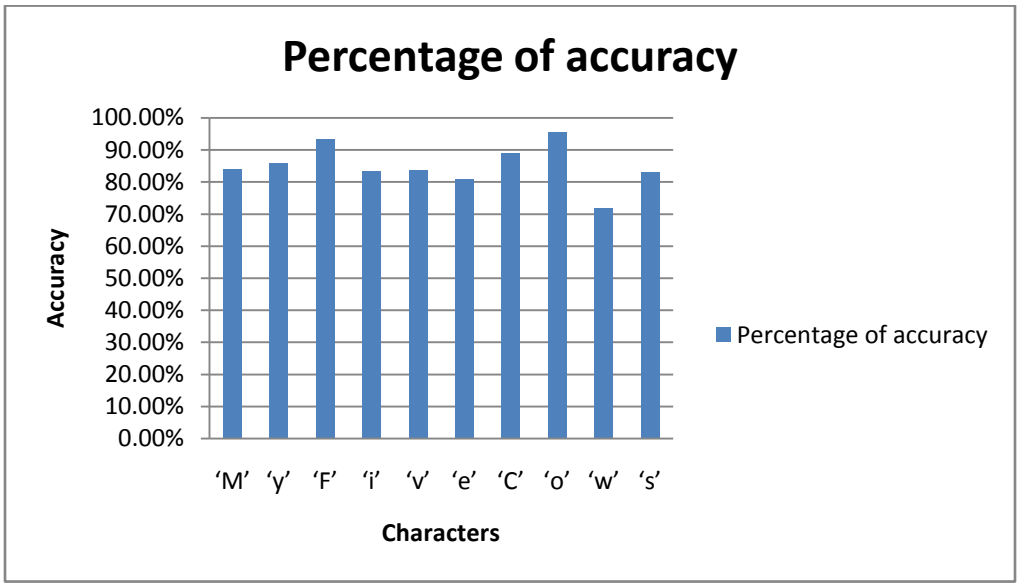

Figure 9.4: Percentage of Accuracy of Paragraph Set-2 by ASIM-net

9.2 Character Recognition using Hoof Segmentation of Image Matrix (HSIM)♣

Hoof Segmentation of Image Matrix (HSIM) approach has been used to simplify the feature extraction method to identify varying character patterns. Segments can be formed in different shapes and presented to the ANN for training and testing purpose [63, 191]. Segments are generated from different characters in horse hoof like structures and presented to a multiple layer ANN, developed here, for training and testing. Common features are extracted out by identifying different horse hoof like shapes generated from a character image matrix.

The input image matrix is compressed into a lower dimension matrix and segmented in such a way that takes one row and two columns of the image matrix forming the shape of a horse hoof. Many hoofs of different sizes are generated from the input image matrix and presented to the ANN for training.

The overall program has been divided into three parts, compression of the input image matrix, segmenting the compressed matrix forming hoof like segments of different sizes and training the ANN. Finally, testing has been done by providing characters extracted out of the same Paragraph Sets as used to test the performances of different ANNs developed here.

9.2.1 Architecture of HSIM-net

HSIM-net (Figure 9.5) consists of three layers of neurons and two layers of weights. Input neuron layer is represented by vector 'x', hidden neuron layer is represented by vector 'y' and the output neuron layer also called the final decisive layer is represented by vector $y\_f\_o_k$ as shown in Figure 9.5.

Input layer has been divided into five segments where each segment consists of varying number of neurons. The first segment consists of 28 neurons, the second segment consists of 24 neurons and so on. Hidden layer also consists of five segments where each segment consists of ten neurons. Each segment in the hidden neuron layer corresponds to a particular segment in the input neuron layer.

♣ Based on author's Publication no. 13 [Appendix B]

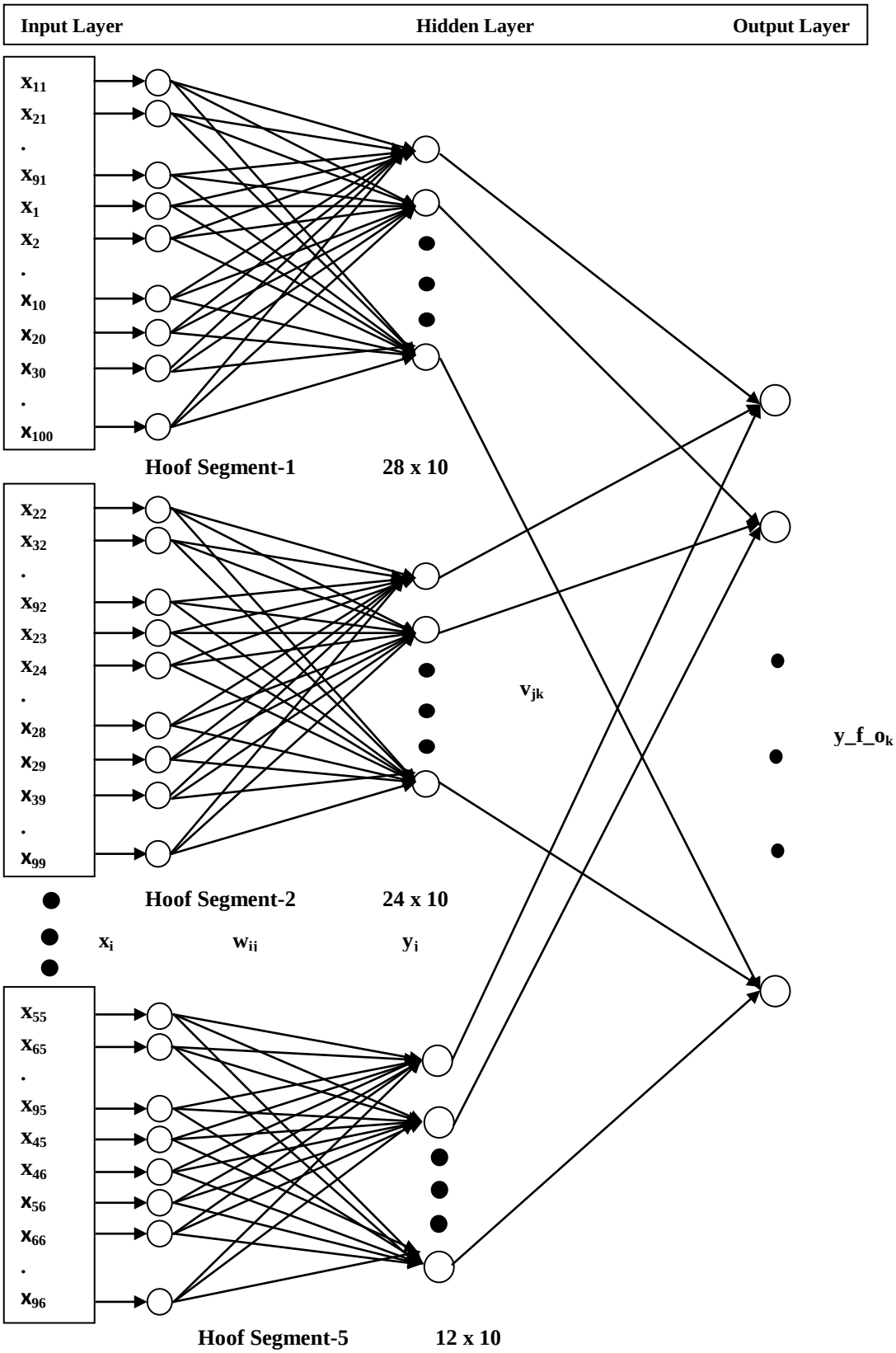

Figure 9.5: Neural Network Representing HSIM-net

Length of the segments in the input neuron layer depends upon the size of the hoof like structures and length of the segment in the hidden layer is taken as ten in order to classify ten distinct alphabets. The output neuron layer consists of ten neurons. The neurons present in each segment of the input neuron layer are completely interconnected to the corresponding segment in the hidden neuron layer.

The weight layer between input neuron layer and the hidden neuron layer has been presented by the vector 'w'. Weights taken between input neuron layer and the hidden neuron layer are initialized to all 0s. The weight layer between hidden layer and the output layer is represented by the vector 'v'. The weights taken between hidden layer and the final decisive layer have been fixed. First neuron of each segment in the hidden layer is connected to first neuron in the final decisive layer, second neuron connected to second neuron and so on.

Neurons present in the hidden layer of first segment contribute maximum weights to the final decisive layer as it is trained by maximum number of neurons in the input layer. Neurons of second segment contribute comparatively less weight and so on.

9.2.2 Methodology of HSIM-net

Each preprocessed binary input character image matrix of dimension '80 x 80' has been further compressed into a comparatively lower order matrix of dimension '10 x 10'. This matrix is segmented taking one row from the top, one column from the extreme left (one pixel less than the row taken) and one column from the extreme right (one pixel less than the row taken) forming an hoof like structure so that pixels contained in the first segment is 10 + 9 + 9 = 28 as shown in Figure 9.6. Remaining portion of the matrix vector is segmented in the same way.

All these segments are presented to the ANN one by one for the training purpose. Once the standard weights are found, test characters are presented to the trained ANN for the testing purpose. Ambiguities due to identical patterns can be removed by applying the method as used in ASIM-net. There are ten neurons for identifying ten distinct characters in each output segment. The neurons in the hidden neurons layers are further connected to ten neurons in the output layer by different weights where each neuron identifies a particular character.

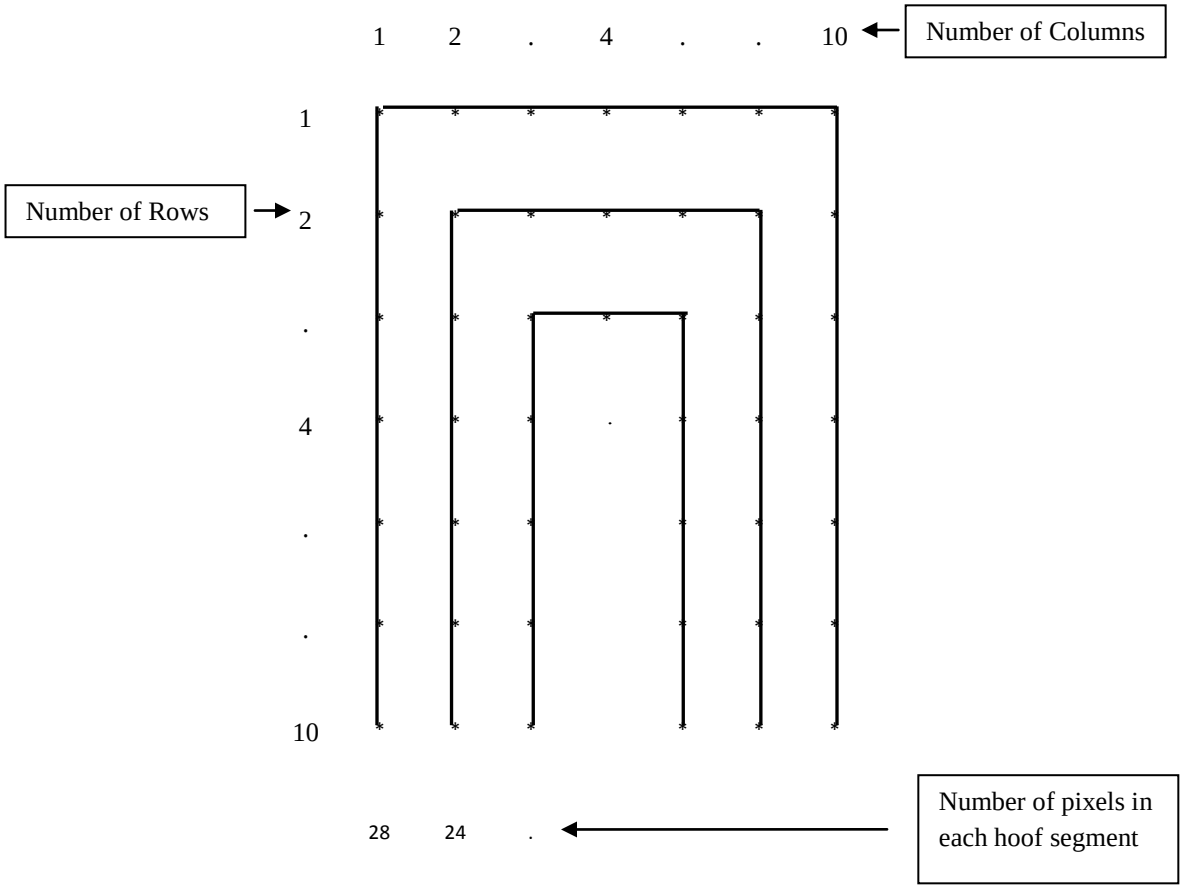

Figure 9.6: Conceptual diagram of HSIM-net

The segment consisting of maximum number of neurons of the image matrix vector contributes maximum weight to identify the character. The final neuron with the maximum value, which is called final decisive output, is considered as the winner to identify the character.

9.2.3 Training the HSIM-net to obtain the Standard Weights

The characters used to train all the ANNs developed and discussed here have been used to train HSIM-net. The weights between input and hidden layers are initially set to all 0s. Each segment of the HSIM-net, Figure 9.5, has been trained using perceptron learning rule [63]. After training the ANN for few epochs, the final weights are obtained. The final weights obtained remain same for the next few epochs and has been considered as the standard weights for testing purpose.

9.2.3.1 Training using Modified Input Data Sets

The ANN is trained using the method as discussed in section 9.1.3.1 to enhance the performance of the net.

9.2.4 Testing the HSIM-net

Another layer with fixed weights has been implemented to test the character sets extracted out from the Paragraph Sets (Section 6.2.6). The final output layer, called final decisive layer, consists of ten neurons. The neuron having the maximum value, also called decisive output factor, wins to choose a test character. Each neuron recognizes a particular test character, like neuron-1 identifies the first character, neuron-2 identifies the second character and so on. The neurons responsible for recognizing a character is connected to the neuron in the decisive layer with different weights. Maximum weight has been given to the link between the segments which is having maximum number of neurons. The decisive output factor has been calculated for each neuron in the decisive output layer. The neuron with the maximum decisive output factor is the winner to identify the character.

$y\_f\_o_k$ is the decisive output factor for the k^{th} neuron in the output layer. $y_1(k)$, $y_2(k)$ …. $y_5(k)$ are outputs produced by the different segments of the hidden neuron layer by ten characters and 0.10, 0.08, …. , 0.04 are the weights of different segments of the hidden neuron layer connecting to the output neuron layer as shown in equation 9.3.

$$y\_f\_ok = 0.10*y1(k) + 0.08*y2(k) + 0.07*y3(k) + 0.06*y4(k) + 0.04*y5(k) \quad \text{…Equation 9.3}$$

Depending on the number of neurons in a particular segment weights are chosen. For example, segment-1 having 28 neurons, the chosen weight is 0.10. which is calculated as shown in Equation 9.4. Here, 28/3 gives 9.33. In order to simplify the calculation ceiling of the number is taken which is 10.

$$v = \lceil \text{number of elements}/3 \rceil /100 \quad \text{…..Equation 9.4}$$

The result is further divided by 100 in order to reduce the weights. Low values of weights are taken only to simplify the calculations. Weights are chosen in this way because it is found more the number of neurons in a segment more is the information the segment is contributing to the ANN, used to identify a character.

Algorithm 9.3 (HSIM): Working of HSIM-net

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 8.2 'Compression' on each 'm x m' matrix and compress into 'n x n' matrix, where 'm>n'.

STEP 3. Form hoof structure like segments from each compressed matrix in such a manner so that each segment consists of a row of length 'q' and two columns of length 'q-1' starting from the topmost row and extreme left and right columns for the input neuron layer. [Where, the value of 'q' is decremented by 1 for the next segments.]

STEP 4. Form an equal number of segments as in input neuron layer, each of length 10, for the hidden layer.

STEP 5. Completely interconnect all the neurons of each segment in the input neuron layer to all the neurons of each corresponding segment in the hidden neuron layer.

STEP 6. Initialize weight matrix 'w' between input neuron layer and hidden neuron layer to all 0s.

STEP 7. Apply perceptron learning rule to each segment and find out the standard weights.

STEP 8. Add an additional neuron layer with fixed weights called the final decisive layer

consisting of ten neurons. [Where, from each segment in the hidden layer all the 10 neurons are connected to the ten neurons in the output layer with fixed as shown in Figure 9.5.]

STEP 9. Present the compressed test characters to the HSIM-net and find out '$y\_f\_o_k$' for each 'k' as shown in Equation 9.3.

STEP 10. Depending upon the value of '$y\_f\_o_k$' identify the character.

STEP 11. STOP.

9.2.4.1 Testing using Modified Input Data Sets

The ANN is tested using the method as discussed in section 9.1.4.1 to enhance the performance of the net.

9.2.5 Performance Analysis of HSIM-net

MATLAB software has been used to test the accuracy of the HSIM-net. The response of the experiment shows satisfactory results.

Neural Network parameters used in the experiment:

Number of neurons in the input vector = 6400
Number of neurons in the compressed vector = 100
Number of input pattern segments = 5
Number of pattern segments in the hidden layer= 5
Number of neurons in first input pattern segment = 28
Number of neurons in second input pattern segment = 24
Number of neurons in third input pattern segment = 20
Number of neurons in fourth input pattern segment = 16
Number of neurons in fifth input pattern segment = 12
Dimension of weight matrix between the first input-hidden segment = 28 x 10
Dimension of weight matrix between the second input-hidden segment = 24 x 10

Table 9.3: Identification of Characters of Paragraph Set-1 using HSIM-net

Alphabets	Number of Segmented characters	Correctly recognized	Percentage of accuracy
'C'	89	65	73.03%
'a'	95	88	92.63%
't'	80	69	86.25%
's'	92	81	88.04%
'A'	100	96	96.00%
'n'	80	75	93.75%
'd'	90	82	91.11%
'D'	92	78	84.78%
'o'	98	76	77.55%
'g'	70	63	90.00%
Total	**886**	**773**	**87.25%**

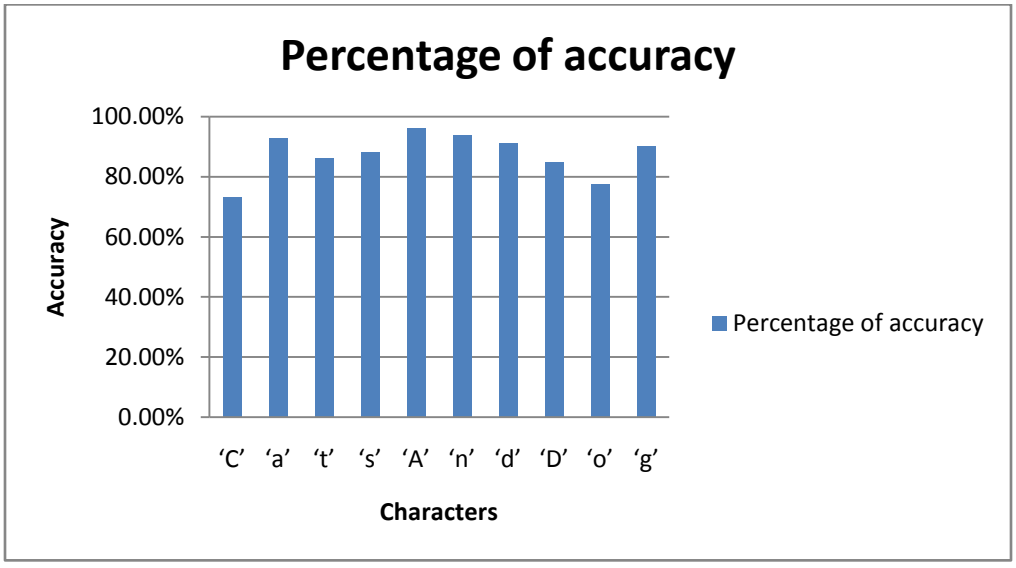

Figure 9.7: Percentage of Accuracy of Paragraph Set-1 by HSIM-net

Dimension of weight matrix between the third input-hidden segment = 20 x 10

Dimension of weight matrix between the fourth input-hidden segment = 16 x 10

Dimension of weight matrix between the fifth input-hidden segment = 12 x 10

Dimension of weight matrix between the hidden-output layer = 5 x 10

Training Algorithm used = Perceptron

Number of Epochs = 3, Threshold (θ) = 0.2, Learning Rate (α) = 1

The accuracy of the system is shown in Table 9.3 and Table 9.4.

Accuracy generated by Paragraph Set 1 = 87.25%

Accuracy generated by Paragraph Set 2 = 85.12%

Average Accuracy of HSIM net= 86.19%

Figure 9.7 displays the column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-1 by HSIM-net. Figure 9.8 displays the column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-2 by HSIM-net.

Table 9.4: Identification of Characters of Paragraph Set-2 using HSIM-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'M'	100	91	91.00%
'y'	71	66	92.96%
'F'	91	78	85.71%
'i'	60	45	75.00%
'v'	73	61	83.56%
'e'	89	69	77.53%
'C'	90	77	85.56%
'o'	92	89	96.74%
'w'	71	57	80.28%
's'	83	65	78.31%
Total	**820**	**698**	**85.12%**

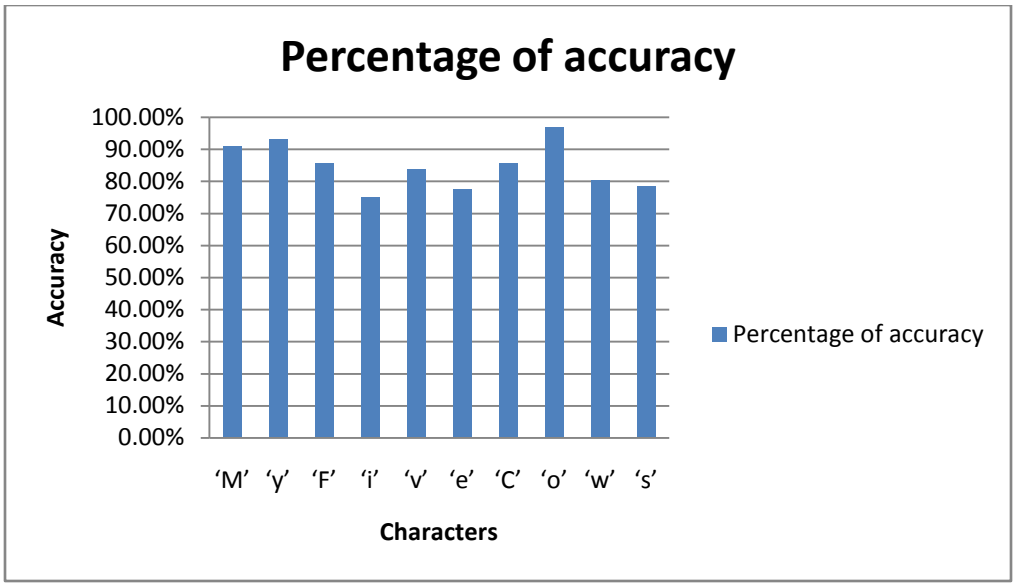

Figure 9.8: Percentage of Accuracy of Paragraph Set-2 by HSIM-net

Multiple layered HSIM-net is a better approach than using a single layered HSIM-net because addition of an extra layer improves the performance of the ANN.

9.3 Character Recognition using Pixel Density Gradient (PDG) Method[♣]

Pixel Density Gradient (PDG) Method is an approach to design a self organizing map in order to identify handwritten characters by observing the gradient of the pixel densities at different segments of the handwritten characters. A self organized ANN is an unsupervised type of ANN where the net tends to converge towards the desired results using fixed weights. Different segments of the characters are observed carefully with the help of suitable computer programs and rigorous experiments. It has been found that the pixel densities at various segments of the character image matrix of different alphabets vary considerably. Gradient of pixel densities, in these segments, are used to generate unique codes for different alphabets, which are found standard for different variations of same alphabet.

[♣] Based on author's Publication no. 1 [Appendix B]

In order to identify handwritten characters, an attempt has been made by dividing the two dimensional image matrix of the character into different segments, where there is a possibility of finding variations in the pixel densities. A combination of different segments of the characters is used to find out unique codes. This helps to train the ANN to find out standard weights. PDG-net is an approach to extract out common features from characters by using the pixel gradient features. Different feature extraction techniques are already applied to recognize characters in different scripts. PDG-net is capable of finding variation (gradient) of the pixel densities, present in the character, in its different segments. So, this Pixel Density Gradient (PDG) net is nothing but a self organized ANN 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 has been used to find out unique binary codes of size 10 bits. These codes are ultimately decoded into 10 lines representing the set of 10 English alphabets. These English alphabet sets may be a combinations of upper and lowercase characters. Test alphabets will specify which output line (out of 10 lines), represent which alphabet.

9.3.1 Architecture of PDG-net

PDG-net (Figure 9.9) consists of three neuron layers and two weight layers. Input neuron layer is represented by vector ‘x’, hidden neuron layer is represented by vectors ‘R’ and ‘C’ where ‘R’ represents row segment outputs and ‘C’ represents column segment outputs. There is another neuron layer, represented by vector ‘D_C’, which generates the codes for different characters. Sometimes some characters produce ambiguous codes.

The outputs produced by ‘D_C’ has been fed to a black box containing a computer program which detects the difference between those characters that produce ambiguous codes and generates unique codes for each character. Output of the black box has been fed to a 10-to-10 line decoder which decodes the code to different characters. There are two weight layers represented by vector ‘v_1’ and ‘v_2’ as shown in Figure 9.9. Neuron layer-1 consists of [6400 + 4 = 6404] 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,….x_{80}}, {x_{81}, x_{82},….x_{160}}, ……, {x_{6321}, $x_{6322,}$ … x_{6400}}] and Column-wise segmentation of the input vector produces columns in the form as shown by the vector [{x_1, x_{81}, …., x_{6321}}, {x_2, x_{82}, …., x_{6322}},…., {x_{80}, x_{160}, ….x_{6400}}].

Four additional neurons are appended to the vector of length 6400 in order to remove the ambiguities produced by different characters. To identify ten characters the worst case ambiguities may be produced by all the ten characters. At least four bits are required to produce ten different combinations.

But, experimental results show hardly two to three such characters produce ambiguous codes. Eighty row and eighty column segments are generated from the first 6400 neurons of the neuron layer-1. Neuron layer-2 consists of 160 neurons out of which first 80 neurons presents outputs generated by row segments and last 80 neurons presents outputs generated by column segments.

Neuron layer-3 consists of ten neurons. First six neurons takes the output from different segments of the neuron layer-2 and last four neurons takes the outputs from neuron layer-1. 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 + 4 =12804] elements, out of which first 6400 weight elements represents weights on row segments and next 6400 weight elements represents weights on column segments and last four links are directly connected to neuron layer-3. 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.

9.3.2 Methodology of PDG-net

In this method, weights are fixed as it is an unsupervised type of ANN and no training is required in advance to adjust the weights. The character samples used in the previous methods are also used in the PDG-net. The binary input character image matrices of dimension [80 x 80 = 6400] are taken for testing purpose.

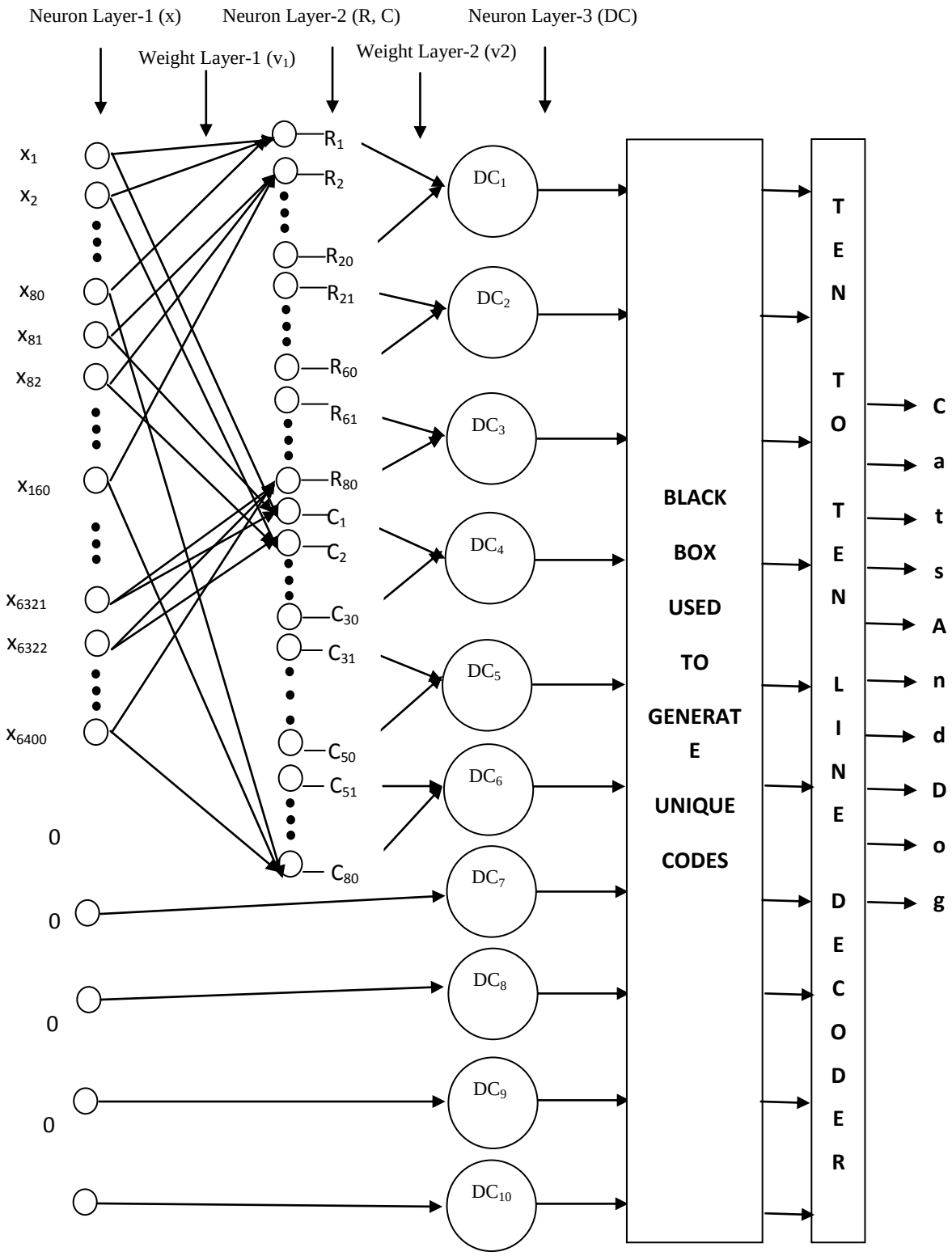

Figure 9.9: Neural Network Representing PDG-net

Four bits are appended to the vector of size 6400 in order to remove ambiguities. No further compression is used in PDG-net as it may increase the ambiguities. Compression may be used for higher dimension matrices.

9.3.2.1 Code Generation using Row Segments

The binary character matrix of dimension '80 x 80' has been segmented row-wise into 80 segments where each row segment consists of 80 neurons as shown in Figure 9.10.

There are three neuron layers. Neuron Layer-1 accepts the input character vector in row major form. The activations of Neuron Layer-2 are calculated as shown in Equation 9.5. Several equations in the article numbers 9.3.2.1 and 9.3.2.2, have been formulated through rigorous experiments and the results obtained thereof.

$$R_j = \{1 \text{ if } R\_tot_j >= \beta \text{ else } 0\} \quad \text{…….. Equation 9.5}$$

'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 experimentally that the 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 Equation 9.6.

$$R\_tot_j = \sum_i v_1 * x_i \quad \text{, where } i = 1 \text{ to } 80 \quad \text{.....Equation 9.6}$$

Between Neuron Layer-1 and Neuron Layer-2 there exists, Weight Layer-1 which is represented by vector 'v_1'. 'v_1'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 in Figure 9.10.

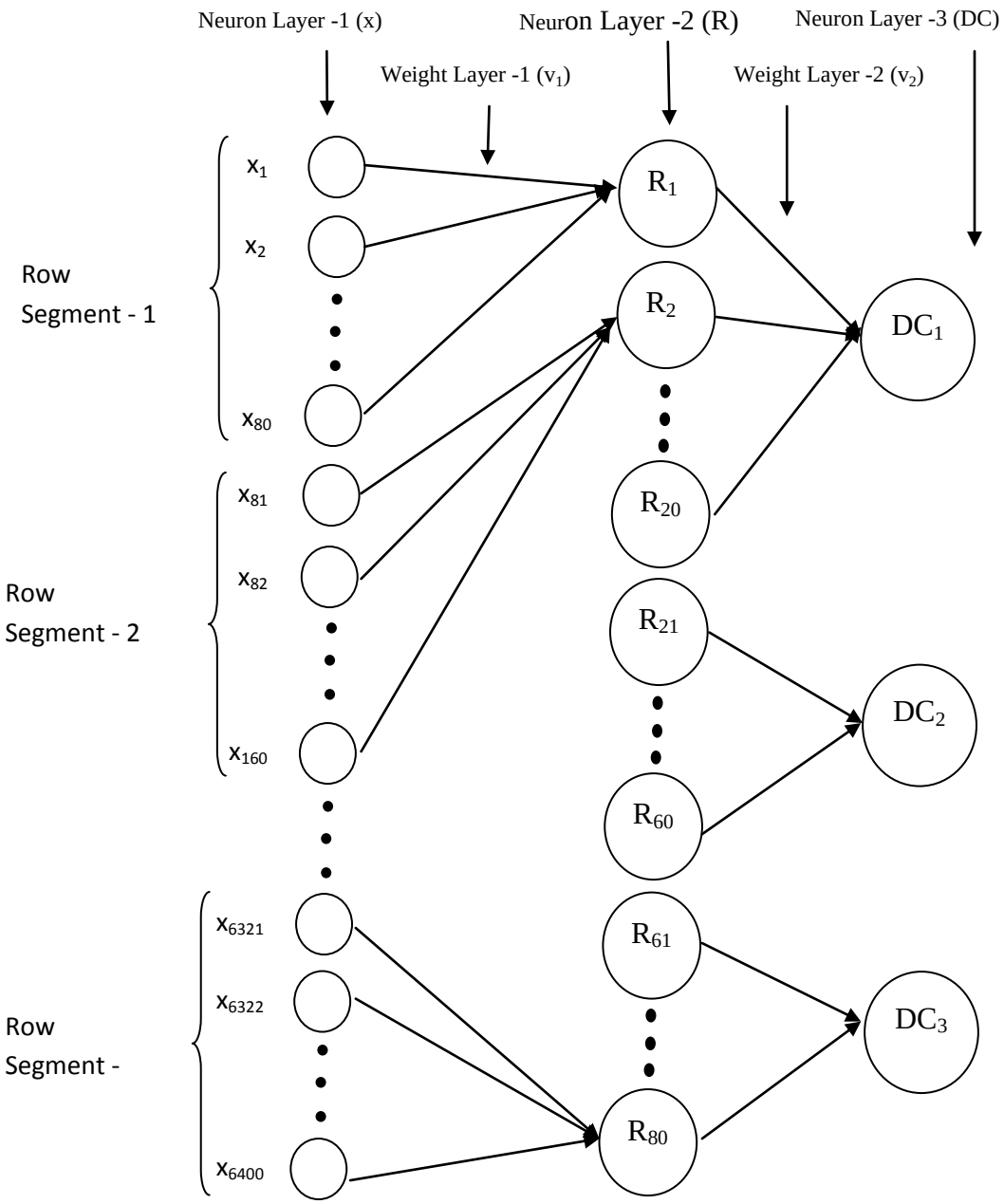

Figure 9.10: Forming Code Bits from Row Segments

$DC_k = \{1 \text{ if } DC\_tot_K >= \alpha_r \text{ else } 0\}$ …….. Equation 9.7

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 black 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 represented by '$segment_k$' as given below:

$DC\_tot_k = \sum_j v_2 * R_j$ ……... Equation 9.8

Where 'j' = 'n' to 'm', if ('k' = 1) {'n' = 1 and 'm' = 20} else if ('k' = 2) {'n' = 21 and 'm' = 60} else if ('k' = 3) {'n' = 61 and 'm' = 80}.

Figure 9.10 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.

9.3.2.2 Code Generation using Column Segments

The binary character image matrix of dimension '80 x 80' has been segmented column-wise into 80 segments as shown in Figure 9.11, where each segment consists of 80 neurons. Neuron Layer-1 accepts the input character in column major form. Figure 9.11 demonstrates the generation of next three bits of the pattern generated. Between Neuron Layer-1 and Neuron Layer-2 there exists 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 Equation 9.9.

$C_j = \{1 \text{ if } C\_tot_j >= \beta_c \text{ else } 0\}$ ……Equation 9.9

'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 experimentally that the 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 Equation 9.10.

$$C\_tot_j = \sum_i v_1 * x_i \quad \text{, where } i = 1 \text{ to } 80 \qquad \text{..... Equation 9.10}$$

Between Neuron Layer-2 and Neuron Layer-3 there exists Weight Layer-2 represented by vector 'v_2' which is fixed to 1. Neuron Layer-3 consists of only 3 neurons, the activations of which may be calculated as shown in Equation 9.11.

$$DC_k = \{1 \text{ if } DC\_tot_K >= \alpha_c \text{ else } 0\} \qquad \text{...Equation 9.11}$$

'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 segments. 'DC_tot_k' is the total number of columns with dense dark pixels present in a particular code segment as shown in Equation 9.12.

$$DC\_tot_k = \sum_j v_2 * C_j \qquad \text{.....Equation 9.12}$$

Where 'j' = 'n' to 'm', if ('k' = 4) {'n' = 1 and 'm' = 20} else if ('k' = 5) {'n' = 21 and 'm' = 60} else if ('k' = 6) {'n' = 61 and 'm' = 80}

The bits obtained from the column segments of the characters forms the next three bits (i.e. 4^{th}, 5^{th} and 6^{th}) of the ten bit binary code segment. The PDG-net produces binary codes for all the characters under consideration. The binary codes produced for each character has been presented to a black box containing a computer program which

removes the ambiguities in the codes generated by two or more different characters. The unique binary codes produced by the black box for each character is decoded by a 10-to-10 line decoder. The output of the decoder indicates the identified alphabet as discussed earlier.

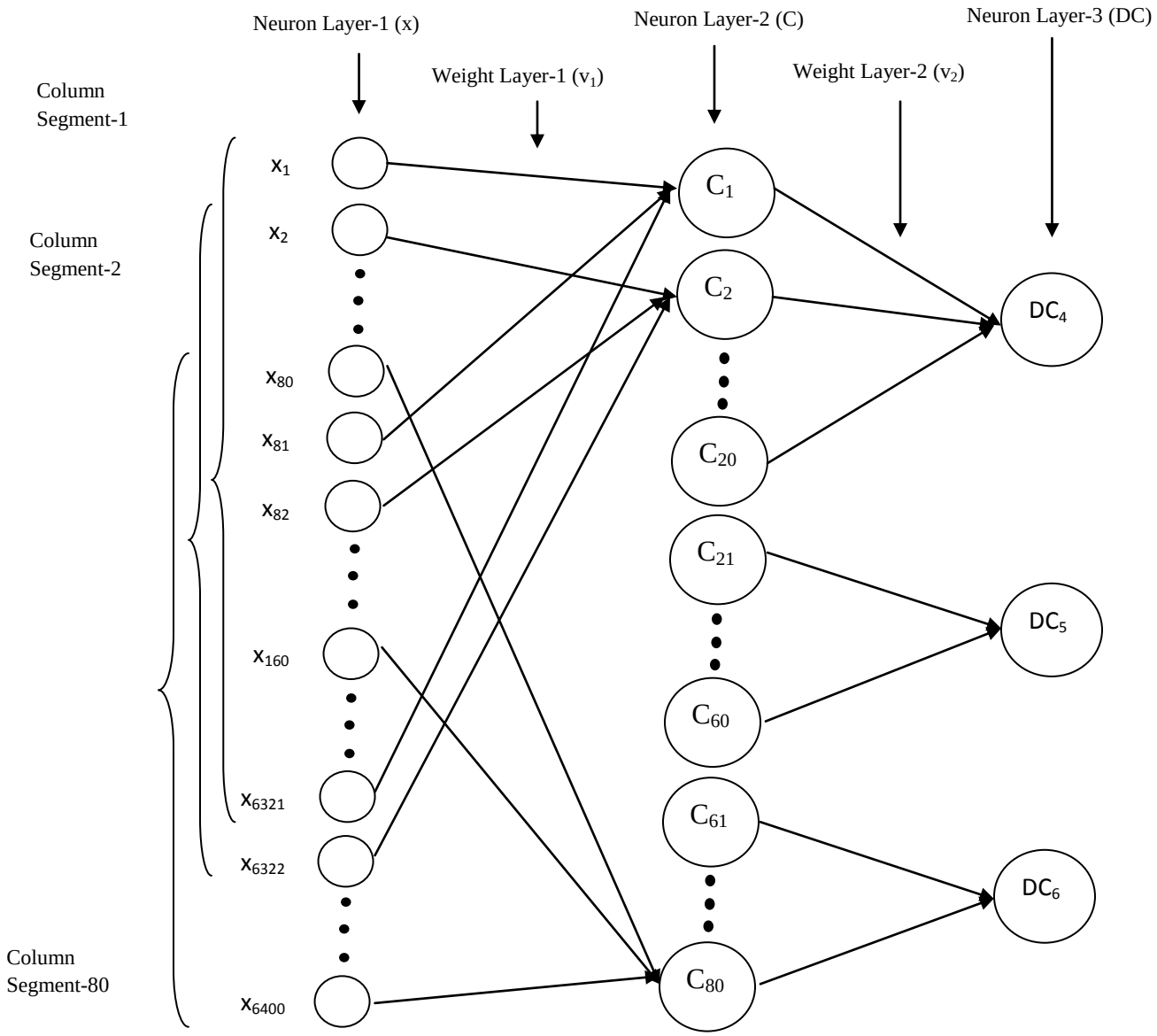

Figure 9.11: Forming Code Bits from Column Segments

9.3.2.3 Ambiguity Removal

Some characters like ‘A’ and ‘g’ generate identical code pattern like ‘111111’. Four bits are appended at the end of six bit code in order to remove such type of ambiguity. Ambiguity is removed by applying the following Algorithm 9.4.

Algorithm 9.4 (Ambiguity Removal): Removing ambiguities

STEP 1. Consider the first set of characters showing the identical patterns.

STEP 2. Calculate the pixel densities in each segment of the characters producing identical code patterns and store the values in ‘s_{ij}’. [Where, ‘s_{ij}’ is the pixel density in j^{th} segment of the i^{th} character.]

STEP 3. If a character shows a greater value of ‘s_{ij}’ for any segment as compared to the other character\characters in the corresponding segment then change the last bit of the four bit additional pattern to ‘1’.

STEP 4. Repeat Steps 1 to 3 for all the characters producing identical code patterns:

[For example in case of ‘A’ and ‘g’ the code pattern generated is ‘111111’. Addition of the last four bits makes it ‘1111110000’. Applying Steps 1 and 2 it is found that the pixel density in case of first row segment in case of ‘g’ is far greater than that of ‘A’. So, ‘g’ is represented as ‘1111110001’ while ‘A’ remains ‘1111110000’. For more than two characters ‘0010’ is appended to the second character showing some difference. Same process is repeated for all the characters generating identical patterns.]

STEP 5. STOP.

9.3.2.4 Application of 10-To-10 Line Decoder to Decode the Unique Codes into Characters

The unique codes, produced by all the characters after application of Algorithm 9.4, are fed to the 10-to-10 line decoder. The decoder decodes the input codes to ten characters.

Algorithm 9.5 (PDGM): Working of PDGM

STEP 1. Consider the binary character image matrixes of dimensions '80 x 80'.

STEP 2. Segment the binary character matrix row-wise into 80 segments where each row segment consists of 80 neurons.

STEP 3. Calculate the activations of neurons representing the rows in Neuron Layer-2 using Equation 9.5 and 9.6.

STEP 4. Calculate the activations of neurons representing the code bits obtained from the row segments in Neuron Layer-3 using Equation 9.7 and 9.8.

STEP 5. Segment the binary character matrix column-wise into 80 segments where each column segment consists of 80 neurons.

STEP 6. Calculate the activations of neurons representing the columns in Neuron Layer-2 using Equation 9.9 and 9.10.

STEP 7. Calculate the activations of neurons representing the code bits obtained from the column segments in Neuron Layer-3 using Equation 9.11 and 9.12.

STEP 8. Append four 0 bits to each six bit binary codes generated by each character presented to PDG-net.

STEP 9. Apply Equation 9.3 to remove the ambiguities from the identical codes

produced by different characters.

STEP 10. STOP.

9.3.3 Performance Analysis of PDG-net

MATLAB software has been used to test the accuracy of the PDG-net. The characters extracted out of the paragraph sets (Section 6.2.6) are taken for testing the performance of the PDG-net. The response of the experiment shows good results.

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 = 10
Total Number of Neurons = 6570
One 10-to-10 line decoder
Fixed weights between Neuron Layer-1 & Neuron Layer-2 = 1
Fixed weights between Neuron Layer-2 & Neuron Layer-3 = 1
The accuracy of the system is shown in Table 9.5 and Table 9.6
Accuracy generated by Paragraph Set 1 = 94.70%.
Accuracy generated by Paragraph Set 2 = 94.63%.
Average Accuracy of PDG-net= 94.67%

Figure 9.12 displays the column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-1 by PDG-net. Figure 9.13 displays the column-chart showing the percentage of accuracy of identifying different characters of Paragraph Set-2 by PDG-net.

Table 9.5: Identification of Characters of Paragraph Set-1 using PDG-net

Alphabets	Number of Segmented Characters Presented	Correctly Recognized	Percentage of Accuracy
'C'	89	83	93.26%
'a'	95	92	96.84%
't'	80	76	95.00%
's'	92	86	93.48%
'A'	100	100	100%
'n'	80	78	97.50%
'd'	90	84	93.33%
'D'	92	85	92.39%
'o'	98	95	96.93%
'g'	70	60	85.71%
Total	**886**	**839**	**94.70%**

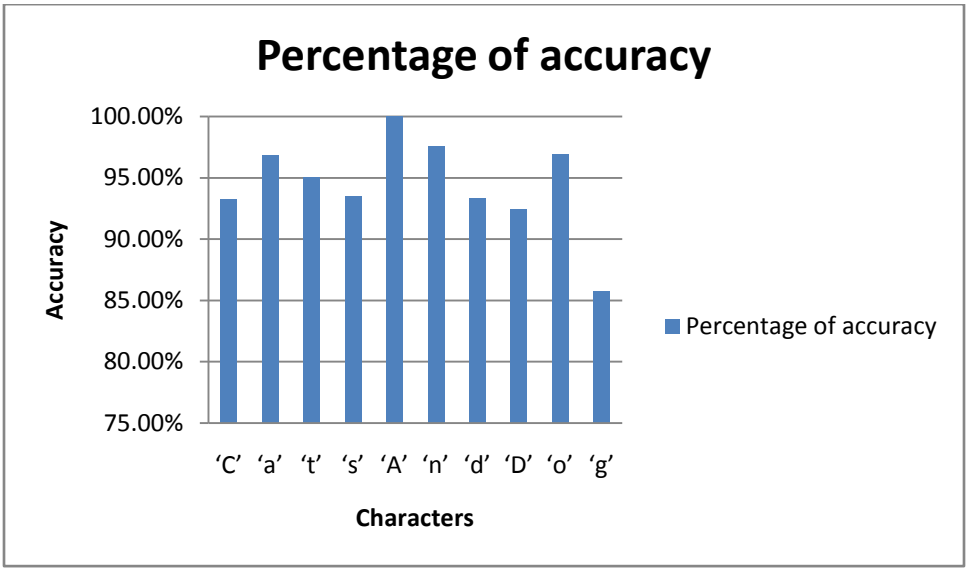

Figure 9.12: Percentage of Accuracy of Paragraph Set-1 by PDG-net

Table 9.6: Identification of Characters of Paragraph Set-2 using PDG-net

Alphabets	Number of Segmented Characters Presented	Correctly Recognized	Percentage of Accuracy
'M'	100	100	100%
'y'	71	65	91.55%
'F'	91	88	96.70%
'i'	60	52	86.67%
'v'	73	70	95.89%
'e'	89	82	92.13%
'C'	90	86	95.56%
'o'	92	87	94.57%
'w'	71	66	92.96%
's'	83	80	96.39%
Total	**820**	**776**	**94.63%**

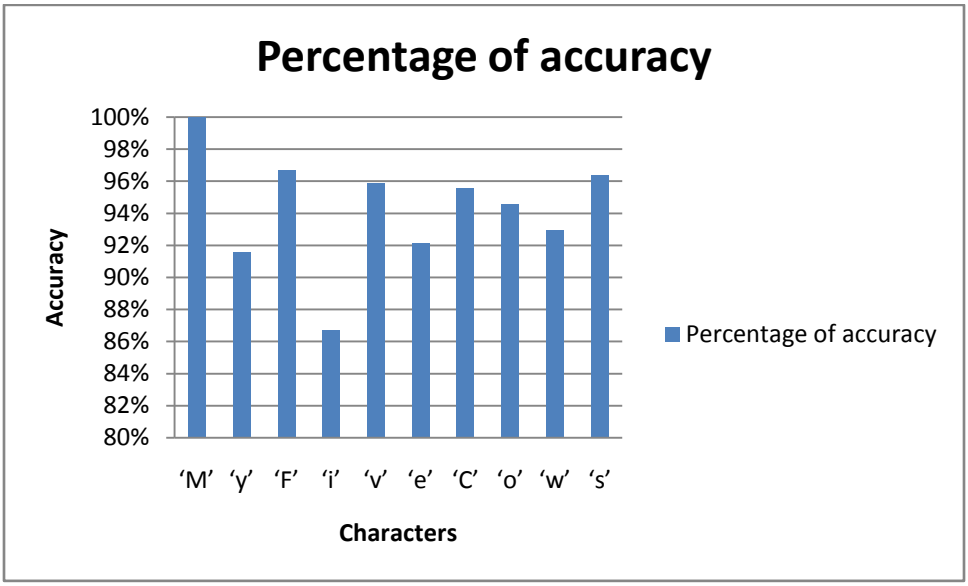

Figure 9.13: Percentage of Accuracy of Paragraph Set-2 by PDG-net

9.4 Conclusion

The segmentation of character image matrix into arrows, in the ASIM-net, has been performed in order to extract out features. This may sometimes lead to skipping of pixel features, present the outside the arrow segments. This may result in little degradation in performance of the net. An advantageous remedy also exists for this type of segmentation which is training by modified inputs. Training the ANN with modified input vectors yields very good results in case of ASIM-net.

In case of HSIM-net, skipping of pixel features, outside the hoof is less as compared to ASIM-net because hoof segmentation covers better scope of the character image matrix. There is a meager chance of performance degradation because of feature skipping. Here also modified input vectors are used and very good results are obtained.

The design of PDG-net is done in such a manner, so that the pixel gradients can be obtained from all the portions of the character image matrix. There is hardly any chance of feature skipping. This makes the mapping of the inputs into different clusters very simple as compared to the complex training of the ASIM-net and HSIM-net. Also, the results produced by the PDG-net shows much better performance as compared to other multiple layer ANNs.