Chapter 7

Development of Prototype Model[♣]

This chapter is a discussion on the development of a prototype model which is used to test the performances of some traditional ANNs like Hebb and Perceptron [4, 63]. To develop and propose a suitable method that can recognize the handwritten characters efficiently it is very essential to find out the pros and cons of already developed classical ANNs. All the characters extracted out with different dimensions, after segmentation, are converted into fixed dimension matrix of 80 x 80 using the preprocessing techniques already discussed Chapter 5. The matrix is then converted into a bivalent vector of size 6400. Here, the size of the input vector has been limited to 6400 for training and testing purpose for simplicity of the ANN model. A prototype model has been designed to test the preprocessed character image.

7.1 Architecture

A single layer prototype ANN model consists of two neuron layers and one weight layer that has been sandwiched between the two neuron layers. The neuron layer-1, which is the input layer, consists of 6400 neurons while the neuron layer-2 is the output layer, consisting of only 10 neurons. This prototype ANN has been developed to train and test only 10 characters at a time, so there are 10 neurons at the output layer. Each neuron in the output layer is responsible for the identification of a particular character. The number of neurons in the input layer is equal to the size of the input vector. All the neurons in the input layer are completely interconnected to the 10 neurons in the output layer. Therefore, the weight layer consists of 6400 x 10 = 64000 interconnections. The weight vector consists of 64000 elements. Figure 7.1 displays a prototype model which can be used to test the performances of Hebb and Percepton ANNs. The input layer is denoted by vector ‘x’, weight layer is denoted by vector ‘w’

[♣] Based on author’s Publication nos. 10 and 12 [Appendix B]

and output layer is denoted by 'y'. The i^{th} element in the input layer is denoted by 'x_i', j^{th} layer in the output layer is denoted by 'y_j' and 'w_{ij}' denotes the weight of the interconnection connecting i^{th} neuron of the input layer with the j^{th} neuron of the output layer.

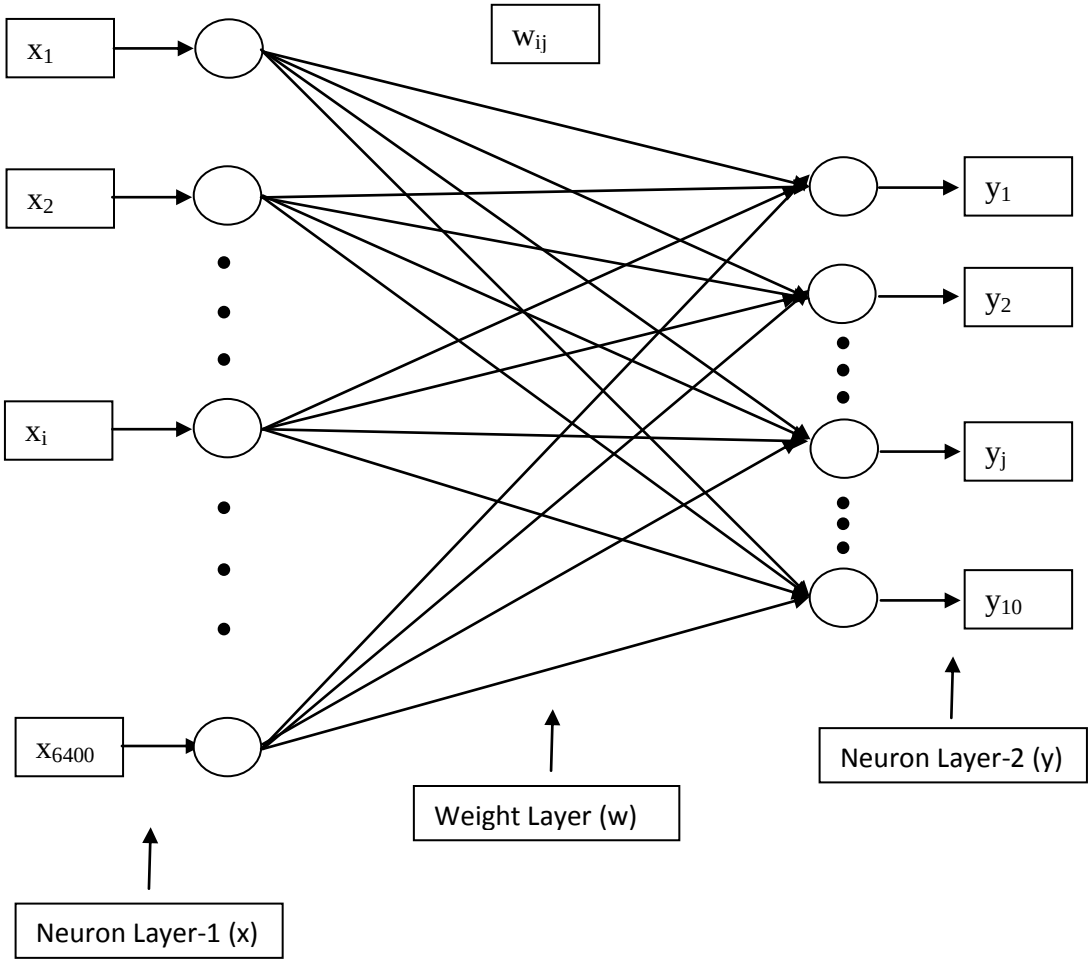

Figure 7.1: A Prototype Model to Test the Performances of Classical ANNs

This model becomes more complex with increase of number of training characters. This increases the number of neurons in the output layer. The weight vector increases geometrically with increase in neurons at the output layer. To solve this problem the number of neurons has been reduced in the output layer. Instead of using 10 neurons in the output layer comparatively lesser number of neurons are used that reduced

the space and time complexity. For example, 10 distinct possible bivalent combinations are used instead of using 10 neurons. Each bivalent combination is responsible for identifying a particular character. Four neurons can be used to produce ten distinct bivalent combinations.

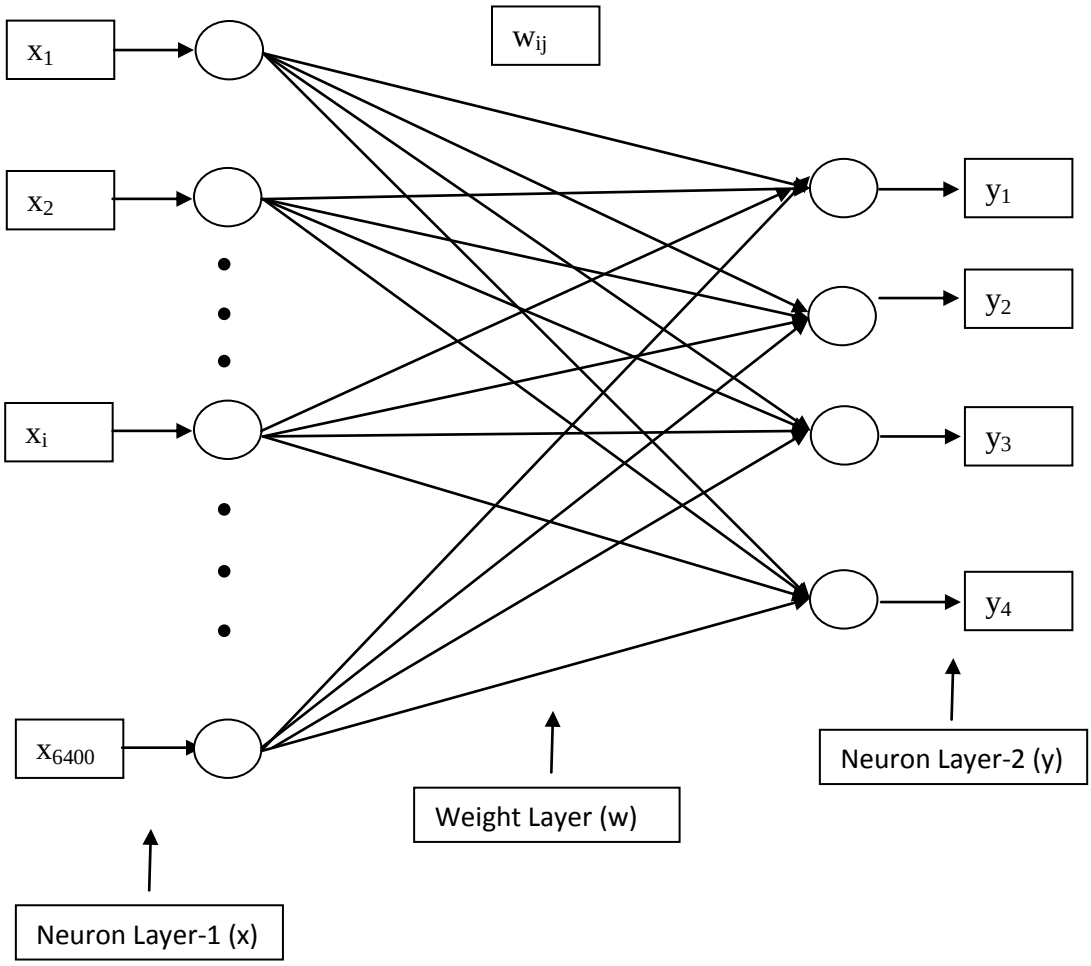

Figure 7.2: An Optimized Prototype Model with less Number of Neurons in the Output Layer

For example, the vector [1, -1, -1, -1, -1, -1, -1, -1, -1, -1] which represents the first character can be replaced by a vector [-1, -1, -1, 1]. If -1s are replaced by 0s the resultant vector becomes '0001' which is decimal 1. Similarly, the vector [-1, 1, -1, -1, -1, -1, -1, -1, -1, -1] which represents the second character can be replaced by a vector [-1, -

1, 1, -1] and so on. Finally, the vector [-1, -1, -1, -1, -1, -1, -1, -1, -1, 1] which represents the tenth character can be replaced by a vector [1, -1, 1, -1]. This saved a major processing time of the ANN as weight modifications are reduced to a greater extent which is very useful in larger and complex nets. Figure 7.2 displays a prototype model which is an optimized form of the prototype model represented in Figure 7.1. In this prototype number of neurons in the output layer has been reduced to four.

7.2 Methodology

Segmented characters from two sample paragraph-sets (Set-1 and Set-2 of Section 6.2.6) are considered for training and testing purpose. Each set contains 1000 individual characters since each set contains ten paragraphs, each paragraph contains ten sentences and each sentence contains ten characters. Only ten distinct different characters, from each paragraph set, say Set-1, are needed to train the ANNs in order to test all the characters present in Set-1. Similarly, ten distinct different characters from Set-2 are chosen for training to test the all characters of Set-2. Training of the nets has been limited to ten different characters, may be lower or upper case, for each set as nets and their interfaces (Chapter10) are designed for doing so. Training characters are chosen from paragraph-sets at random.

The characters, from Set-1 and Set-2 that are used here for training purpose are saved and used as training sets for all other developed ANNs done in this work.

During training after few epochs standard weights are generated by each of the classical ANNs as discussed in Sections 7.3 and 7.4. The characters which are extracted out by the BPBM method discussed in Chapter 6 **a**re preprocessed and presented to the prototype model for identification. Figure 7.3 displays the conceptual diagram of the overall methodology of the prototype model designed to test the performances of the ANNs. The weight matrix is initialized to zero. Target vectors for different characters are set as shown in Table 7.1.

Table 7.1: Target Vector Set for Different Characters

S. No.	Character	Equivalent Decimal Code	Equivalent Binary Number	Bivalent Target Vector
1.	'C'	1	0001	[-1, -1, -1, 1]
2.	'a'	2	0010	[-1, -1, 1, -1]
3.	't'	3	0011	[-1, -1, 1, 1]
4.	's'	4	0100	[-1, 1, -1, -1]
5.	'A'	5	0101	[-1, 1, -1, 1]
6.	'n'	6	0110	[-1, 1, 1, -1]
7.	'd'	7	0111	[-1, 1, 1, 1]
8.	'D'	8	1000	[1, -1, -1, -1]
9.	'o'	9	1001	[1, -1, -1, 1]
10.	'g'	10	1010	[1, -1, 1, -1]

As the prototype model has been designed to train and test only 10 characters at a time, training samples are presented to the ANN one by one and the activations are calculated for each training sample.

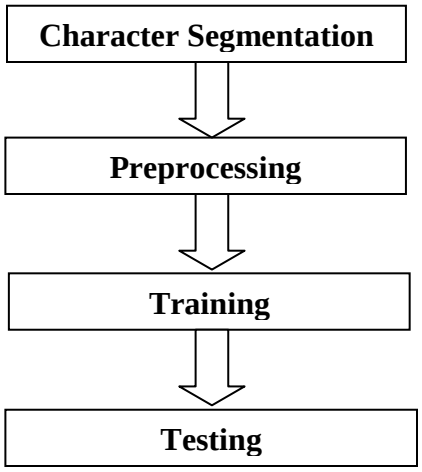

Figure 7.3: Conceptual Diagram of the Overall Methodology

7.3 Testing the Performance of Hebb ANN

Weights are initially set to zero. The preprocessed characters from different sets are presented to the ANN one by one. For each character net output in the output layer is calculated using Equation 7.1. The net output of the j^{th} neuron in the output layer is represented by 'y_out_j'. The value of the i^{th} neuron in the input vector is represented by 'x_i' and 'w_{ij}' is the weight at the link connecting i^{th} neuron in the input layer with the j^{th} neuron in the output layer.

The activation function is 'f' and 'θ' is the threshold value, which decides that the neuron will fire or not as given in Equation 7.2. If y_net_j exceeds the threshold value while training, the neuron fires and the value of y_j is set to 1 else the value is set to -1. The threshold is set to a random value.

$$y\_out_j = \sum_i x_i * w_{ij} \quad \text{.....Equation 7.1}$$

$$y_j = f(y\_out_j) = \{ 1 \text{ if } y\_net_j >= \theta \text{ else } -1\} \quad \text{.....Equation 7.2}$$

After training the ANN for one epoch the vector 'y' is compared with the target vector. If the vector is identical to the target vector, weights are not modified else weights are modified till the output vector 'y' matches the target vector. The weights can be modified by using Equation 7.3 as follows:

$$w_{ij}(new) = w_{ij}(old) + x_i y_j \quad \text{.....Equation 7.3}$$

Where, '$w_{ij}(new)$' is the modified weight, '$w_{ij}(old)$' is the existing weight, 'x_i' is the input at the 'i^{th}' neuron and 'y_j' is the activation produced by the ANN for the j^{th} neuron in the output layer.

7.3.1 Result Analysis of Hebb ANN

Number of neurons in the input unit=6400

Number of Neurons in the output unit = 4

Dimension of each weight matrix = 6400 x 4

Training Algorithm used = Hebb

Number of Epochs = 7, Threshold (θ) = 0.2,

Accuracy generated by Paragraph Set 1 = 39.05%.

Accuracy generated by Paragraph Set 2 = 45.61%.

Average Accuracy of Hebb-net= 42.33%

The accuracy of the system is shown in Table 7.2, Table 7.3 and corresponding column graphs are shown in Figure 7.4 and Figure 7.5.

Table 7.2: Identification of Characters of Paragraph Set-1 using Hebb-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'C'	89	41	46.07%
'a'	95	35	36.84%
't'	80	39	48.75%
's'	92	31	33.69%
'A'	100	36	36.00%
'n'	80	30	37.50%
'd'	90	32	35.56%
'D'	92	37	40.22%
'o'	98	33	33.67%
'g'	70	32	45.71%
Total	**886**	**346**	**39.05%**

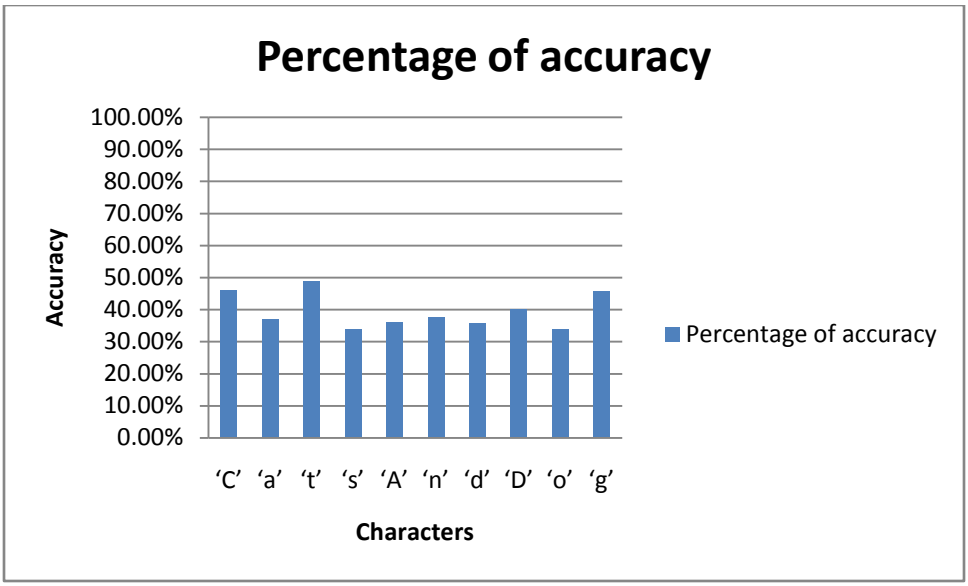

Figure 7.4: Percentage of Accuracy of Paragraph Set-1 by Hebb-net

Table 7.3: Identification of Characters of Paragraph Set-2 using Hebb-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'M'	100	35	35.00%
'y'	71	39	54.93%
'F'	91	41	45.05%
'i'	60	39	65.00%
'v'	73	41	56.16%
'e'	89	39	43.82%
'C'	90	33	36.67%
'o'	92	30	32.61%
'w'	71	41	57.75%
's'	83	36	43.37%
Total	**820**	**374**	**45.61%**

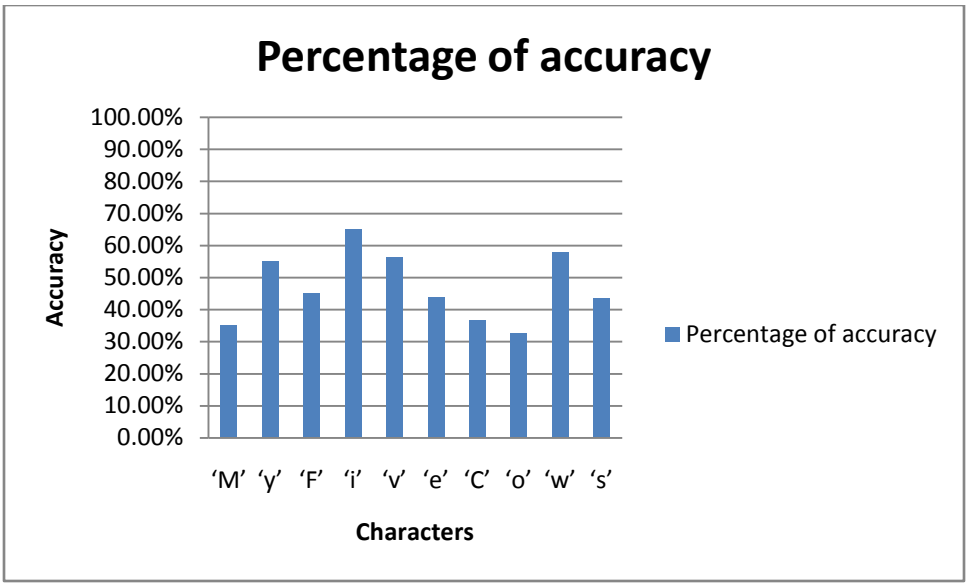

Figure 7.5: Percentage of Accuracy of Paragraph Set-2 by Hebb-net

7.4 Testing the Performance of Perceptron ANN

In the case of Perceptron ANN, the net output is calculated using Equation 7.1 and activation is calculated using Equation 7.4 as follows:

$$y_j = f(y\_out_j) = \begin{cases} 1 \text{ if } y\_out_j > \theta \\ 0 \text{ if } -\theta <= y\_out_j <= \theta \\ -1 \text{ if } y\_out_j < -\theta \end{cases} \quad \text{……..Equation 7.4}$$

The weights are modified using Equation 7.5, where, 'α' is the learning rate and 't_j' is the target value set for j^{th} neuron in the output layer.

$$w_{ij}(new) = w_{ij}(old) + \alpha t_j x_i \quad \text{…….Equation 7.5}$$

7.4.1 Result Analysis of Perceptron ANN

Number of neurons in the input unit=6400

Number of Neurons in the output unit = 4

Dimension of each weight matrix = 6400 x 4

Training Algorithm used = Perceptron

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

Accuracy generated by Paragraph Set 1 = 49.66%.

Accuracy generated by Paragraph Set 2 = 49.15%.

Average Accuracy of Perceptron-net= 49.41%

The accuracy of the system is shown in Table 7.4 and Table 7.5 and corresponding column graphs are shown in Figure 7.6 and Figure 7.7.

Table 7.4: Identification of Characters of Paragraph Set-1 using Perceptron-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'C'	89	40	44.94%
'a'	95	44	46.31%
't'	80	46	57.50%
's'	92	49	53.26%
'A'	100	50	50.00%
'n'	80	40	50.00%
'd'	90	42	46.67%
'D'	92	41	44.56%
'o'	98	46	46.94%
'g'	70	42	60.00%
Total	**886**	**440**	**49.66%**

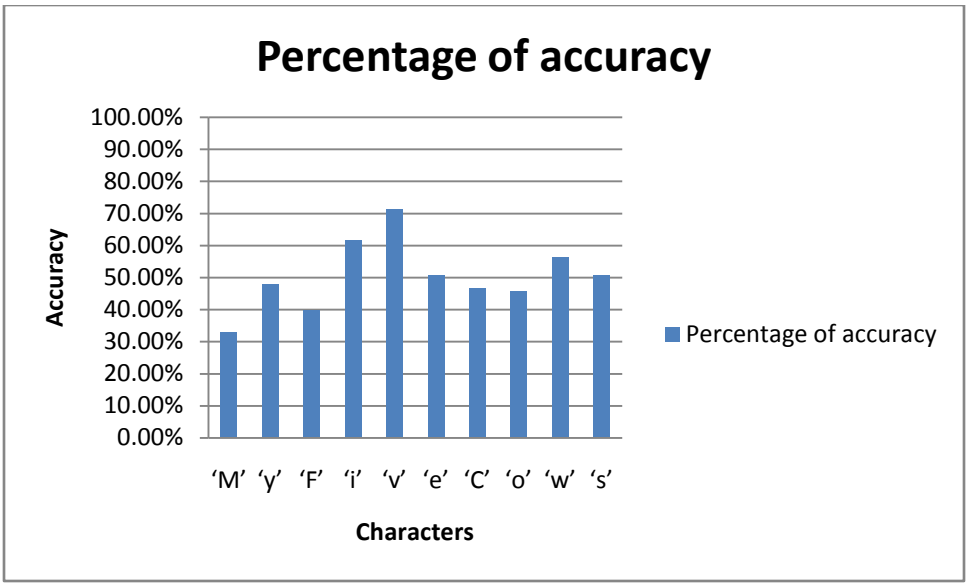

Figure 7.6: Percentage of Accuracy of Paragraph Set-1 by Perceptron-net

Table 7.5: Identification of Characters of Paragraph Set-2 using Perceptron-net

Alphabets	**Number of Segmented characters**	**Correctly recognized**	**Percentage of accuracy**
'M'	100	33	33.00%
'y'	71	34	47.89%
'F'	91	36	39.56%
'i'	60	37	61.67%
'v'	73	52	71.23%
'e'	89	45	50.56%
'C'	90	42	46.67%
'o'	92	42	45.65%
'w'	71	40	56.34%
's'	83	42	50.60%
Total	**820**	**403**	**49.15%**

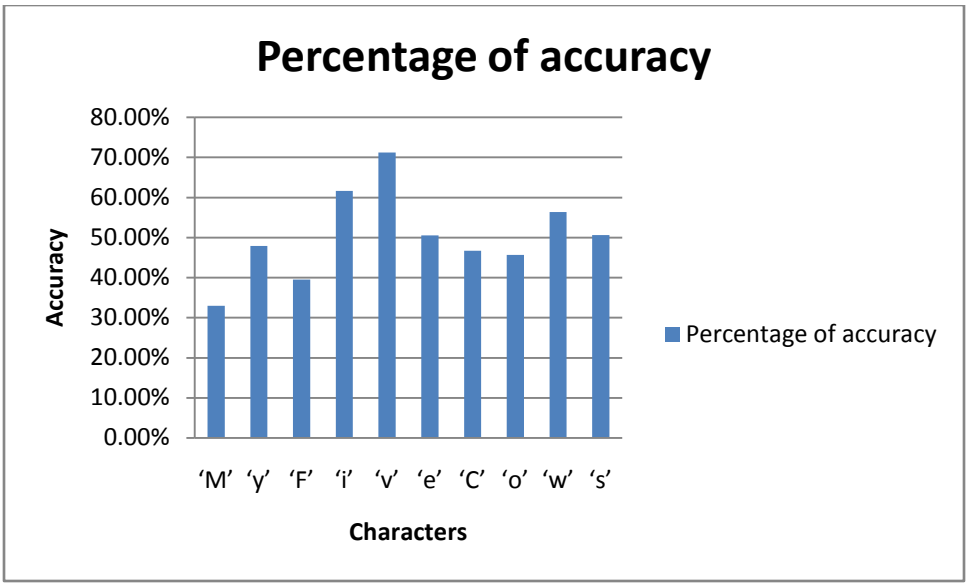

Figure 7.7: Percentage of Accuracy of Paragraph Set-2 by Perceptron-net

7.5 Conclusion

A prototype model has been designed to test the performances of two single layer ANNs, Hebb and Perceptron. The same data samples that are generated to test the performance of all the models developed in this work are applied. It has been found that both Hebb and Perceptron ANN performs very well for those characters which are not much deviated from the sample characters used to train the ANN. But performance degrades with increase in deviations. So, these methods are not found suitable where generalizations are required to identify the deviations. To overcome the problem different single / multiple layer ANNs, based on feature extraction, are necessary.