Volume 4, Issue 5, May 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Huffman Encoding using Artificial Neural Network

Surbhi Garg* *Department of ECE, LCET PTU, India*

Balwinder Singh Dhaliwal *Department of ECE, GNDEC PTU, India*

Priyadarshni *Department of ECE, LCET PTU, India*

Abstract- This paper presents a model of artificial neural network (ANN) for a famous source coding algorithm in the field of Information Theory- the Huffman scheme. This model helps to easily realize the algorithm without going into a cumbersome and prone to error theoretical procedure of encoding the symbols. Neural network architecture used in the work is multilayer perceptron which is trained using backpropagation algorithm. ANN results have been compared with theoretical results. Matching of ANN results with theoretical results verifies the accuracy of model.

Keywords- Source coding, Huffman encoding, Artificial neural network, Multilayer perceptron, Backpropagation algorithm

I. INTRODUCTION

Source coding may be viewed as a branch of information theory which attempts to remove redundancy from the information source and therefore it achieves data compression. Source coding has important application in the areas of data transmission and data storage. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium [1]. In general, data is compressed without losing information (lossless source coding) or with loss of information (lossy source coding). Huffman coding algorithm [2] is used for lossless source coding. The following difficulties are faced when encoding problems using Huffman algorithm are solved theoretically; - (i) Theoretical encoding process is tedious; (ii) It is difficult to track any error made in between and, hence, this can lead to wrong answers; and (iii) The procedure takes long time to complete. An alternative solution for the stated difficulties has been found to be the use of artificial neural network (ANN) for the encoding process which will accept inputs from the user and then perform the required task accordingly.

An ANN is considered as a highly simplified model of the structure of the biological neural network. It is composed of a set of processing elements, also known as neurons or nodes, which are interconnected. It can be described as a directed graph in which each node i performs a activation or transfer function f_i of the form, $y_i = f_i (\sum w_{ij} x_j - \theta_i)$ where j = 1 to n, y_i is the output of the node i, x_j is the jth input to the node, w_{ij} is the connection weight between nodes i and j, and θ_i is the threshold or bias of the node. Usually, f_i is nonlinear, such as a heaviside, sigmoid, or gaussian function [3].

Knowledge is acquired by the ANN through a learning process which is typically accomplished using examples. This is also called training in ANNs because the learning is achieved by adjusting the connection weights in ANNs iteratively so that trained or learned ANNs can perform certain tasks. The essence of a learning algorithm is the learning rule, i.e. a weight-updating rule which determines how connection weights are changed. Examples of popular learning rules include the Hebbian rule, the perceptron rule and the delta learning rule [4].

ANNs can be divided into feedforward and recurrent classes according to their connectivity. In feedforward networks, the signal flow is from input to output units, strictly in a feedforward direction. The data processing can extend over multiple (layers of) units, but no feedback connections are present. Feedback/Recurrent networks can have signals travelling in both directions by introducing loops in the network. A special class of layered feedforward networks known as multilayer perceptrons (MLP) is very popular and is used more than other neural network type for a wide variety of tasks, such as pattern and speech recognition, audio data mining [5], integrated circuit chip layout, process control, chipfailure analysis, machine vision, voice synthesis and prediction of stroke diseases [6].

The architecture of an MLP is variable but in general will consist of several layers of neurons. The input layer plays no computational role but merely serves to pass the input vector to the network. The terms input and output vectors refer to the inputs and outputs of the MLP and can be represented as single vectors, as shown in Fig. 1. An MLP may have one or more hidden layers and finally an output layer [7]. The training of an MLP is usually accomplished by using backpropagation (BP) algorithm [8] as it is simple to implement.

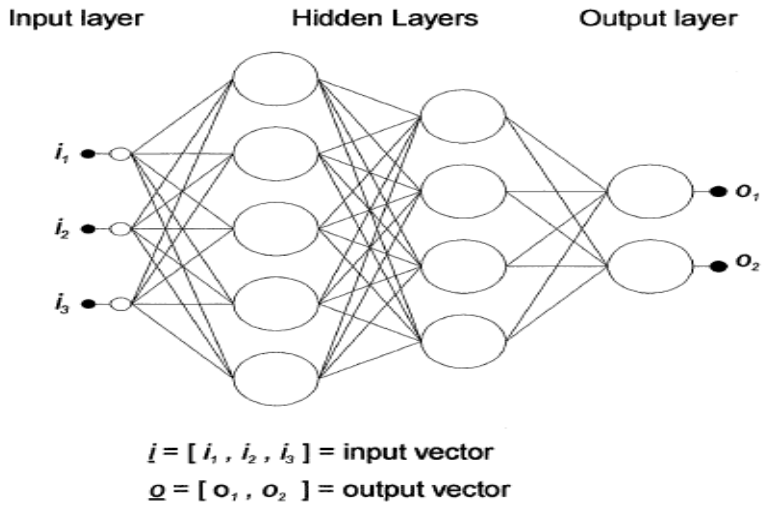

Fig. 1 A multilayer perceptron with two hidden layers [7]

II. PROBLEM FORMULATION

To reduce difficulty and save time in theoretical encoding process using Huffman algorithm, there has been a number of works [9]-[11]. All such implementations of Huffman algorithm are based on sequential approach of programming that means execution takes place step by step. These sequential approaches are not fault tolerant i.e. a single error may lead to unsatisfactory performance. To make the implementation more reliable, ANNs can be used.

ANNs have been finding applications in almost every field because of their advantages such as fault tolerance, generalization and massive parallel processing. Application of ANN in Huffman encoding has not yet been explored. In this work, ANN model for Huffman encoding scheme is developed and accuracy of model is checked by comparing ANN results with theoretical results. The proposed work has been carried out using MATLAB software.

III. PROPOSED ANN MODEL

Various steps are followed in the development of model. These are discussed one by one as follows:

A. Preparation of Data Sheet: The model is constructed based on a training set i.e. a set of examples of network inputs with known outputs called target or desired outputs. Here network inputs correspond to the probability of occurrence of symbols or characters in a particular text which is assumed to be known and target outputs correspond to the set of codewords computed by Huffman algorithm. In this work, Huffman encoding for four symbols is considered and corresponding probability values are taken as inputs.

The four symbols are taken as A, B, C and D. Their corresponding probability of occurrence P_A, P_B, P_C and P_D are assumed to be known such that $P_A \geq P_B \geq P_C \geq P_D$ and $P_A + P_B + P_C + P_D = 1$. Following table is then prepared by theoretically calculating the codewords through Huffman algorithm where C_A, C_B, C_C and C_D are codewords for symbols A, B, C and D respectively.

After careful analysis of the Table I, it is found that there are two cases.

1) Case 1: if magnitude $(P_A - (P_B + P_C + P_D)) \leq$ magnitude $((P_A + P_B) - (P_C + P_D))$

Then C_A is 1 bit in length as P_A has the highest probability, C_B is 2 bit in length as P_B is less than P_A. Each C_C and C_D is 3 bit in length as P_C and P_D are less than P_B and nearly equal.

2) Case 2: if magnitude $(P_A - (P_B + P_C + P_D)) >$ magnitude $((P_A + P_B) - (P_C + P_D))$

Then all C_A, C_B, C_C and C_D are 2 bit in length as P_A, P_B, P_C and P_D are nearly equal.

According to these two cases, two data sheets (Table II and Table III) are prepared.

TABLE I: CODEWORDS COMPUTED THEORETICALLY THROUGH HUFFMAN ALGORITHM

S. No.	Probability				Code			
	P_A	P_B	P_C	P_D	C_A	C_B	C_C	C_D
1	0.36	0.33	0.18	0.13	1	00	010	011
2	0.47	0.23	0.19	0.11	1	01	000	001
3	0.54	0.19	0.15	0.12	0	11	100	101
4	0.40	0.30	0.20	0.10	1	00	010	011
5	0.25	0.25	0.25	0.25	00	01	10	11
6	0.30	0.30	0.20	0.20	00	01	10	11
7	0.94	0.03	0.02	0.01	0	10	110	111
8	0.69	0.21	0.05	0.05	0	10	110	111
9	0.77	0.14	0.06	0.03	0	10	110	111
10	0.67	0.11	0.11	0.11	0	11	100	101
11	0.30	0.30	0.25	0.15	00	01	10	11

12	0.33	0.33	0.21	0.13	00	01	10	11
13	0.28	0.28	0.28	0.16	00	01	10	11
14	0.84	0.10	0.03	0.03	0	10	110	111
15	0.60	0.20	0.10	0.10	0	10	110	111
16	0.31	0.23	0.23	0.23	00	01	10	11
17	0.29	0.27	0.23	0.21	00	01	10	11
18	0.40	0.20	0.20	0.20	1	01	000	001
19	0.50	0.20	0.20	0.10	0	11	100	101
20	0.31	0.30	0.28	0.11	00	01	10	11
21	0.63	0.18	0.18	0.01	0	11	100	101
22	0.58	0.31	0.10	0.01	0	10	110	111
23	0.44	0.24	0.20	0.12	1	01	000	001
24	0.27	0.26	0.25	0.22	00	01	10	11
25	0.26	0.26	0.25	0.23	00	01	10	11
26	0.81	0.09	0.08	0.02	0	11	100	101
27	0.55	0.15	0.15	0.15	0	11	100	101
28	0.32	0.28	0.22	0.18	00	01	10	11
29	0.37	0.29	0.20	0.14	1	01	000	001
30	0.41	0.31	0.18	0.10	1	00	010	011
31	0.51	0.24	0.15	0.10	0	11	100	101
32	0.64	0.31	0.03	0.02	0	10	110	111

TABLE II: DATA SHEET FOR CASE 1

S. No.	**Network Inputs**				**Target Outputs**								
	Probability				C_A	C_B		C_C			C_D		
	P_A	P_B	P_C	P_D	C_A	C_{B1}	C_{B2}	C_{C1}	C_{C2}	C_{C3}	C_{D1}	C_{D2}	C_{D3}
1	0.36	0.33	0.18	0.13	1	0	0	0	1	0	0	1	1
2	0.47	0.23	0.19	0.11	1	0	1	0	0	0	0	0	1
3	0.54	0.19	0.15	0.12	0	1	1	1	0	0	1	0	1
4	0.40	0.30	0.20	0.10	1	0	0	0	1	0	0	1	1
5	0.94	0.03	0.02	0.01	0	1	0	1	1	0	1	1	1
6	0.69	0.21	0.05	0.05	0	1	0	1	1	0	1	1	1
7	0.77	0.14	0.06	0.03	0	1	0	1	1	0	1	1	1
8	0.67	0.11	0.11	0.11	0	1	1	1	0	0	1	0	1
9	0.84	0.10	0.03	0.03	0	1	0	1	1	0	1	1	1
10	0.60	0.20	0.10	0.10	0	1	0	1	1	0	1	1	1
11	0.40	0.20	0.20	0.20	1	0	1	0	0	0	0	0	1
12	0.50	0.20	0.20	0.10	0	1	1	1	0	0	1	0	1
13	0.63	0.18	0.18	0.01	0	1	1	1	0	0	1	0	1
14	0.58	0.31	0.10	0.01	0	1	0	1	1	0	1	1	1
15	0.44	0.24	0.20	0.12	1	0	1	0	0	0	0	0	1
16	0.81	0.09	0.08	0.02	0	1	1	1	0	0	1	0	1
17	0.55	0.15	0.15	0.15	0	1	1	1	0	0	1	0	1
18	0.37	0.29	0.20	0.14	1	0	1	0	0	0	0	0	1
19	0.41	0.31	0.18	0.10	1	0	0	0	1	0	0	1	1
20	0.51	0.24	0.15	0.10	0	1	1	1	0	0	1	0	1
21	0.64	0.31	0.03	0.02	0	1	0	1	1	0	1	1	1

TABLE III: DATA SHEET FOR CASE 2

S. No.	**Network Inputs**				**Target Outputs**							
	Probability				C_A		C_B		C_C		C_D	
	P_A	P_B	P_C	P_D	C_{A1}	C_{A2}	C_{B1}	C_{B2}	C_{C1}	C_{C2}	C_{D1}	C_{D2}
1	0.25	0.25	0.25	0.25	0	0	0	1	1	0	1	1
2	0.30	0.30	0.20	0.20	0	0	0	1	1	0	1	1
3	0.30	0.30	0.25	0.15	0	0	0	1	1	0	1	1
4	0.33	0.33	0.21	0.13	0	0	0	1	1	0	1	1
5	0.28	0.28	0.28	0.16	0	0	0	1	1	0	1	1
6	0.31	0.23	0.23	0.23	0	0	0	1	1	0	1	1
7	0.29	0.27	0.23	0.21	0	0	0	1	1	0	1	1
8	0.31	0.30	0.28	0.11	0	0	0	1	1	0	1	1
9	0.27	0.26	0.25	0.22	0	0	0	1	1	0	1	1

10	0.26	0.26	0.25	0.23	0	0	0	1	1	0	1	1
11	0.32	0.28	0.22	0.18	0	0	0	1	1	0	1	1

B. Neural Network Architecture: The architecture of the neural network used in this work is the multilayer perceptron (MLP). The number of input and output layer neurons is taken equal to the number of inputs and outputs respectively in the data sheet. The number of hidden layer neurons and the number of hidden layers are found by trial and error method. The activation function considered for hidden layer neurons is tan-sigmoid and for output layer neurons is linear. Multiple layers of neurons with nonlinear transfer functions e.g. sigmoid allow the network to learn nonlinear and linear relationships between network inputs and outputs. The linear output layer lets the network produce values outside the range -1 to +1.

1) *Network architecture for case 1:* There are 4 input and 9 output neurons as determined by its data sheet (Table II). Number of hidden neurons and hidden layers found by trial and error method are 6 and 1 respectively. It is shown in Fig. 2.
2) *Network architecture for case 2:* There are 4 input and 8 output neurons as determined by its data sheet (Table III). Number of hidden neurons and hidden layers found by trial and error method are 4 and 1 respectively. It is shown in Fig. 3.

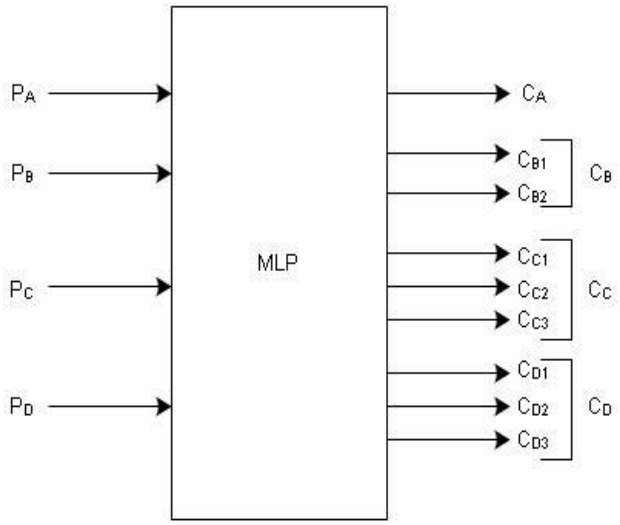

Fig. 2 Network architecture for case 1

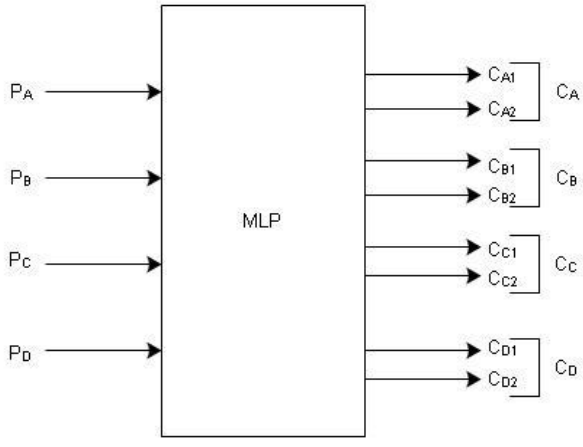

Fig. 3 Network architecture for case 2

C. Training: The training of the above networks is accomplished by using BP algorithm. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function. The default performance function for feedforward networks is mean square error (MSE), which is the average squared error between the network outputs and the target outputs. Further BP learning is implemented in batch mode where adjustments are made to the network parameters on an epoch-by-epoch basis, where each epoch consists of the entire set of training examples. The

Levenberg-Marquardt (LM) algorithm is incorporated into the BP algorithm for training the networks because this is the fastest training algorithm for networks of moderate size. In each case, the training is stopped if the performance function drops below predefined tolerance level set at 10^{-5} in this work otherwise training is continued for another epoch. Training is shown in Fig. 4 and Fig. 5.

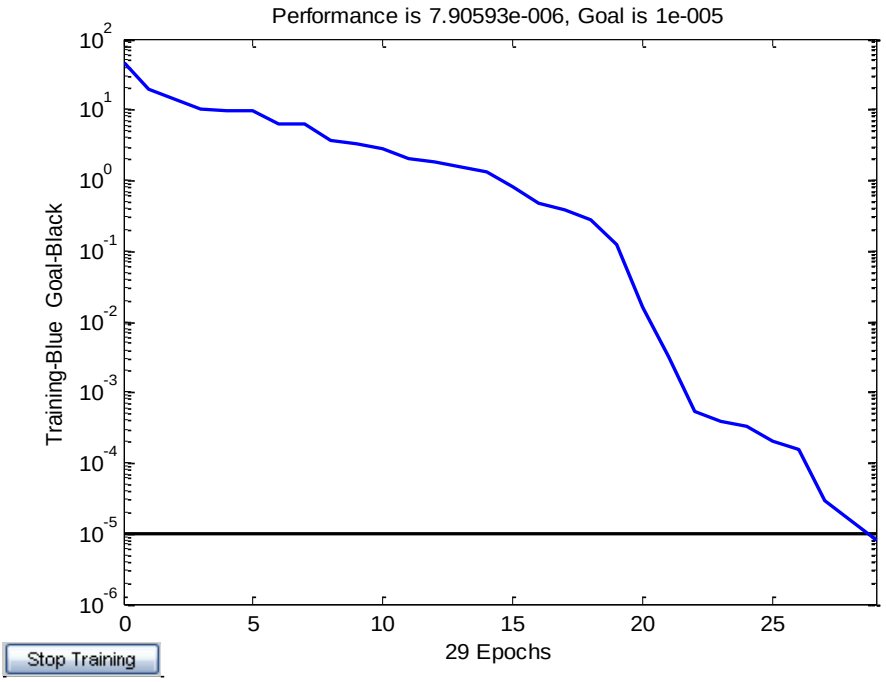

Fig. 4 Training for case 1

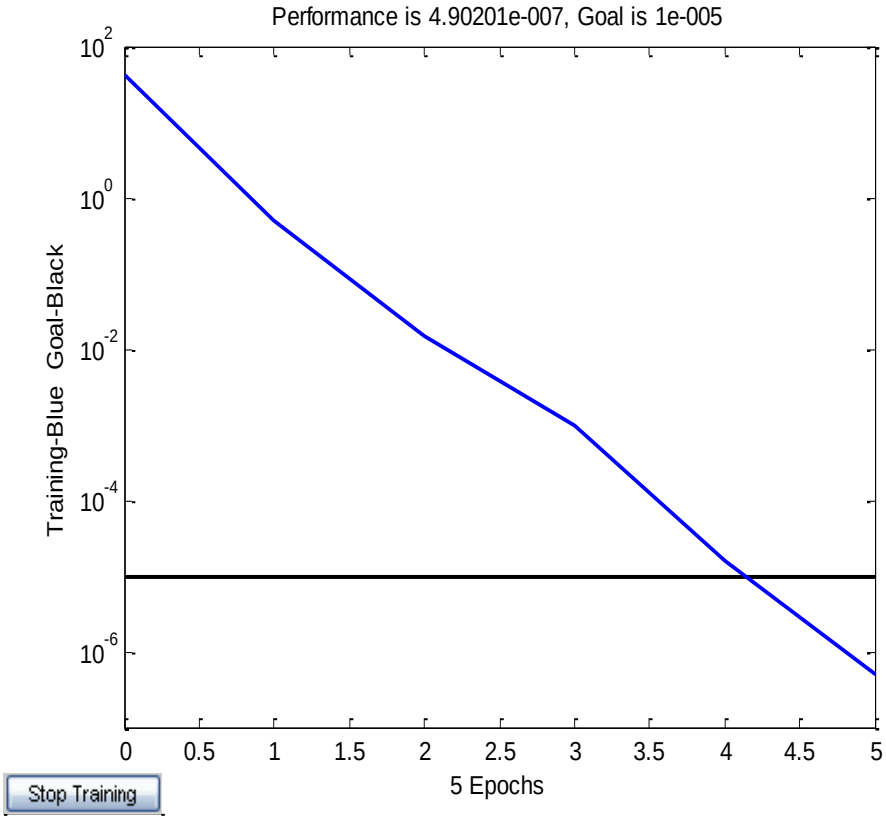

Fig. 5 Training for case 2

D. Development of General Model

A general model has been developed by combining the network architectures shown in Fig. 2 and Fig. 3. During encoding, specific network architecture is selected automatically depending upon given probabilities of symbols. The flow chart for this general model is shown in Fig. 6.

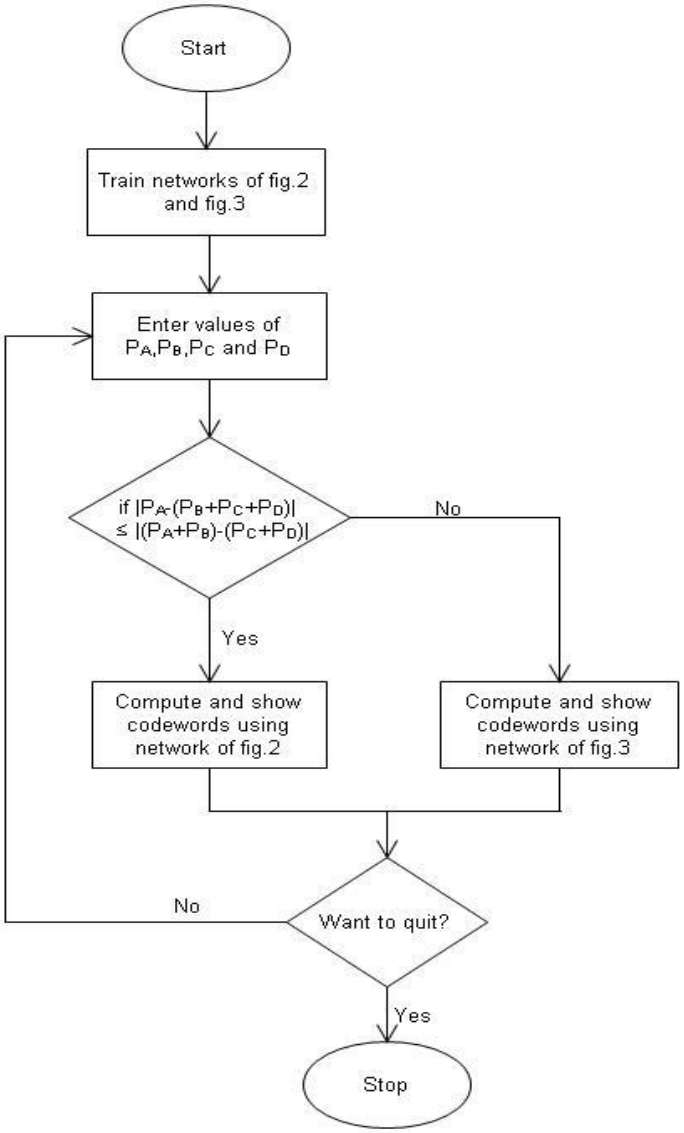

Fig. 6 Flow chart for general model

TABLE IV: COMPARISON OF ANN RESULTS WITH THEORETICAL RESULTS

S. No.	Probability				ANN results				Theoretical results			
	P_A	P_B	P_C	P_D	C_A	C_B	C_C	C_D	C_A	C_B	C_C	C_D
Case 1												
1	0.47	0.21	0.19	0.13	1	01	000	001	1	01	000	001
2	0.56	0.33	0.06	0.05	0	10	110	111	0	10	110	111
3	0.71	0.21	0.04	0.04	0	10	110	111	0	10	110	111
4	0.71	0.10	0.10	0.09	0	11	100	101	0	11	100	101
5	0.89	0.05	0.04	0.02	0	11	100	101	0	11	100	101
6	0.39	0.30	0.20	0.11	1	01	000	001	1	01	000	001
7	0.42	0.34	0.21	0.03	1	00	010	011	1	00	010	011
8	0.60	0.30	0.05	0.05	0	10	110	111	0	10	110	111
9	0.34	0.32	0.18	0.16	1	01	000	001	1	01	000	001
Case 2												
1	0.32	0.32	0.22	0.14	00	01	10	11	00	01	10	11
2	0.29	0.29	0.29	0.13	00	01	10	11	00	01	10	11
3	0.37	0.21	0.21	0.21	00	01	10	11	00	01	10	11
4	0.28	0.28	0.22	0.22	00	01	10	11	00	01	10	11
5	0.27	0.26	0.24	0.23	00	01	10	11	00	01	10	11
6	0.32	0.30	0.28	0.10	00	01	10	11	00	01	10	11

IV. RESULTS AND DISCUSSION

Each trained network has been tested with a test set, in which the outcomes are known but not provided to the network, to see how well the training has worked. The test set is used to test what errors will occur during future network application. This set is not used during training and thus can be considered as consisting of new data entered by the user for the neural network application.

When test set is presented to the network, it is seen that the output of network is in decimal notation. Since actual output is either 0 or 1 so ANN results are rounded to the nearest integer means if ANN result is near 0 e.g. 0.0016, 0.0031, -0.0005 then it is rounded to 0 and if it is near 1 e.g. 1.0009, 0.9999, 0.9987 then it is rounded to 1. ANN results have been compared with theoretical results and comparison is given in Table IV which shows that ANN results are matching with theoretical results. Thus once the network is trained, accuracy of 100% is achieved successfully for new data.

V. CONCLUSIONS

A new application of ANNs in the field of source encoding has been explored. A general ANN model for Huffman encoding scheme has been developed successfully for 4 input symbols. Matching of ANN results with theoretical results verifies the accuracy of developed model. This approach gives a fast and easy way to write Huffman code. This model can be used as a teaching aid as well as in any application requiring encoding of data.

REFERENCES

[1] D.A. Lelewer and D.S. Hirschberg, "Data compression", *Computing Surveys*, vol. 19, pp. 261-296, Sept. 1987.

[2] D.A. Huffman, "A method for the construction of minimum redundancy codes", in *Proc. IRE*, Sept. 1952, vol. 40, pp. 1098-1101.

[3] X. Yao, "Evolving artificial neural networks", in *Proc. IEEE*, Sept. 1999, vol. 87, pp. 1423-1447.

[4] B. Yegnanarayana, *Artificial Neural Networks*, New Delhi, India: Prentice-Hall of India Private Limited, 2006.

[5] S. Shetty and K.K. Achary, "Audio data mining using multi-perceptron artificial neural network", *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 8, pp. 224-229, Oct. 2008.

[6] D. Shanthi, G. Sahoo and N. Saravanan, "Designing an artificial neural network model for the prediction of Thrombo-embolic stroke", *International Journals of Biometric and Bioinformatics (IJBB)*, vol. 3, pp. 10-18, Jan. 2009.

[7] M.W. Gardner and S.R. Dorling, "Artificial neural networks (The multilayer perceptron)—A review of applications in the atmospheric sciences", *Atmospheric Environment*, vol. 32, pp. 2627-2636, June 1998.

[8] D.E. Rumelhart, G.E. Hinton and R.J. Williams, *Learning Internal Representations by Error Propagation*, in D.E. Rumelhart and J.L. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA: MIT press, 1986, vol. 1, pp. 318-362.

[9] M. Sharma, "Compression using Huffman coding", *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 10, pp. 133-141, May 2010.

[10] W. Chanhemo, H.R. Mgombelo, O.F. Hamad and T. Marwala, "Design of encoding calculator software for Huffman and Shannon-Fano algorithms", *International Science Index*, vol. 5, pp. 500-506, June 2011.

[11] D. Scocco (2011) Implementing Huffman coding in C. [Online]. Available: http://www.programminglogic.com/implementing-huffman-coding-in-c/

