International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online), Volume 2, Special Issue 4, June 2014)

International Research Conference on Engineering, Science and Management 2014 (IRCESM 2014)

An Intelligent Technique for Image Compression

Athira Mayadevi Somanathan[1], V. Kalaichelvi[2]

[1]Dept. Of Electronics and Communications Engineering, BITS Pilani, Dubai, U.A.E.
[2]Dept. Of Electronics and Instrumentation Engineering, BITS Pilani, Dubai, U.A.E.

[1]athiranath@gmail.com
[2]kalaichelvi@dubai.bits-pilani.ac.in

***Abstract*— Classic image compression techniques such as JPEG and MPEG have serious limitations at high compression rates – the decompressed image gets really fuzzy or indistinguishable. To mitigate this problem Artificial Neural Network (ANN) techniques, considered to be intelligent and adaptive models, are used for image compression. In this paper, image compression is achieved by implementing an intelligent technique like the backpropagation neural network algorithm. Analysis is also done on achieving a good quality compressed image by varying and comparing the neural network parameters such as number of hidden layer nodes, momentum constant and number of epochs.**

***Keywords*— Artificial Neural Networks, Backpropagation Neural Network, Image Compression**

I. INTRODUCTION

Over the last ten years we have been witnessing an evolution in the way we converse – the perpetually developing internet, the rapid progress of mobile communications, and the ever expanding significance of video communications are all a part and parcel of this essential transformation. All these features of multimedia revolution are possible because of data compression. If it were not for data compression algorithms, it would not have been possible to put images, audio or video on websites [1]. Artificial Neural Networks are gaining immense popularity for their ability to solve complex real-world problems in image compression. They have displayed their superiority in tackling noisy or blurred data [2]. Over the years, these computational models have been used for various applications, particularly for compression of data. Their ability to pre-process input sequence to generate less sophisticated sequences with fewer components makes them well-suited for this application [3]. The application of neural network for image compression with lifting scheme and RLC was explained by Srikala etal [4]. Matsuoka R et al have analyzed the quantitative analysis of image quality of lossy compression images [5]. The objective of the present work includes implementation of backpropagation neural network for image compression using MATLAB software.

The quality and compression ratio of an image is compared by varying network training parameters such as momentum constant, number of epochs and number of hidden layer nodes.

II. DATA COMPRESSION

In computer science and information theory, data compression is essentially encoding of data using lesser number of bits than the actual representation. . In short, it is the science of portraying information in a condensed form. In a compression technique, there are two algorithms involved – the compression algorithm and the reconstruction algorithm. The compression algorithm produces an output with less number of bits to depict the information as compared to the input. The compressed information is then reconstructed back to the original representation by using the reconstruction algorithm. A new algorithm for data compression is being developed by I Made Agus and Dwi Suarjaya [6]. The use of data compression technique for wireless sensor network technology for environmental monitoring is analyzed by Capo-chichi et al [7].

III. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) are models influenced by animal Central Nervous System (CNS), and these networks are capable of machine learning and pattern recognition. ANNs are normally represented as an organized web of interconnected neurons that can determine values from inputs by feeding information through the network.

Neural network structures are based on the singular hope that atleast some of the adaptability and power of the human brain can be replicated by artificial means. These networks consist of many simple computing elements connected together by varying strength, a gross abstraction of the brain, which in reality consists of large numbers of far more complex neurons connected together with far more complex and structured couplings [8].

A. Neural Network Structure

Neural Networks are modeled after biological neural network structures. Figure1 shows a model neuron which is the starting point for neural networks.

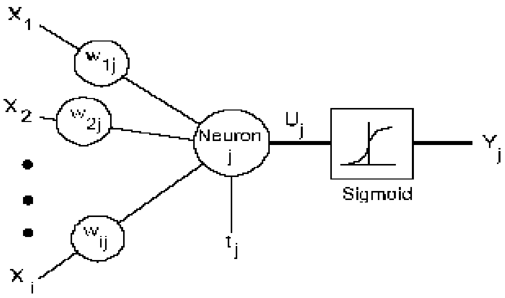

Fig. 1. Neuron Model

The model neuron shown here consists of several inputs and a single output. The input values are altered by multiplying with ‘weights’. These weighted inputs are then combined and referenced against a threshold value and activation function, and the output is subsequently determined.

B. Learning and Training in Artificial Neural Networks

The ability to learn is a fundamental trait of intelligence [9]. The process of learning in ANN can be thought of as the problem of amending the network model and the interconnecting weights, so that it can perform a given task effectively. The multilayer architecture, though is very efficient at solving complex problems, also introduces the problem of training the hidden layers for getting the desired output which is unknown. The neural network must learn the weights from the training patterns that are available. This ability of the ANNs to automatically learn from examples sets them apart from the traditional systems and makes them attractive. The weights are iteratively updated in the network, leading to better performance over time. There are three main modes of learning [10] - supervised learning, unsupervised learning and hybrid learning. As the supervised learning has a network that is provided with a correct output for every pattern that is given as an input to the network, this type of leaning is being used in the present work.

IV. BACKPROPAGATION NEURAL NETWORK ALGORITHM

The backpropagation neural network algorithm, a method to monitor learning, is a multilayer feedback network, trained according to error back propagation algorithm and is the most pervasive and popular neural network algorithms in use today. Also known as the ‘generalized delta rule’, it is a supervised learning method.

The back propagation learning algorithm can be detailed as follows:

1. Forward propagation of operating signal – The input signal fed into the network is propagated to the output layer through the hidden layer. During this propagation, the network weights and offsets are maintained constant, and based on the current state of the synaptic weights, the network produces some output. This output is compared to the expected good output, and if the two do not match, back propagation of error signal takes place.
2. Back propagation of error signal and updating of weights – The mean squared error (MSE) signal is computed and this value is propagated backwards from the output layer to the input layer. During this procedure, small changes are made to the weights and the weight value of the network is regulated by the error feedback. This whole cycle is repeated until the error value falls below a certain threshold value.

Firstly, initialization of the neural network is done by fixing all its weights to be small random numbers between –1 and +1. The output is then calculated according to the pattern that was provided at the input. This process is referred to as the ‘forward pass’. Since initially all the weights are random, the calculated output would be different from the target output. Error, which is Target – Calculated Output is then found for each neuron. This error value is then used to alter the weights in such a manner as to make the error smaller in subsequent passes/iterations. This process where the neuron output gets closer to its specified target value is known as the ‘reverse pass’. The cycle is repeated over and over again until a minimum error value is obtained.

V. Image Compression Using Backpropagation Algorithm

In this paper, the backpropagation algorithm is used to compress data, specifically images. Methods based on ANN provide means for image compression at the input side (transmitter), and decompression at the output side (receiver). ANN techniques are efficient at maintaining security of the data, in addition to providing adequate data compression rates.

The neural network structure for image compression is shown in Figure 2. In order to achieve compression of the image, the number of hidden layer neurons must be less than that of the input and output layer neurons.

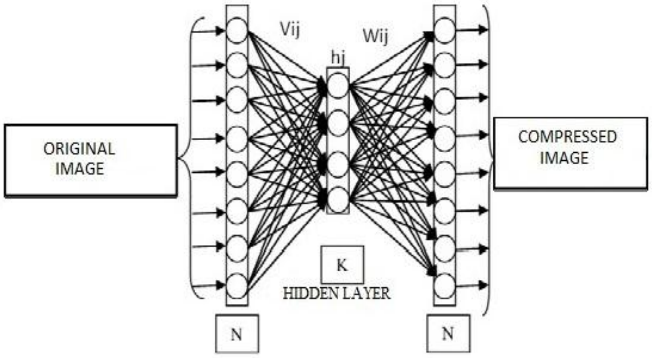

Fig. 2. Neural Network for Image Compression

The image to be compressed is first passed through the input layer and then through a hidden layer consisting of very small number of neurons. This hidden layer stores the compact features of the image, and hence, the number of neurons in this layer needs to be smaller in order to achieve a higher compression ratio. The compressed image is finally observed after passing through the output layer. This compressed image retains much of the input data, while discarding all the redundant information. In the present work, the network architecture used for image compression is [256 240 256] where 256 corresponds to the number of nodes in the input, 240 corresponds to number of nodes in the hidden layer (which is found to be optimum for compression in this scenario, as shown by the simulation results) and 256 corresponds to number of nodes in the output layer.

VI. Simulation Results In Matlab

Study of image compression using Back Propagation Neural Network (BPNN) was done on images and image compression was subsequently simulated in MATLAB 8.0 [11]. A ‘Lena’ image of size 65.9KB was used for this purpose as shown in Figure 3.

Fig. 3. Lena image before compression

The image is saved in the MATLAB directory or pathway so that it can be invoked in the program. Since the image is a colored image, it had to be gray scaled so that it can be given as an input to the neural network. This gray scaled image is then resized to a pixel size of 256x256, so that the input and output layers have the same number of neurons (256 neurons). The feedforward neural network was then invoked and the training parameters were defined. The function used for the purpose of network training was the gradient descent with momentum and adaptive learning rate backpropagation. After specifying the number of hidden layer nodes (which must be less than the number of input layer neurons), the resized and gray scaled ‘Lena’ image was fed to the neural network. Depending on the quantity of hidden neurons, the learning rate, the number of epochs and the momentum constant, various compressed images were produced, and the compression ratio was calculated for different cases to find the optimum parameters. Simulation studies were performed by varying the network parameters, such as the hidden nodes, number of epochs or iteration, the momentum constant and the learning rate.

A. Effect of Varying Momentum Constant on Compression Ratio

In the first set of analysis, the momentum constant was varied from 0.2 to 0.8, keeping all the other parameters constant. The number of epochs was fixed at 10,000, the learning rate was set to 1, and the number of hidden layer nodes was set to 240 (a value less than the input number of nodes of 256). Table I summarizes the simulation results as observed by varying the momentum constant.

TABLE I
SIMULATION RESULTS SHOWING EFFECT OF VARYING MOMENTUM CONSTANT

	Case 1	Case 2	Case 3	Case 4	Case 5
No Of Epochs	10000	10000	10000	10000	10000
Hidden Layer Nodes	240	240	240	240	240
Learning Rate	1	1	1	1	1
Momentum Constant	0.2	0.4	0.45	0.6	0.8
Time (min:sec)	01:18	02:50	01:05	00:39	01:05
Size (KB)	30.8	27.9	31.8	32	29
Compression Ratio	2.139	2.362	2.072	2.059	2.272
No of Iterations	164	368	139	80	135

As observed from Table I, a momentum constant of 0.7 resulted in a higher compression ratio. Though this gave an image that was highly compressed, the quality of the image was severely degraded, and it also required a higher simulation time compared to the rest in the analysis set.

From this analysis, it was observed that the compression ratio improved with the momentum constant only up to a certain point (upto 0.7), after which the quality of the image was severely compromised. A good quality image in this analysis was formed when the momentum constant was set to 0.45 (Case 3 in Table I). Fig 4 shows the output of Case 3.

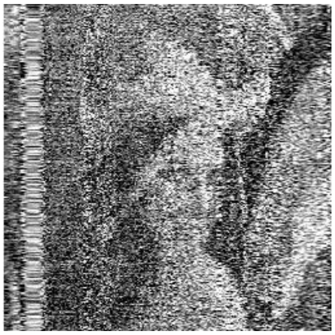

Fig. 4. Compressed image formed with momentum constant set to 0.45

A. Effect of Varying Number of Hidden Nodes on Compression Ratio

In this set of analysis, the hidden layer nodes were changed from 50 to 200 in steps of 50, keeping the momentum constant as 0.45, the learning rate as 1, and the epoch count as 5000. Finally, the number of nodes in hidden layer was set to 240. Table II shows the result of this analysis.

TABLE II
SIMULATION RESULTS SHOWING EFFECT OF VARYING NUMBER OF HIDDEN LAYER NODES

	Case 1	Case 2	Case 3	Case 4	Case 5
No Of Epochs	5000	5000	5000	5000	5000
Hidden Layer Nodes	50	100	150	200	240
Learning Rate	1	1	1	1	1
Momentum Constant	0.45	0.45	0.45	0.45	0.45
Time (min:sec)	00:14	00:45	00:49	01:28	00:57
Size (KB)	26.6	27.2	28.7	29.6	31.4
Compression Ratio	2.477	2.423	2.296	2.226	2.099
No of Iterations	148	261	180	237	124

Table II shows that as the number of hidden layer nodes increases, the image compression ratio also decreases, giving rise to less compressed images, i.e., fewer the number of hidden layer nodes, the higher the compression ratio. Though the best compression ratio was achieved when the hidden layer nodes were set to 50, the image quality was inferior as compared to the output image formed in Case 5 where the number of hidden layer nodes is set to 240. Fig 5 shows the output compressed image of Case 5. The first two analyses show that a momentum constant of 0.45 and 240 hidden layer nodes gives a distinguishable and good quality compressed image.

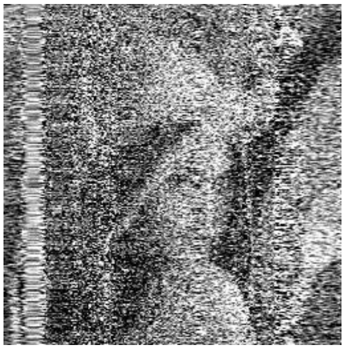

Fig. 5. Compressed image formed with number of hidden layer nodes set to 240

B. *Effect of Varying Number of Epochs on Compression Ratio*

As the previous analyses showed, a value closer to 10,000 for the number of epochs results in a better quality compressed image. Hence, in this set, the number of epochs was varied from 11,000 to 14,000 in steps of 1000, and finally analysis was also done for a value of 20,000. The learning rate was a constant value of 1 for this entire set. The momentum constant and the hidden layer nodes were also set to a fixed value of 0.45 and 240 respectively. Table III summarizes the results of this analysis.

As clearly observed from Table III, a higher compression ratio is observed in Case 5 when the number of epochs is set to 20,000, but a better quality image is formed in Case 1, when the epoch number is set to 11,000. Figure 6 shows the output image of Case 1.

TABLE III
SIMULATION RESULTS SHOWING EFFECT OF VARYING NUMBER OF EPOCHS

	Case 1	Case 2	Case 3	Case 4	Case 5
No Of Epochs	11000	12000	13000	14000	20000
Hidden Layer Nodes	240	240	240	240	240
Learning Rate	1	1	1	1	1
Momentum Constant	0.45	0.45	0.45	0.45	0.45
Time (min:sec)	03:59	01:42	01:03	01:17	02:41
Size (KB)	28.6	29.7	31.6	30.2	28.4
Compression Ratio	2.304	2.219	2.085	2.182	2.320
No of Iterations	440	222	137	163	343

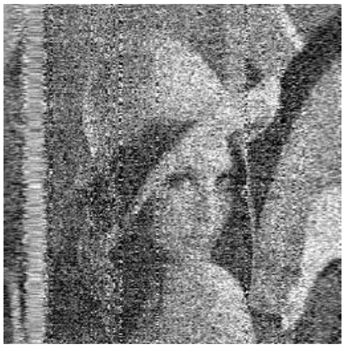

Fig. 6. Compressed image formed with number of epochs set to 11,000

C. Results and Discussion

Through the different analyses sets that were studied, it was observed that the quality of the image as well as the compression ratio depends on its network parameters – epoch number, momentum constant, hidden layer nodes, and learning rate. Over the course of this research work, it was observed that a sufficiently high epoch number, a higher simulation time, combined with a higher learning rate, and a sufficient but not too large momentum constant resulted in a good quality compressed image. Epoch number closer to 10,000 resulted in better quality images, and though a value of 20,000 simulated an image that was highly compressed, the image quality was severely degraded. The analysis showed that a momentum constant of 0.45, a learning rate of 1, 240 hidden layer nodes and 11,000 epochs resulted in a highly compressed and good quality image.

VII. CONCLUSION

In this paper, artificial neural networks were employed for the particular application of image compression. Images were given as the input to the network, and various compressed images were produced, depending on the variation of neural network parameters. In the future, this work can be extended to not just images, but also to other kinds of data, such as texts, for learning about the efficiency of ANNs in data compression.

[illegible]

The authors are grateful to the authorities of BITS Pilani, Dubai Campus, U.A.E. for the encouragement and support to carry out the research work.

REFERENCES

[1] Guy E Blelloch, "Introduction to Data Compression," Computer Science Department, Carnegie Mellon University, January 31, 2013. Available: http://www.blellochcs.cmu.edu

[2] A. Khashman, and K. Dimililer, "Comparison Criteria for Optimum Image Compression," Computer as a Tool, EUROCON 2005, The International Conference on IEEE, vol. 2, 2005.

[3] R. D. Dony, and S. Haykin, "Neural Network Approaches to Image Compression", Proc. IEEE, vol. 83, 1995, pp. 288-303.

[4] Srikala P, and Umar S., "Neural Network Based Image Compression with Lifting Scheme and RLC," International Journal of Research in Engineering and Technology, vol. 1, no. 1, 2012, pp. 13-19

[5] Matsuoka R, Sone M, Fukue K, Cho K, and Shimoda H., "Quantitative Analysis of Image Quality of Lossy Compression Images," International Society for Photogrammetry and Remote Sensing, 20 September, 2013. Available: http://www.isprs.org/proceedings/XXXV/congress/comm3/papers/348.pdf.

[6] D. Suarjaya, and I. Made Agus, "A New Algorithm for Data Compression Optimization," International Journal of Advanced Computer Science & Applications, vol. 3, no. 8., 2012

[7] Capo-Chichi, Eugène Pamba, Hervé Guyennet, and J. M. Friedt, "K-RLE: A New Data Compression Algorithm for Wireless Sensor Network," Sensor Technologies and Applications, SENSORCOMM'09, Third International Conference on IEEE, 2009, pp. 502-507.

[8] B. Kosko, Neural networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice-Hall, 1992

[9] G. U. Chaudhari, and B. Mohanty, "Function Approximation Using Back Propagation Algorithm in Artificial Neural Networks," Ph.D. dissertation, Dept. Elect. Eng., NIT Rourkela, Odisha, 2007

[10] A. K. Jain, J. Mao, and K. M. Mohiuddin, Artificial Neural Networks: A tutorial, IEEE Computer, vol. 29, no. 3, 1996, pp. 31-44.

[11] Athira Mayadevi Somanathan, "A Neural Network Approach for Data Compression," Submitted to BITS Pilani, Dubai Campus as part of the project work, 2013.