Journal of Engineering Science and Technology
Vol. 2, No. 2 (2007) 188 - 199

LEARNING ALGORITHM EFFECT ON MULTILAYER FEED FORWARD ARTIFICIAL NEURAL NETWORK PERFORMANCE IN IMAGE CODING

OMER MAHMOUD[1], FARHAT ANWAR[1], MOMOH JIMOH E. SALAMI[2]

[1] Department of Electrical and Computer Engineering, Faculty of Engineering, International Islamic University Malaysia, MALASIA.
[2] Department of Mechatronics Engineering, Faculty of Engineering, International Islamic University Malaysia, MALAYSIA.
*Corresponding Author: farhat@iiu.edu.my

Abstract

One of the essential factors that affect the performance of Artificial Neural Networks is the learning algorithm. The performance of Multilayer Feed Forward Artificial Neural Network performance in image compression using different learning algorithms is examined in this paper. Based on Gradient Descent, Conjugate Gradient, Quasi-Newton techniques three different error back propagation algorithms have been developed for use in training two types of neural networks, a single hidden layer network and three hidden layers network. The essence of this study is to investigate the most efficient and effective training methods for use in image compression and its subsequent applications. The obtained results show that the Quasi-Newton based algorithm has better performance as compared to the other two algorithms.

Keywords: Image Compression /Decompression, Neural Network, Optimisation

1. Introduction

The need for effective data compression is evident in almost all applications where storage and transmission of digital images are involved. For examples an 1024 X1024 color image with 8 bit/pixel generates 25Mbits data, which without compression requires about 7 minutes of transmission time over 64 kbps line. A Compact Disk with storage capacity of 5 Gbits can only hold about 200 uncompressed images. Reducing the image size by applying compression techniques is usually possible because images contain a high degree of spatial

Nomenclatures

B	Approximate inverse Hessian matrix
E	Error function over all neurons in output layer
e	Error signal at the output layer of the i^{th} neuron at iteration n
g	Gradin vector
H	Hessian matrix
l	Index of the image blocks
n	Number of iterations
p	Direction vector
w	Vector of all weights
X_i	Desired output
X'_i	Actual output

Greek Symbols

$\acute{\eta}$	Learning rate
α	Constant

redundancy due to correlation between neighbouring pixels. There are two basic types of image compression: lossless compression and lossy compression. A lossless scheme encodes and decodes the data perfectly, and the resulting image matches the original image exactly. Lossy compression schemes allow redundant and nonessential information to be lost and it has higher compression ratio. Compression ratio is simply the size of original image divide be the size of the compressed one. Typically with lossy compression methods there is a tradeoff between compression ratio and obtained image quality. Apart from the existing technology on image compression represented by series of JPEG, MPEG standards, new technology such as neural networks and genetic algorithms are being developed to explore the future of image coding [1, 2]. Oja [3] proposed the use of a simple neural network that can perform nonlinear principal component analysis as a transform-based method in image compression. Since this pioneering work, several new learning algorithms have been proposed for extending this approach [4, 5].

Learning algorithms has significant impact on the performance of neural networks, and the effects of this depend on the targeted application. The choice of suitable learning algorithms is therefore application dependent. This paper presents a comparative study that shows the effect of using different learning algorithms on the performance of Multilayer Feed Forward Artificial Neural Network (MFFANN) with Error Back- propagation Algorithm in images coding. The paper starts with an Introduction followed by an overview of Multilayer Feed Forward Artificial Neural Network with Error Back propagation Algorithm approach in image compression. It then explains the functionalities of three different learning algorithms. Finally the results obtained from the simulation study are presented.

2. Image Compression Using MFFANN

This section presents the architecture of a Feed Forward Neural network that is used to compress images in the research work. The MFFANN consists of one Input Layer (IL) with N neurons, one Output Layer (OL) with N neurons and one (or more) Hidden Layer (HL) with Y neurons where, the network is trained using Error Back propagation Algorithm. The network is designed in a way such that N>Y, where N is input layer/output neurons and Y is hidden layer neurons as shown in Fig. 1

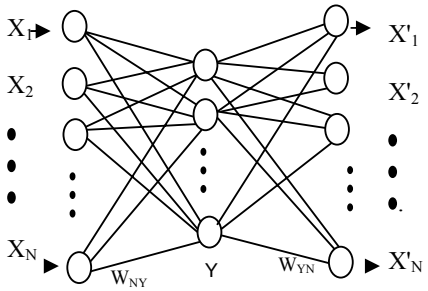

Fig. 1. Three layers Neural Network.

Compression Yi = ∑ Wji * Xi, decompression X'j= ∑W'ij * Yj

When MFFANN is used for image compression the process require the following three steps

2.1 Training the MFFANN

The input image is split up into blocks or vectors of 4×4, 8×8 or 16×16 pixels. These vectors are used as inputs to the network. The network is provide by the expected (or the desired) out put, and it is trained so that the coupling weights, $\{w_{ji}\}$, scale the input vector of N-dimension into a narrow channel of Y-dimension ($Y < N$) at the hidden layer and produce the optimum output value which makes the quadratic error between output and the desired one minimum. In fact this part represents the learning phase, where the network will learn how to perform the task. In this process of leering a training algorithm is used to update network weights by comparing the result that was obtained and the results that was expected. It then uses this information to systematically modify the weight throughout the network till it finds the optimum weights matrix. As explained in section 3

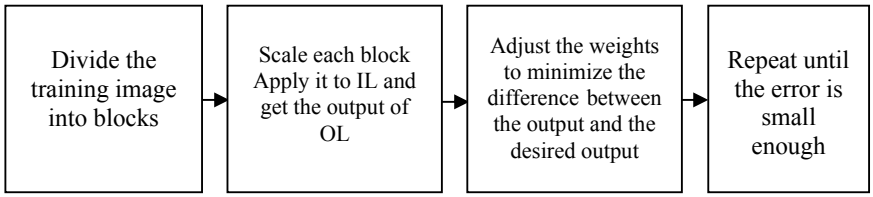

Fig. 2. Block Diagram of MFFANN Training

2.2 Encoding

The trained network is now ready to be used for image compression which, is achieved by dividing or splitting the input images into blocks after that scaling and applying each block to the input of Input Layer (IL) then the out put of Hidden layer HL is quantized and entropy coded to represent the compressed image. Entropy coding is lossless compression that will further squeeze the image; for instance, Huffman coding code be used here. Fig. 3 shows the encoding steps

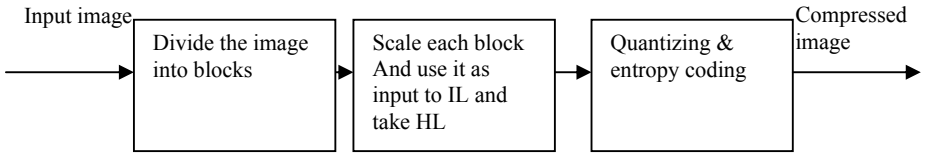

Fig. 3. Block Diagram of the Encoding Steps.

2.3 Decoding

To decompress the image; first decode the entropy coding then apply it to the out put of the hidden layer and get the out put of the OL scale the it and reconstruct the image. Fig. 4 show the decoder block diagram.

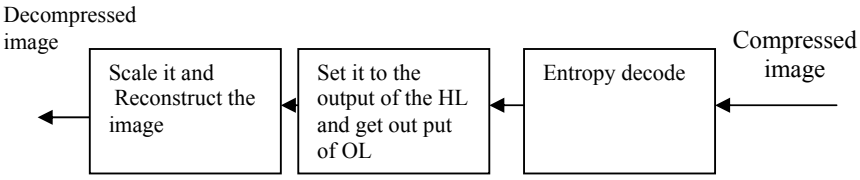

Fig. 4. Block Diagram of the Decoding steps

3. Weight Adjustment

The networks weights need to be adjusted in order to minimise the difference or the error between the output and the expected output. This is explained in the equations below.

The error signal at the output layer of the i^{th} neuron at iteration n is given by

$$e_i(n) = X_i(n) - X'_i(n) \quad (1)$$

where X_i represent the desired out put and X'_i represent the actual out put. The error function over all neurons in output layer is given by Eq. (2).

$$E_l(n) = \sum e_i^2(n) \quad (2)$$

The error function, over all input vectors in the training image, is

$$E = \sum E_l \,,\; E_l = (X', w) \tag{3}$$

where l indexes the image blocks (inputs vector), X' is the vector of outputs, and w is the vector of all weights. In order to minimise the error function with respect to weight vector (w) it is necessary to find an optimal solution (w*) that satisfy the condition

$$E(w^*) \le E(w) \tag{4}$$

The necessary condition for the optimality is

$$\Delta E(w) = 0 \tag{5}$$

where Δ is gradient operator

$$\Delta = \left[\partial / \partial w\right] \tag{6}$$

and ΔE (w) is gradient vector (g) of error function is defined as follows

$$\Delta E(w) = \partial E / \partial w \tag{7}$$

The solution can be obtained using a class of unconstrained optimization methods based on the idea of local iterative descent. Starting with initial guess denoted w(0), generate a sequence of weight vectors w(1),w(2)… such that the error function is reduced for each iteration

$$E(w_{n+1}) \le E(w_n) \tag{8}$$

In this study three optimisation methods will be considered, *Gradient Descent, Conjugate Gradient method and Quasi Newton method.* These methods are the most popular techniques used for iteratively solving unconstrained minimization problems.

3.1 Gradient descent (GD)

In this method of *Gradient descent (also known as steepest descent) [7]* the successive adjustments applied to the weight vector are in the direction of the steepest descent that is the direction opposite to the gradient vector Δ E (w)

$$w_{n+1} = w_n + \Delta w_n \tag{9}$$

$$\Delta w_n = -\eta_n g_n \tag{10}$$

where g is gradient vector and η is a small positive number called the *learning rate*, which is the step size needed to take for the next step. Gradient descent only indicates the direction to move, however the step size or learning rate needs to be decided as well. Too low a learning rate makes the network learn very slowly, while too high a learning rate will lead to oscillation. One way to avoid oscillation for large η is to make the weight change dependent on the past weight change by adding a *momentum* term,

$$\Delta w_{n+1} = -\eta_n g_n + \alpha \Delta w_n \tag{11}$$

That is, the weight change is a combination of a step down the negative gradient, plus a fraction α of the previous weight change, where $0 \le \alpha < 0.9$

3.2 Conjugate gradient methods (CG)

The conjugate-gradient method [7, 8] uses a direction vector which is a linear combination of past direction vectors and the current negative gradient vector. In so doing the conjugate-gradient method reduces oscillatory behavior in the minimum search and reinforces weight adjustment in accordance with previously successful path directions.
Let **p**(*n*) denotes the direction vector at the *n*th iteration of the conjugate-gradient method algorithm. Then the weight vector of the network is updated as a linear combination of the previous weight vector and the current direction vector according to.

$$w_{n+1} = w_n + \eta_n p_n \tag{12}$$

The initial direction vector, **p**(0), is set equal to the negative gradient vector, **g**(0) at the initial weight w(0).
Successive direction vectors are computed as a linear combination of the current negative gradient vector and the previous direction vector as shown below (Fletcher -Reeves version).

$$p_{n+1} = -g_{n+1} + b_n p_n \tag{13}$$

where

$$b_n = \left[g_{n+1}^T g_{n+1}\right] / \left[g_n^T g_n\right] \tag{14}$$

3.3 Quasi Newton methods (QN)

Quasi-Newton methods [7] are based on Newton's method The basic idea behind Quasi-Newton methods is to use an approximation of an inverse Hessian in place of true inverse as required in Newton's method. A typical iteration for this method is

$$w_{n+1} = w_n + \eta_n d_n \text{ ; Where } d_n = -B_n g_n \tag{15}$$

Where B_n is a positive definite matrix (approximate inverse Hessian matrix which is adjusted from iteration to iteration) chosen so that the directions d_n tend to approximate Newton's direction. The step size η_n is usually chosen by a line search. The important idea behind the methods is that two successive iterates x_n and x_{n+1} together with the gradients $\Delta f(x_n)$ and $\Delta f(x_{n+1})$ contain curvature (i.e., Hessian) information, in particular,

$$\Delta f(x_{n+1}) - \Delta f(x_n) \approx H(x_n)(x_{n+1} - x_n) \tag{16}$$

Therefore, at every iteration it would be necessary to choose B_{n+1} to satisfy

$$B_{n+1} q_n = z_n \tag{17}$$

Where

$$z = x_{n+1} - x_n; q = \Delta f(x_{n+1}) - \Delta f(x_n) \tag{18}$$

In the most popular class of Quasi- Newton methods the matrix B_{n+1} is obtained from the previous B_n , vector q and z by suing the following equation. (BFGS version).

$$B_{n+1} = B_n + zz^T / q^T q - B_n q q^T B_n / q^T B_n q + q B_n q^T + z / z^T z - B_n q / q^T B_n q \quad (19)$$

4. Results and Discussion

The MFFANN with three layer and five layers is implemented. The network is trained using Error Back propagation Algorithm that utilizes optimization methods described in section 3. The obtained results are presented in the subsequent sections.

4.1 Learning algorithms performance during training

In order to find the best method for training a neural network that performs image compression and decompression. The above mentioned methods used to update the weights in Feed forward neural network with error back propagation algorithms. The Initial layers weight matrix was the same for all methods. Table 1 shows the details obtained for each method and Fig. 5 show performances (RMSE vs. Number of Epochs) of the three methods during training. The data shown in table 1 and Fig. 5 belong to a network trained using the image in Fig. 6a. Furthermore, similar results were obtained using different sample images, for instance MIR images, and different number of epochs (e.g. 700, 1000).

Table. 1. Performance Parameters During Training.

	Epochs	Time (sec)	RMSE	Image size (pixels)	Compression ratio (IL/HL)
GD	400	17.22	$149.83 * 10^{-5}$	291x240	4:1
CG	400	39.70	$18.925 * 10^{-5}$	291x240	4:1
QN	400	41.09	$11.110 * 10^{-5}$	291x240	4:1

In Fig. 5, the x-axis represent the training time in seconds and the y-axis represents the performance of the network in term error between in input and output. Image size in pixels refers to the size of image used in the training. Fig. 5 and Table 1 show that GD method takes less time as compared to CG and QN methods. However in term of error between the input and the output QN has performed better than other two methods. Despite the fact that QN methods take longer time during training it the suitable among the three methods since it has least RMSE error. As results the decompressed image has higher quality compared to the other two methods. CG training time is slightly less than QN

time. The CG decompressed images quality could be enhanced by increasing the number of training epochs.

Fig. 6 show the resulted output from the previous training for each methods of weight updating

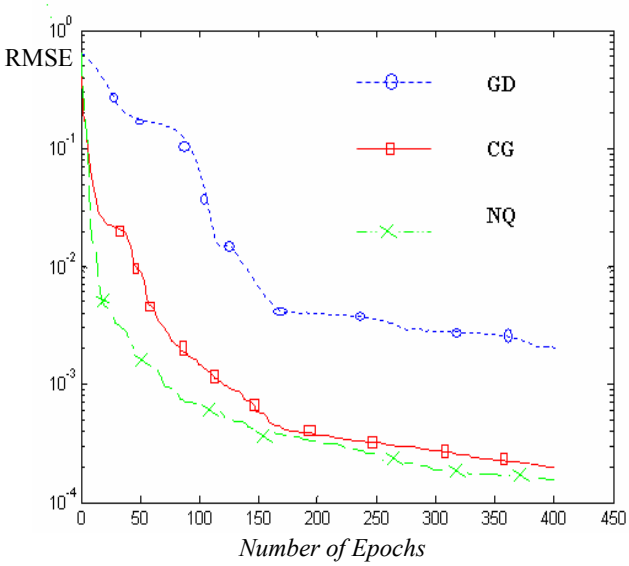

Fig. 5. Training Performance.

An important feature of neural networks is the generalization ability which, refers to the performance ability of the network with new data that were not used during training [7]. To test the network generalization ability the pervious Network has been used to compress the image shown in Fig. 7a corresponding decompressed images are shown in Fig. 7b and Fig. 7c. The image in Fig. 7.c has higher quality compared to image in Fig. 7b which indicates that QN has better performance with respect to processing new images.

4.2 Still image compression /decompression

The following sections present the results obtained for MFFANN with one hidden layer and three hidden layers. Networks trained using error back propagation with QN optimization methods to update the weights. The compression ratio shown here represents ratio between input layer and hidden layer. Without entropy coding, that can compress the image further, up to three or four times, with out affecting its quality.

a. Original image

b. Decompressed image 4:1using GD

c. Decompressed Image 4:1e using CG

d. Decompressed Image 4:1 using QN

Fig. 6. Original Image and Decompressed Image.

The results in Fig. 8 show that it possible to achieve good decompressed image with ratio 8:1 using neural networks with different number of neurons and layers. For instance, the image in Fig. 8b is obtained using network with 16 neurons in the input/output layer and single hidden layers with 2 neurons while the image in Fig. 8c obtained using network with 64 neurons in the input/output layer and single hidden layers with 8 neurons. Fig. 8d is obtained using network with five layers, two input and output layers with 64 neurons and three hidden layers with 16, 8 and 16 neurons.

Generally the training time will increase for increased number of neurons and layers. It is also observed, in general, that single hidden layer network compressed image has a better quality compared to the three hidden layers network compressed one.

a. Original Image.

b. CG Decompressed Image.

d. QN Decompressed Image.

Fig. 7. Generalization Test

4.3 Video compression

This section discusses an extension of the MFFANN compression approach to compress video frames. In order to handle video compression, the MFFANN compression applied to each frame in a sequence which resulted in a compressed video sequence similar to the technique used in Motion-JPEG [10]. Fig. 9 shows the results obtained by using this approach on video frames Fig. 9a show original frame. Fig. 9b is the corresponding decompressed frame

a. Original Image.

b. Decompressed Image 8:1
RSME=0.0518
Original Image Sampled 4×4 Pixels
(One Hidden Layer)

c. Decompressed image 8:1
RMSE = 0.0431
Original Image Sampled 8×8 Pixels
(One Hidden Layer)

d. Decompressed Image 8:1
RMSE = 0.0630
Original Image Sampled 8×8 Pixels
3 Hidden Layers (16, 8 and 16 neurons)

Fig. 8. Decompressed Images of Single and Three Hidden Layers Networks

a. Original Frame.

b. Decompressed Frame.

Fig. 9. Video Frames

5. Conclusion s

In this paper neural network has been used for image compression. Based on the obtained results the gradient descent takes less time during training as compared to Conjugate Gradient methods and Quasi Newton methods. However Quasi Newton performs better in term of minimizing the error as such the image compressed by Quasi Newton has a higher quality and better generalization ability. It is also observed that, in general, one hidden layer network compressed image has better quality opposed to the three hidden layers network compressed one.

References

1. Soliman, H. S. & Omari, M. (2006). A neural networks approach to image data compression. *Applied Soft Computing*, 6(3), 258-271.
2. Laskaris, N. A. & Fotopoulos, S. (2004). A novel training scheme for neural-network-based vector quantizers and its application in image compression. Short Communication. *Neurocomputing*, 61, 421-427.
3. Oja, E. (1982). A simplified neuron model as a principal component analyzer. *J. Math. Biol*.**,** 15**,** 267–273.
4. Sanger, T. D. (1989). Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network. *Neural Networks*, 2, 459-73.
5. Meyer-Bäse, A., Jancke, K., Wismüller, A., Foo, S. & Martinetz, T. (2005). Medical image compression using topology-preserving neural networks. *Engineering Applications of Artificial Intelligence*, 18(4), 383-392.
6. Jianxun, M.I. & Huang, D. (2004). Image compression using principal component neural network.*Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision*. Kunming, China.
7. Hykin, S. (1999). Neural Network: *A Comprehensive Foundation* (2nd Ed.). New Jersey: Prentice-Hall Inc.
8. Belegundu, A. D. & Chandrupatla, T. R. (1999). *Optimization Concepts and Application in Engineering*. New Jersey: Prentice- Hall Inc.
9. Sadka, A. H. (2002). *Compressed Video Communication*. West Sussex: John Wiley& sons.
10. Nishantha, D., Hayashida, Y. & Hayashi, T. (2004). Application level rate adaptive motion-JPEG transmission for medical collaboration systems. *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops.*