

# Image Compression Using Neural Networks

Yahya M. Masalmah  
Advisor: Dr. Jorge Ortiz

Electrical and Computer Engineering Department  
University of Puerto Rico, Mayagüez Campus  
Mayagüez, Puerto Rico 00681-5000  
masalmah@larsip.uprm.edu

## Abstract

In this project, multilayer neural network will be employed to achieve image compression. The network parameters will be adjusted using different learning rules for comparison purposes. Mainly, the input pixels will be used as target values so that assigned mean square error can be obtained, and then the hidden layer output will be the compressed image. It was noticed that selection between learning algorithms is important as a result of big variations among them with respect to convergence time and accuracy of results.

## 1. Introduction

Since images can be regarded as two-dimensional signals with the spatial coordinated as independent variables, image compression has been an active area of research since the inception of digital image processing. It is extremely important for efficient storage and transmission of image data. Since it was an area of interest of many researchers, many techniques have been introduced. Artificial Neural network have found increasing applications in this field due to their noise suppression and learning capabilities.

A number of different neural network models, based on learning approach have been proposed. Models can be categorized as either linear or nonlinear neural nets according to their activation function.

In the literature different approaches of image compression have been developed, one of these algorithms [Watanabe01] have been developed on the basis of a modular structured neural network which consists of multiple neural networks with different block size (the number of input units) for the region segmentation. Multilayer neural network [Charif00] have been developed in [Wahhab97] as another image compression algorithm, in which the use of

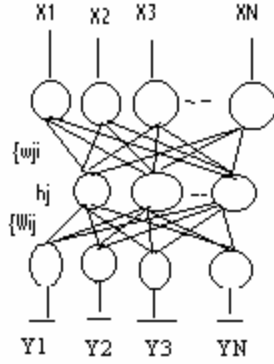
two-layer neural networks had been extended to multilayer network. Also in [Wahhab97] Auto associative transform coding approach which employs two-layer feed forward neural net to compress images have been compared to the new technique of four or more layers which in turn needs along time to be trained. As a continuation of algorithm developments, a new approach has been introduced in [Patnaik01] which uses auto-associative neural network and embedded zero-tree coding. In [Patnaik01] network training is achieved through recursive least square (RLS) algorithm.

A self organizing neural network has been used [Erickson92] in which vector quantization learning rule have been employed. As seen from the literature different coding methods have been addressed, in [Dony95] predictive coding, transform coding, and vector quantization have been utilized in multilayer perceptron training. In this project, feed forward back propagation algorithm will be employed to achieve image compression. A two-layer feed forward neural network will be used and different learning rules will be employed for training the network. Back propagation could be trained by different rules.

In this project, different learning rule will be employed to train multilayer neural network. The network will be constructed from input layer, hidden layer, and output layer. The image should be subdivided into sub blocks and the pixels gray level values within the block will be reshaped into a column vector and input to the neural network through the input layer. Input pixels will be used as the target values, so that the mean square error could be adjusted as needed. The network will be trained by back propagation, using different learning algorithms. Mainly one step secant, Newton's method, gradient descent and adaptive gradient descent learning algorithms will be used for this purpose.

## 2. Back-propagation Neural Network

The neural network structure can be illustrated in fig. 1. Three layers, one input layer, one output layer and one hidden layer, are assigned. Both of input layer and output layer are fully connected to hidden layer. Compression is achieved by designing the value of the number of neurons at the hidden layer, less than that of neurons at both input and output layers.



**Figure1** Back propagation neural network [Watanabe01]

The above neural network could be either linear or nonlinear network according to the transfer function employed in the layers. Log-sigmoid function which is given in equation1 is one of the most common functions employed in different neural networks problems.

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (1)$$

Some networks employs log-sigmoid in the hidden layer and linear in the output layer. It was shown that nonlinear functions have a more capability of learning both linear and nonlinear problems than linear ones.

The network shown in Fig.1 also have been used in [Charif00] as a feed forward neural network. The output  $Z_j$  of the  $j^{\text{th}}$  neuron in the hidden layer is given by

$$Z_j = f^1 \left( \sum_{i=1}^N w_{ji} X_i + b_j \right) \quad (2)$$

And the output  $Y_k$  of the  $k^{\text{th}}$  neuron in the output layer is given by

$$Y_k = f^2 \left( \sum_{l=1}^M w_{kl} Z_l + b_k \right) \quad (3)$$

Where  $f^1$ ,  $f^2$  are the activation function of the hidden layer and the output layer respectively,  $w_{ji}$  in (2) is the

synaptic weight connecting the  $i^{\text{th}}$  input node to the  $j^{\text{th}}$  neuron of the hidden layer,  $b_j$  is the bias of the  $j^{\text{th}}$  neuron in the hidden layer.  $N$  is the number of neurons in the hidden layer,  $f^1$  is the activation function, and  $Z_j$  is the output in the hidden layer. Similarly, (3) describes the subsequent layer where  $Y_k$  is the output of the  $k^{\text{th}}$  neuron in the output layer,  $M$  is number of neurons in the output layer.

## 3. Network Training

Training the network is an important step to get the optimal values of weights and biases after being initialized randomly or with a certain values. The network could be trained for different purposes like function approximation, pattern association, or pattern classification. The training processes require a set of prototypes and targets to learn the proper network behavior. During training, the weights and biases of the network are iteratively adjusted to minimize the network performance function which is the mean square error for the feed forward networks. The mean square error is calculated as the average squared error between the inputs and targets.

In the basic back propagation training algorithm, the weights are moved in the direction of the negative gradient, which is the direction in which the performance function decreases most rapidly. Iteration of this algorithm can be written as:

$$X_{k+1} = X_k - a_k g_k \quad (4)$$

Where  $X_{k+1}$  is a vector of current weights and biases,  $g_k$  is the current gradient, and  $a_k$  is the learning rate. Based on the same idea as before, different algorithms had been introduced. In [Bouzerdoum01] different learning methods have been introduced, Classic back propagation (BP), and conjugate gradient (back propagation (CGBP) have been addressed.

In this project, all training algorithms have been developed using MATLAB.

## 4. Procedure

As our purpose in this project is image compression, it is important to explain the steps which have been done. For our purpose three layers feed forward neural network had been used. Input layer, hidden layer with 16 neurons, and output layer with 64 neurons. Back propagation algorithms had been employed for the training processes. To do the training input prototypes and target values are necessary to be introduced to the network so that the

suitable behavior of the network could be learned. The idea behind supplying target values is that this will enable us to calculate the difference between the output and target values and then recognize the performance function which is the criteria of our training. For training the network, the 256x256 pixels Lena image had been employed.

#### 4.1 Pre-processing

For the original image to be used, it has to be divided into 8x8 pixel blocks, and then each block should be reshaped into a column vector of 64x1 elements. In my case I had 1024 blocks of which 64x1024 matrix have been formed with each column representing one block. Also for scaling purposes, each pixel value should be divided by 255 to obtain numbers between 0 and 1.

#### 4.2 Training algorithms

With the input matrix constructed in (4.1) and each column represents a prototype, and with the target matrix equal to the input matrix, the training could be started. Different algorithms have been employed; they can be classified into two main categories: slow and fast algorithms.

##### 4.2.1 Fast algorithms

From fast algorithms: quasi-Newton (trainbfg, trainoss), Conjugate gradient (traincfg, traincpg), and Levenberg-Marquardt (trainlm). The quasi-Newton algorithms were employed and it was found that trainbfg is very slow but on the other hand, it is good at memory management, whereas trainoss is fast and good at memory management. The Levenberg-Marquardt is very fast but it is very bad at memory management since it keeps some matrices from previous iterations.

##### 4.2.2 Slow algorithms

From slow algorithms: gradient descent (traingd, traingda). It is slow at speed and good at memory management.

#### 4.3 Simulation of Results

After training, the network had been simulated by the input matrix and the target matrix. In the simulation process two outputs obtained, the hidden layer output, and the output layer output. The hidden layer produces a matrix of 16x1024, whereas the output layer produces a matrix of 64x1024.

#### 4.4 Post-Processing

The next step is to display both matrices in (4.3) as images. This can be done by reshaping each column

into a block of the desired size and then arrange the blocks to form the image again. In the hidden layer each column should be reshaped into 4x4 pixel blocks, while in the output layer, each column should be reshaped into 8x8 pixel blocks as the input. In both cases each pixel value should be multiplied by 255 to obtain the original gray level value of the pixels.

#### 4.5 Algorithm

1. Divide the original image into 8x8 pixel blocks and reshape each one into 64x1 column vector.
2. Arrange the column vectors into a matrix of 64x1024.
3. Let the target matrix equal to the matrix in step 2.
4. Choose a suitable learning algorithm, and parameters to start training.
5. Simulate the network with the input matrix and the target matrix.
6. Obtain the output matrices of the hidden layer and the output layer.
7. Post-process them to obtain the compressed image, and the reconstructed image respectively.

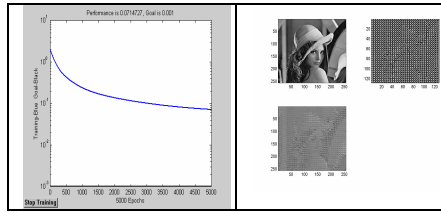
#### 5. Results

The algorithm had been tested with different learning algorithms and the results had been observed. Herein, all the results were displayed in pairs; the left part is the performance function which is the criteria of training, and the right part which includes the original image in the left top corner, the compressed image in the right top corner, and the reconstructed image in the left bottom corner. The entire three images in right were displayed for comparison purposes. Since our target is the original image, the accuracy of the compression will be how close the reconstructed image is from the original image.

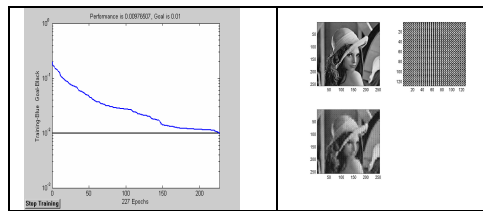
1. Using tansig function in the hidden layer and linear function in the output. This part was done to check the ability of learning of linear activation functions (see Figure 2.).
2. The trainbfg algorithm takes long time to converge, so the goal had been changed to 0.01 instead of 0.001 (see Figure 3.).

It can be easily shown that reconstructed image needs more iteration to converge. From Figure 3, the number of epochs needed to reach the goal was 227, while it took long time to get there. For better results, the goal could be .001 while the time needed will be more than 4 hours.

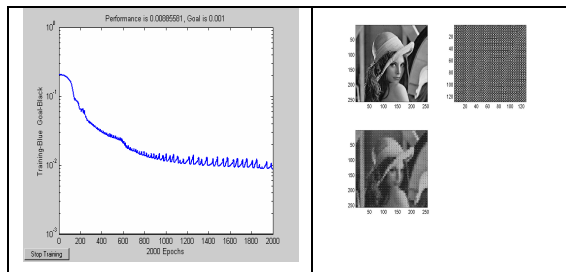
3. The traingda algorithm is an algorithm where the learning rate is variable; the results are shown in Figure 4.



**Figure 2** Performance function (left), original , compressed, reconstructed



**Figure 3** Trainbfg algorithm results



**Figure 4** Traingda algorithm results.

## 5. Conclusion

This paper has introduced a comparison among back propagation training algorithms in image compression. Gradient descent (traingd, traingda), Newton method (trainbfg, trainoss) were be tested while Levenberg-Marquardt (trainlm) was not completed; it was difficult as a result of memory management errors. The employed algorithms were tested for against parameters, like the number of epochs, and goal. With respect to time of execution, it is important to choose algorithms of acceptable time of execution. It was noted that there was a notable time variation between the used algorithms. It is important to choose the goal as small as possible to improve the image reconstruction.

Finally, Image compression could be achieved with high accuracy using feedforward back propagation.

Our judge of accuracy was the comparison of recovered image pixels value with the original image; and minimization of the mean square error between them. For future work the compression process could be done and the histogram of the recovered image and the original image could be compared.

## References

- [Watanabe01] Watanabe, Eiji, and Mori, Katsumi, "Lossy image compression using a modular structured neural Network," Proc. of IEEE signal processing society workshop, pp.403-412, and 2001.
- [Wahhab97] Abdel-Wahhab, O., and Fahmy, M. M., "Image compression using multilayer neural networks," IEEE proc. V is Signal Processing, vol. 144, No. 5, October 1997.
- [Erickson92] Erickson, D. S., and Thyagarajan, K. S., "A neural network approaches to image compression," Proc. of IEEE international symposium on circuits and systems, vol. 6, pp.2921-2924, 1992.
- [Patnaik01] Patnaik, Suprava and Pal, R. N., "Image compression using Auto-associative neural network and embedded zero-tree coding," "Third IEEE processing workshop advances in wireless communications, Taiwan, March 2001.
- [Charif00] Charif, H. Nait, and Salam, Fathi M., "Neural networks-based image compression system," Proc. 43<sup>rd</sup> IEEE Midwest symp. on circuits and systems, Lansing MI, August 2000.
- [Dony95] Dony, Robert D., and Haykin, Simon, "neural network approaches to image compression," Proc. of the IEEE, vol. 83, No. 2, February 1995.
- [Bouzerdoun01] Bouzerdoun, Abdessalam, "Image compression using a stochastic competitive learning algorithm (SCOLA)," International Symposium on signal processing and its applications (ISSPA), pp. 541-544, Kuala Lumpur, Malaysia, August 2001.