Open access Journal International Journal of Emerging Trends in Science and Technology

Back Propagation Neural Network Based Image Compression

Authors
Neelam[1], Ashu Bansal[2]
Hindu College of Engineering. Sonipat
Deen Bandhu Chotu Ram University Murthal,
Haryana-India

Abstract
Compression algorithms are methods that reduce the number of symbols used to represent source information, therefore reducing the amount of space needed to store the source information or the amount of time necessary to transmit it for a given channel capacity. This paper presents a neural network based technique and wavelet based compression. A three layered Back propagation Neural Network (BPNN) was designed for building image compression system. The Back propagation neural network algorithm (BP) was Used for training the designed BPNN.

Keywords: *Compression, Back Propagation Network, Neural Network*

1. Introduction

Image compression refers to the task of reducing the amount of data required to store or transmit an image. At the system input, the image is encoded into its compressed form by the image coder. The compressed image may then be subjected to further digital processing, such as error control coding, encryption or multiplexing with other data sources, before being used to modulate the analog signal that is actually transmitted through the channel or stored in a storage medium. At the system output, the image is processed step by step to undo each of the operations that were performed on it at the system input. At the final step, the image is decoded into its original uncompressed form by the image decoder. If the reconstructed image is identical to the original image the compression is said to be lossless, otherwise, it is lossy. Digital image processing: [1] improves the information content of the picture for better understanding of what the picture is made up of so that an image appears to be better as its contrast is increased; [2] provides data compression for efficient storage and transmission. Although, storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity. Recently, the need for the transmission of the visual data has increased, particularly over the Internet. Image compression is a commonly used practice in Joint Photographic Experts Group (JPEG) image compression standard. The JPEG image files generally have the jpeg in their name extensions. Several compression techniques have been developed, such as Differential Pulse Code Modulation, Discrete Cosine Transform, Discrete Fourier Transform, and numerous Vector Quantization (VQ) methods.

2. Overview of Backward Propagation Based Image Compression.

Back propagation training algorithm is a supervised learning Algorithm for multilayer feed forward neural network. Since it is a supervised learning algorithm, both input and target output vectors are provided for training the network. The error data at the output layer is calculated using network output and target output. Then the error is back propagated to intermediate layers, allowing incoming weights to these layers to be updated [3]. This algorithm is based on the error-correction learning rule.

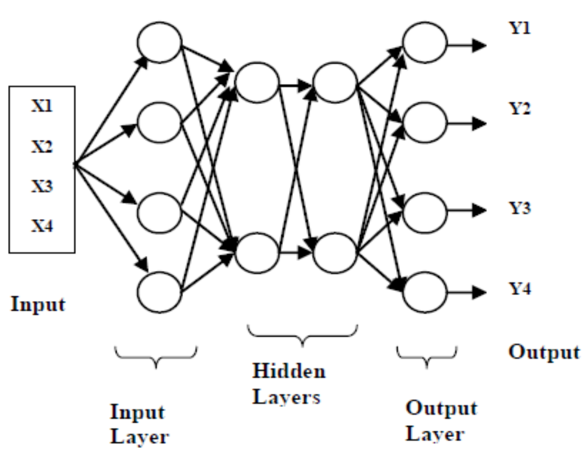

Figure 1: Neural Network

Basically, the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, input vector is applied to the network, and its effect propagates through the network, layer by layer [4]. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of network are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with the error-correction rule. The actual response of the network is subtracted from a desired target response to produce an error signal. This error signal is then propagated backward through the network, against direction of synaptic connections - hence the name "error back-propagation". The synaptic weights are adjusted so as to make the actual response of the network move closer the desired response [5]. If the artificial neural network has M layers and receives input of vector p, then the output of the network can be calculated using the following equation

$$a^m = f^m(W^m f^{m-1}(W^{m-1} f^{m-2}(\ldots . W^2 f^1(W^1 p + b^1) + b^2) + b^{m-1}) + b^m) \qquad (1)$$

For three layer network shown in fig. 1, the output a3 is given as:

$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3) \qquad (2)$$

Where f is the transfer function, and W^m & b^m are weight and bias of the layer m. For m=1, 2 …M. The neurons in the first layer receive external inputs:

$$a^0 = p \qquad (3)$$

The above equation provides the starting point of the network. The outputs of the neurons in the last layer are considered the network outputs.

$$a = a^M \qquad (4)$$

The performance index of the algorithm is the mean square error at the output layer. Since the algorithm is the supervised learning method, it is provided with set of examples of proper network behavior. This set is the training set for the neural network, for which the network will be trained as shown in equation below

$$\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_q, t_q\} \qquad (5)$$

Where, p_q is an input vector to the network, and t_q is the corresponding target output vector of the network. At each iteration, input is applied to the network and the output is compared with the target, to find the mean square error. The algorithm should adjust the network parameters i.e., weights and biases in order to minimize the Mean Square Error (MSE) given below

$$\text{MSE}=F = \frac{E[e^2]}{N} = \frac{E[(t-a)^2]}{N} = \frac{E[(t-a)^T(t-a)]}{N} \qquad (6)$$

The back propagation algorithm calculates how the error depends on the output, inputs, and weights. After which the weights $\Delta w(k)$ and $\Delta b(k)$ biases are updated. The change in weight and bias is given by the equation (7) and (8) respectively

$$\Delta w(k) = \alpha \frac{\partial F}{\partial w} \qquad (7)$$

$$\Delta b(k) = \alpha \frac{\partial F}{\partial b} \qquad (8)$$

Where, a is the learning rate at which the weights and biases of the network are adjusted at iteration k. The new weights and biases for each layer at k iteration are given in equations 9 and 10 respectively.

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\partial F}{\partial w_{i,j}^{m}} \quad (9)$$

$$b_{i}^{m}(k+1) = b_{i}^{m}(k) - \alpha \frac{\partial F}{\partial b_{i}^{m}} \quad (10)$$

So, we only need to find the derivative F of w.r.t w and b. This is the goal of the back propagation algorithm, since we need to achieve this backwards. First, we need to calculate how much the error depends on the output. By using the

Concept of chain rule

$$\frac{\partial F}{\partial w_{i,j}^{m}} = \frac{\partial F}{\partial n_{i}^{m}} * \frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} \quad (11)$$

$$\frac{\partial F}{\partial b_{i}^{m}} = \frac{\partial F}{\partial n_{i}^{m}} * \frac{\partial n_{i}^{m}}{\partial b_{i}^{m}} \quad (12)$$

Where, $n_{i}^{m} = \sum w_{i,j}^{m} a_{j}^{m-1} + b_{i}^{m}$ (13)

Therefore,

$$\frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} = a_{j}^{m-1} \quad , \quad \frac{\partial n_{i}^{m}}{\partial b_{i}^{m}} = 1 \quad (14)$$

Now if we define

$$s_{i}^{m} = \frac{\partial F}{\partial n_{i}^{m}} \quad (15)$$

In matrix form the updated weight matrix and bias matrix at k+1th pass can be written as:

$$W^{m}(k+1) = W^{m}(k) - \alpha s^{m} (a^{m-1})^{T} \quad (16)$$

$$b^{m}(k+1) = b^{m}(k) - \alpha s^{m} \quad (17)$$

Where s^{m}, the sensitivity matrix is

$$s^{m} = \frac{\partial F}{\partial n^{m}} = \begin{pmatrix} \frac{\partial F}{\partial n_{1}^{m}} \\ \frac{\partial F}{\partial n_{2}^{m}} \\ \vdots \\ \vdots \\ \frac{\partial F}{\partial n_{s}^{m}} \end{pmatrix} \quad (18)$$

The BPNN Image compression process can be summarized by the following steps [6]:

Step 1: Read image pixels from file and then normalize it by Converting it from range [0-255] to range [0-1].

Step 2: Divide the image into non-overlapping blocks.

Step 3: Rasterizing the image blocks.

Step 4: Apply the rasterized vector into input layer units

Step 5: Compute the outputs of hidden layer units by multiplying the input vector by the weight matrix (V).

Step 6: Store the outputs of hidden layer units after demoralizing them in a compressed file.

Step 7: While there are more image vectors go to

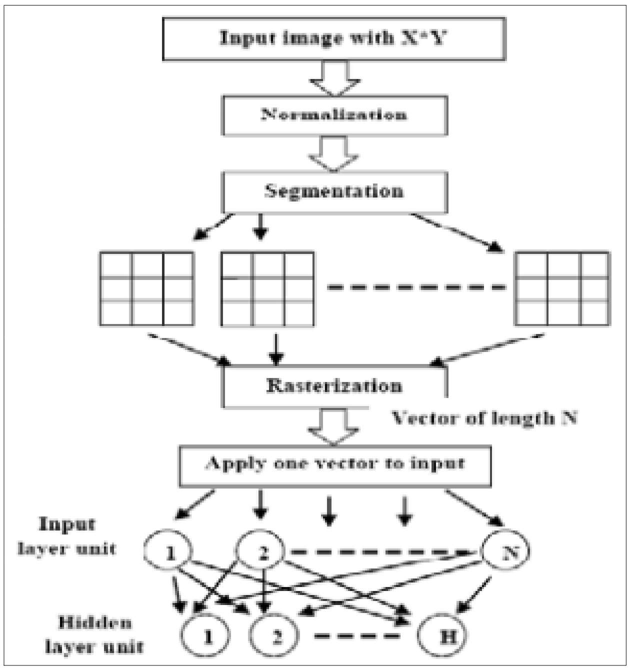

Figure 2: Flow Graph of Image Compression Algorithm

3. Overview of Levenberg Marquardt algorithm

1. An image, F, is divided into rxc blocks of pixels. Each block is then scanned to form a input vector x (n) of size p=rxc
2. It is assumed that the hidden layer of the layer network consists of L neurons each with P synapses, and it is characterized by an appropriately selected weight matrix Wh.
3. All N blocks of the original image is passed through the hidden layer to obtain the hidden signals, h(n), which represent encoded input

image blocks, x(n) If L<P such coding delivers image compression.

4. It is assumed that the output layer consists of m=p=rxc neurons, each with L synapses. Let Wy be an appropriately selected output weight matrix. All N hidden vector h(n), representing an encoded image H, are passed through the output layer to obtain the output signal, y(n). The output signals are reassembled into p=rxc image blocks to obtain a reconstructed image,
5. Training is conducted for a representative class of images using the Levenberg Marquardt algorithm.
6. Once the weight matrices have been appropriately selected, any image can be quickly encoded using the Wh matrix, and then decoded (reconstructed) using the Wy matrix.

4. Result Analysis

In this paper, we have done the compression using both back Propagation and their results are as under. Result of Back Propagation is shown in figure 3.

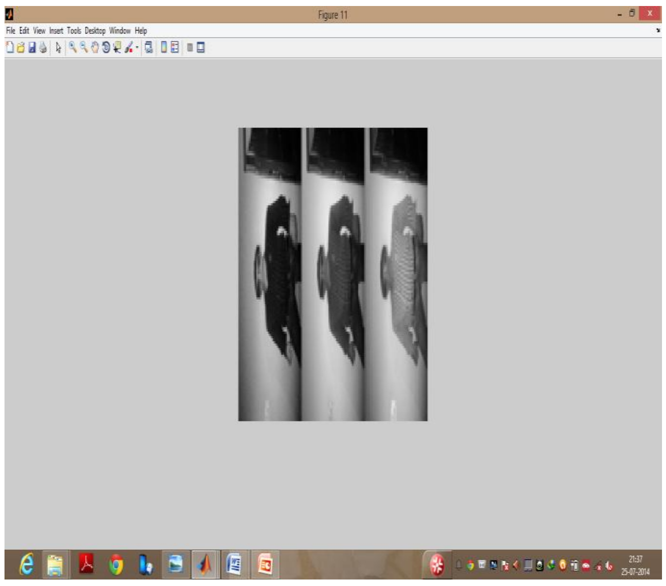

Figure 3: Compression through Back Propagation

5. Conclusion

In this paper, the performance of the designed BPNN image compression system can be increased by modifying the network itself, learning parameters and weights. Practically, we can note that the BPNN has the ability to compress untrained images but not in the same performance of the trained images.

References

1. Aslam Khan, Sanjay Mishra,"Image Compression using Growing Self Organizing Map", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 2, pp. 404-408, Feb. 2013.
2. Juha Vesanto, Johan Himberg, Esa Alhoniemi, Juha Parhankangas,"Self-organizing map in Matlab: the SOM Toolbox", Proceeding of the MatLab DSP Conference pp. 35-39, Nov. 1999.
3. Martin T. Hagan, Howard B. Demuth, Mark Beale, "Neural Network Design", China Machine Press, 2002.
4. Himavathi, Anitha, Muthuramalingam,"Feed forward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization", Neural Networks, IEEE Transactions-2007
5. Rafael Gadea, Franciso Ballester, Antonio Mocholí, Joaquín Cerdá,"Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation", ISSS Proceedings of the 13th international symposium on System synthesis, IEEE Computer Society 2000.
6. S. Areepongsa, N. Kaewkamnerd, Y. F. Syed, K. R. Rao, "Wavelet Based Compression for Image Retrieval Systems".