Image Compression using Multilayer Feed Forward Artificial Neural Network and DCT

Fatima B. Ibrahim

Information and communication engineering, Baghdad University, Baghdad, Iraq

Abstract: The Neural Networks are good alternative for solving many complex problems. In this paper multi-layer neural network has been employed to achieve image compression. The proposed technique breaks down large images into smaller windows and applies Discrete Cosine Transform (DCT) to these windows as a pre-process technique. 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. A compression among different back propagation training algorithms were introduced with different compression ratio ,and different block sizes were expressed. Block sizes play role in image compression even with the same compression ratio. The proposed technique has been implemented using Matlab®.

Key words: Image Compression, Neural networks, Back propagation Algorithm, Gradient descent algorithms, Quasi Newton algorithm, Discrete Cosine Transform.

INTRODUCTION

The study of image compression methods has been an active area of research since the inception of digital image processing. Since images can be regarded as two-dimensional signals with the independent variables being the coordinates of a two-dimensional space, many digital compression techniques for one-dimensional signals can be extended to images with relative ease. As a result, a number of approaches to the problem are well established. Most current approaches fall into one of three major categories: predictive coding, transform coding, or vector quantization. Alternatively, a combination of these techniques may be applied in a hybrid approach[1].

Artificial Neural Networks have been applied to image compression problems, due to their superiority over traditional methods when dealing with noisy or incomplete data. Artificial Neural networks seem to be well suited to image compression, as they have the ability to preprocess input patterns to produce simpler patterns with fewer components. This compressed information preserves the full information obtained from the external environment, not only can Artificial Neural Networks based techniques provide sufficient compression rates of the data in question, but also security is easily maintained. This occurs because the compressed data that is sent along a communication line is encoded and does not resemble its original form. Many different training algorithms and architectures have been used. Different types of Artificial Neural Networks have been trained to perform Image Compression. Feed-Forward Neural Networks, Self-Organizing Feature Maps, Learning Vector Quantizer Network, have been applied to Image Compression. These networks contain at least one hidden layer, with fewer units than the input and output layers. The Neural Network is then trained to recreate the input data. Its bottleneck architecture forces the network to project the original data onto a lower dimensional manifold from which the original data should be predicted[2].

2. Image Compression: Following the rapid development of information and communication technologies, more and more information has to be processed, stored, and transmitted in high speed over networks. The need for data compression and transmission is increasingly becoming a significant topic in all areas of computing and communications. Computing techniques that would considerably reduce the image size that occupies less space and bandwidth for transmission over networks form an active research. Image compression deals with reducing the amount of data required to represent a digital image[3].

Image compression is concerned with efficiently encoding images for storage and transmission. Lossless image compression encodes the data exactly, the decoded image being identical to the original. Lossy image compression encodes an approximation of the original image in order to reduce the encoded length. Lossy algorithms must trade-off the level of...
compression and the amount of distortion in the reconstructed image\(^4\).

3. Neural Network: Artificial Neural Networks have been applied to many problems, and have demonstrated their superiority over classical methods when dealing with noisy or incomplete data. One such application is for data compression. Neural networks seem to be well suited to this particular function, as they have an ability to preprocess input patterns to produce simpler patterns with fewer components\(^5\). Neural networks are computer algorithms inspired by the way information is processed in the nervous system. An important difference between neural networks and other AI techniques is their ability to learn. The network “learns” by adjusting the interconnections (called weights) between layers. When the network is adequately trained, it is able to generalize relevant output for a set of input data. A valuable property of neural networks is that of generalization, whereby a trained neural network is able to provide a correct matching in the form of output data for a set of previously unseen input data. Learning typically occurs by example through training, where the training algorithm iteratively adjusts the connection weights (synapses). Back-propagation (BP) is one of the most famous training algorithms for multilayer perceptrons\(^6\). 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\(^7\).

3.1 Back-propagation Algorithm: In the field of image compression, one of the most exciting and potentially profitable recent developments is the increasing use of artificial intelligence techniques like neural networks in clustering, encoding and decoding. Artificial neural networks have been applied to many problems, and have demonstrated their superiority over classical methods when dealing with noisy or incomplete data. Neural networks are well suited to this method, as they have the ability to pre-process input patterns to produce simpler patterns with fewer components. A fascinating feature of the brain is that its physical organization reflects the organization of the external stimuli that are presented to it. In view of this back propagation algorithm has been used to classify the domain cells. In this back propagation algorithm the weights from input layer-hidden layer-output layer are updated iteratively during the learning phase\(^8\).

Back-propagation algorithm is a widely used learning algorithm in Artificial Neural Networks. The Feed-Forward Neural Network architecture (Fig. 1) is capable of approximating most problems with high accuracy and generalization ability. This algorithm is based on the error correction learning rule. Error propagation consists of two passes through the different layers of the network, a forward pass and a backward pass\(^9\).

In the backward pass, the updating of weights in back-propagation algorithm is done as follows:

The error signal at the output of neuron \(j\) at iteration \(n\) is given by

\[
e_j(n) = d_j(n) - y_j(n)
\]

The instantaneous value of error for neuron \(j\)

\[
\frac{1}{2} e_j^2(n).\text{ This instantaneous value } e(n) \text{ of total error is obtained by summing } \frac{1}{2} e_j^2(n) \text{ of all neurons in output layer}
\]

\[
e(n) = \frac{1}{2} \sum_{j \in e} e_j^2(n)
\]

where \(e\) includes all neurons in the output layer. Average squared error is given by

\[
E_{\text{avg}} = \frac{1}{N} \sum_{n=1}^{N} e(n)
\]

where \(N\) is total number of patterns in training set. So minimization of \(E_{\text{avg}}\) is required. Back propagation algorithm is used to update the weights. Induced local field \(v_j(n)\) produced at input of activation function is given by

\[
v_j(n) = \sum_{i=0}^{m} w_{ji}(n)X_i(n)
\]

where \(m\) is the number of inputs applied to neuron \(j\). So the output can be written as

\[
y_j(n) = \phi_j(v_j(n))
\]
The back propagation algorithm applies a correction \( \Delta w_{ji}(n) \) to synaptic weights \( w_{ji}(n) \)
which is proportional to partial derivative
which can be written as
\[
\frac{\partial E(n)}{\partial w_{ji}(n)} \frac{\partial e_j(n)}{\partial w_{ji}(n)}
\]
Differentiating the Eq. (2) with respect to \( e_j(n) \)
\[
\frac{\partial E(n)}{\partial e_j(n)} = e_j(n)
\]
Differentiating Eq. (1) with respect to
\[
\frac{\partial e_j(n)}{\partial y_j(n)} = -1
\]
Differentiating Eq. (5) we get
\[
\frac{\partial y_j(n)}{\partial v_j(n)} = \phi_j^k(v_j(n))
\]
The discrete cosine transform of an \( N \times N \) image, \( f(x, y) \) is defined by:
\[
F(u, v) = C(u)C(v) \sum_{x=0}^{(N-1)} \sum_{y=0}^{(N-1)} f(x, y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}
\]
The inverse transform is defined by:
\[
f(x, y) = \sum_{u=0}^{(N-1)} \sum_{v=0}^{(N-1)} C(u)C(v)F(u, v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}
\]
4. The Proposed Neural Network Method:
4.1 DCT-based Pre-process Method: The DCT has been used in many practical applications, especially in signal compression. The DCT decomposes a signal into its elementary frequency components. When applied to an \( M \times N \) image/matrix, the 2D-DCT compresses all the energy/information of the image and concentrates it in a few coefficients located in the upper-left corner of the resulting real-valued \( M \times N \) DCT/frequency matrix.
4.2 Procedure: In this paper multilayer feed forward neural network architecture is used. It is trained with different learning rules. As our purpose in this paper is image compression, it is important to explain the steps which have been done. Many steps must be taken before an image can be successfully compressed using the neural network. The steps proposed for image compression are as follows:
1. Divide the original image into blocks (nxm).
2. For scaling purposes, each pixel value should be normalized between 0 and 1 (i.e. divided by 255 to obtain numbers between 0 and 1).
3. Chose to use either traditional method or modified by preproccessing using DCT:
4. Leave the sub-image as it is.
5. Apply the 2 dimensional Discrete Cosine Transform (DCT) for each sub-image.
6. Rearrange the blocks into column vectors.
7. Arrange the column vectors into a matrix to be the input matrix.
8. Let the target matrix be equal to the input matrix.
9. Train the network with suitable learning algorithm and parameters.
10. Save the adjusted weights.
11. Simulate the network to obtain the reconstructed image.

### 4.3 Architecture:

The multi-layer back propagation-learning network will train each sub-image. The number of neurons in the middle hidden layer will be designed for the desired compression. The number of neurons in the output layer will be the same as that in the input layer. The input layer and output layer are fully connected to the hidden layer.

As shown in figure (2) and figure (3) the input to hidden layers represent the compress (encode) part of the work, while the hidden to output layers represent the decompress (decode) part. The input image is split up into a number of blocks, each block has N pixels, which is equal to the number of input neurons.

The Mean Square error of the difference between the network output and the desired output is calculated. This error is back propagated and the weight synapses of output and input neurons are adjusted. With the updated weights error is calculated again. Iterations are carried out till the error is less than the tolerance. The compression performance is assessed in terms of Peak Signal to Noise Ratio (PSNR).

The network used for image compression is breaking into two parts as shown in Figure (3). The transmitter encodes and then transmits the output of hidden layer (only K values as compared to N of the original image). The receiver receives and decodes the K hidden outputs and generates the N outputs. Since the network is implementing an identity map, the reconstruction of the original image is achieved.

For the original image to be used, it has to be divided into blocks, and then each block should be reshaped into a column vector. And they are gathered into one matrix. The blocks are overlapped to get more training sets to the network. Also for scaling purposes, each pixel value should be normalized between 0 and 1, i.e. divided by 255 to obtain numbers between 0 and 1.

With the input matrix constructed 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.

The next step is to display the output matrix as an image. 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. Each pixel value should be multiplied by 255 to obtain the original gray level value of the pixels.

### 5. Simulation and Results:

In this paper, all training algorithms have been developed using Matlab®. The performance of the Back-Propagation Neural Network for image compression has been tested in various block sizes of images and different compression ratio. A gray level image of size 512 × 512 has been considered for training the network using back-propagation algorithm and using different training algorithms. The results were taken using the traditional technique as well as the DCT-based pre-processing technique. The feed forward back-propagation neural network is trained using two main different training algorithms, the Quasi-Newton Back-propagation algorithm (trianbfg), and the Gradient Descent algorithms in different two ways (Gradient descent with adaptive learning rate back-propagation traineda, Gradient descent with momentum and adaptive learning rate back-propagation trainedx).

Quasi-Newton Back-propagation Algorithm shows a very slow offline training converge against the Gradient Descent algorithms, but it gives better results for the reconstructed image in both the calculated PSNR standard and human vision. The reconstructed image looks so smooth. For the reconstructed image obtained from the Quasi-Newton Back-propagation Algorithm suggestion to post-process it by some filters to give it a smooth vision.

Different structural architecture were used to obtained the same compression ratio (three layers architecture, five layers architecture, and seven layers architecture) all give the same compression ratio. The performance must be better as much as layers increased because the data stored in the hidden layers are extra, but the results show different reverse fact, that 3 layers architecture is better than the others.

First, the image is sub-divided into (8x8) pixel blocks, normalized between 0 and 1; they are reshaped into (64x1) columns vectors. These vectors arranged into one matrix to be the training matrix. The target matrix will be equal to the input matrix. This is for the traditional method with the DCT-based pre-process method the same was done but each block is applied to the DCT transform function and at the end for the
Fig. 1: Feed Forward Neural Network Architecture

Fig. 2: Multilayer architecture for high order data compression-decompressions

Fig. 3: compression/decompression pair neural network.

Table 1: Experimental PSNR results with different methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Traditional</th>
<th></th>
<th>DCT-based</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>traingda</td>
<td>traingdx</td>
<td>traingda</td>
<td>traingdx</td>
</tr>
<tr>
<td>3 layers</td>
<td>25.7421</td>
<td>27.0381</td>
<td>27.2483</td>
<td>29.0327</td>
</tr>
<tr>
<td>5 layers</td>
<td>24.6434</td>
<td>25.1194</td>
<td>25.5395</td>
<td>23.8873</td>
</tr>
<tr>
<td>7 layers</td>
<td>24.5949</td>
<td>25.1808</td>
<td>22.6045</td>
<td>25.8106</td>
</tr>
</tbody>
</table>
deconstruction apply the results to the inverse DCT. These information tested over the three different structural architecture with compression ratio 4:1.

With the 3 layers architecture using in the traditional method using Gradient descent with momentum and adaptive learning rate back-propagation the PSNR obtained is 27.0381 at 3000 fast training epochs. At 3000 fast training epochs using Gradient descent with momentum and adaptive learning rate back-propagation with the DCT-based pre-process method the PSNR obtained is 29.0327. With DCT method not just the PSNR improved but also the human eye vision looks better. The results obtained with Gradient Descent algorithms using both and algorithms with both traditional method and DCT-based pre-process method are shown in figure(4), figure(5) and table(1).

By using the Quasi-Newton Back-propagation Algorithm, in traditional method the PSNR obtained is 35.1378 after training the network for 757 epochs for about 4 hours. Figure (6) shows the reconstructed image. While for this algorithm using the DCT-based pre-process method the PSNR obtained is 35.6917 after training the network for 780 epochs for about 4 hours. Figure (7) shows the reconstructed image.

The above options were tested and also retrained against different images and different blocks sizes, as well as compression ratio and training parameters. All the results tend to be in the same direction with slight difference on the PSNR values depends on the type of the image and the distribution of its gray scales.

6. Conclusion: This paper has presented a DCT-based pre-process method. This method shows a good performance on the reconstructed image in both the PSNR standard and Human Vision over the traditional method. Different training algorithms used show that Quasi Newton Back-propagation Algorithm is better in performance in spite of its slowness in time of convergence. On the other hand Gradient Descent
Algorithms considered fast training algorithms but their performances are acceptable. Number of hidden layers means the amount of data stored in them so as much as it will be large the looseness must be less, but the investigations on this paper show the reverse fact. The block size is important even with the same compression ratio; it must be large enough to hold enough training data and small enough for the size of the neural network.

REFERENCES