A Neural Network based Technique for Data Compression

B. Verma, M. Blumenstein and S. Kulkarni
School of Information Technology
Faculty of Engineering & Applied Science
Griffith University, Gold Coast Campus, QLD 4217, Australia
{ brijesh, michael, sidhi }@eassun.eas.gu.edu.au

Abstract: This paper presents a neural network based technique that may be applied to data compression. The proposed technique breaks down large images into smaller windows and eliminates redundant information. Finally, the technique uses a neural network trained by direct solution methods. Conventional techniques such as Huffman coding and the Shannon Fano method are discussed as well as more recent methods for the compression of data and images. Intelligent methods for data compression are reviewed including the use of Backpropagation and Kohonen neural networks. The proposed technique has been implemented in C on the SP2 and tested on digital mammograms and other images. The results obtained are presented in this paper.

Keywords: Neural Network, Data Compression, Image Compression, Digital Mammography

1. Introduction

The transport of data across communication paths is an expensive process. Data compression provides an option for reducing the number of characters or bits in transmission. It has become increasingly important to most computer networks, as the volume of data traffic has begun to exceed their capacity for transmission. Typical algorithms that have been developed are: Huffman coding, Arithmetic coding, Shannon-Fano method, Statistical modeling and their variations [4].

Artificial Neural Network (ANN) based techniques provide other means for the compression of data at the transmitting side and decompression at the receiving side [6]. The security of the data can be obtained along the communication path as it is not in its original form on the communication line. The purpose of this paper is to present the different techniques that may be employed for data compression and propose a new technique for better and faster compression.

The remainder of the paper is divided into 5 major sections. In Section 2, we revise some very commonly used conventional methods for data compression, Section 3, discusses some alternate intelligent methods. Section 4 gives an overview of the experimental method used for our research, results are presented in Section 5, and conclusions are drawn in Section 6.

2. Conventional Methods for Data Compression

2.1 Huffman Coding

Huffman coding [4] is a widely used compression method. With this code, the most commonly used characters contain the fewest bits and the less commonly used characters contain the most bits. It creates variable-length codes that contain an integral number of bits. Huffman codes have the unique prefix attribute, which means they can be correctly decoded despite being of variable length. A binary tree is used to store the codes. It is built from the bottom up, starting with the leaves of the tree and working progressively closer to the root. The procedure for building the tree is quite simple. The individual symbols are laid out as a string of leaf nodes that are going to be connected to the binary tree. Each node has a weight, which is simply the probability of the symbol's appearance. The tree is then built by the following steps:

1. The two tree nodes with the lowest weights are located.
2. A parent node for these two nodes is created. It is assigned a weight equal to the sum of the two child nodes.
3. The parent node is added to the list of free nodes, and the two child nodes are removed from the list.
4. One of the child nodes is designated as the path taken from the parent node when decoding a 0 bit. The other is arbitrarily set to the 1 bit.

5. The previous steps are repeated until only one free node is left. This free node is designated the root of the tree.

2.2. Shannon-Fano Method

The Shannon-Fano [4] tree is built according to a specific algorithm designed to define an effective code table. The actual algorithm is as follows:

1. For a given list of the symbols, develop a corresponding list of the probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.
2. Sort the list of the symbols according to the frequency, with the most frequently used symbols at the top and the least common at the bottom.
3. Divide the list into two parts, with the total frequency counts of the upper half being as close to the total of the bottom half as possible
4. The upper half of the list is assigned the binary digit 0, and the lower half is assigned the digit 1. This means that the codes for the symbols in the first half will all start with 0, and the codes in the second half will all start with 1.
5. Recessively apply the same procedure to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the node.

2.3. Arithmetic Coding

Arithmetic coding [4] bypasses the idea of replacing input symbols with a single floating-point output number. More bits are needed in the output number for longer, complex messages. This concept has been known for some time, but only recently were practical methods found to implement arithmetic coding on computers with fixed sized-registers. The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0. The single number can be uniquely decoded to create the exact stream of symbols that went into construction.

Arithmetic coding seems more complicated than Huffman coding, but the size of the program required to implement it, is not significantly different. Runtime performance is significantly slower than Huffman coding. If performance, and squeezing the last bit out of the coder is important, arithmetic coding will always provide as good or better performance than Huffman coding. But careful optimisation is needed to get performance up to acceptable levels.

3. Intelligent Methods for Data Compression

Artificial Neural Networks have been applied to many problems [3], 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 [1]. This compressed information (stored in a hidden layer) preserves the full information obtained from the external environment. The compressed features may then exit the network into the external environment in their original uncompressed form. The main algorithms that shall be discussed in ensuing sections are the Backpropagation algorithm and the Kohonen self-organising maps.

3.1 Backpropagation Neural Network

The Backpropagation (BP) algorithm has been one of the most successful neural network algorithms applied to the problem of data compression [7],[8]. The data compression problem in the case of the BP algorithm is posed as an encoder problem. The data or image to be compressed passes through the input layer of the network, and then subsequently through a very small number of hidden neurons. It is in the hidden layer that the compressed features of the image are stored, therefore the smaller the number of hidden neurons, the higher the compression ratio. The output layer subsequently outputs the decompressed image to the external environment. It is expected that the input and output data are the same or very close.

If the image to be compressed is very large, this may sometimes cause difficulty in training, as the input to the network becomes very large. Therefore in the case of large images, they may be broken down into smaller, sub-images [9],[12]. Each sub-image may then be used to train an individual ANN. Experiments have been conducted that have successfully compressed and decompressed images with impressive compression ratios, and little or no loss of data. Experiments have also been conducted to show the performance of a network when trained with a particular image, and then tested with a larger image. It was found that the generalisation capability of the backpropagation ANN could cope sufficiently when the difficulty of the problem was substantially increased [8].

Another algorithm similar to Dynamic Node creation developed by Ash [7], could be considered when choosing an appropriate number of units in the hidden layer. It starts with a network that contains only a single hidden unit in the hidden layer. The network is trained and if the weights obtained after training do not give a response with a desired accuracy then one more hidden unit is added and the network is again retrained. This process is repeated until a final network with the desired accuracy has been obtained.

3.2 Kohonen Self -Organising Maps

Another neural-type algorithm that has also been used for data and image compression is the Kohonen network [11]. It has been found that the Kohonen network can give better results for data compression than the BP neural network. The Kohonen network uses an unsupervised training algorithm. There is no feedback in the Kohonen network, and input vectors are organised into categories depending on their similarity to each other.

For data compression, the image or data is broken down into smaller vectors for use as input. For each input vector presented, the Euclidean distance to all the output nodes are computed. The weights of the node with the minimum distance, along with its neighbouring nodes are adjusted. This ensures that the output of these nodes are slightly enhanced. This process is repeated until some criterion for termination is reached. After a sufficient number of input vectors have been presented, each output node becomes sensitive to a group of similar input vectors, and can therefore be used to represent characteristics of the input data. This means that for a very large number of input vectors passed into the network, (uncompressed image or data), the compressed form will be the data exiting from the output nodes of the network (considerably smaller number). This compressed data may then be further decompressed by another network.

3.3 Hybrid BP and Kohonen Network

When the Kohonen and Backpropagation networks were compared, it was found that a better signal to noise ratio was achieved for compression when using a Kohonen network[11]. In fact it was found that the signal to noise ratio was quite poor when using a BP neural network [11]. However, it was observed that training time for a Kohonen network could become quite timely, and a method was found to combine the two, to increase training time.

The BP algorithm was first used to compute initial weights for the Kohonen network. It was found that by using random weights for the Kohonen network, training time was too long. Therefore, the combination of both networks reduced training time and maintained a satisfactory signal to noise ratio.

3.4 Image Indexing

At present, the research in image indexing and compression is running in separate directions, ie image indexing normally assumes that the images stored are not compressed and compression algorithms are developed. Jiang [1], proposes a neural network (similar to those already discussed) which contains an input layer, output layer and a hidden layer of neurons. At the input layer, pixel blocks of the input image are presented. The hidden layer plays major roles in compressing and processing the input images. Basically all the neurons in this layer act as a codeword in the finalised code-book and attend the competition governed by learning rules to obtain the optimised codeword. The outcome of the competition and learning is then forwarded to the output layer where a number of operations are completed:

1. Best possible code-words are selected to represent blocks of image pixels.
2. Indices or labels of codeword are stored in data base for the reconstruction of input images.
3. A histogram is constructed for each individual image for its future retrieval in visual database.

4. Proposed techniques for image compression

Many steps must be taken before an image can be successfully compressed using either conventional or intelligent methods. The steps proposed for image compression are as follows: 1. Binarisation, 2. segmentation of the image, 3. Preparation of training pairs, 4. Exclusion of similar training pairs, 5. Compressing the image using the Direct Solution Method (DSM), 6. Reconstruction of the image.

4.1 Binarisation

The image is first converted into a monochrome bitmap. Each pixel of the image is then converted into a “1” or “0”. The black pixels of the image are converted into “1s” and the white pixels are converted to “0s”. Binarisation is very important, for further preprocessing of the image.

4.2 Segmentation of image

The image is then segmented into smaller images or windows. This step is very important so as to limit the number of inputs into the ANN. This step is also important to later exclude redundant training pairs.

4.3 Preparation of training pairs

After the larger image is broken down into smaller more useable windows, it is necessary to alter them into a form ready for use with the ANN. A file is prepared where each window is written as two identical vectors to form a training pair. In other words, the first vector would be the input vector and the second vector would be the desired output.

4.4 Exclusion of similar training pairs

As many of the images tested were extremely large, there were many similar training pairs after segmentation. To eliminate this redundant information, a program was used to search the training file for similar training pairs and delete them. This not only reduced the number of redundant training pairs, but reduced training time of the ANN.

4.5 Compression of the image using an ANN and the Direct Solution Method

At this stage the training data was ready for input into the ANN. Instead of using a time-consuming iterative method for training the network, the problem was solved by a DSM [13]. The image was “compressed” in the hidden units of the ANN. The weights (unknowns) are determined by solving a system of equations, which may then be used for “decompression” of the image.

4.6 Reconstruction of the image

Finally, the output produced by the ANN in the form of decompressed vectors or windows, is reconstructed to produce the full original image. The binarised pixels are converted into their proper graphical representation, and are compared with the original image.

5. Experimental Method

5.1. Image Acquisition

The images were scanned using a Macintosh Flatbed scanner and a Macintosh Personal Computer running the OFOTO software package. The images were saved in Tagged Image Format (TIF). They were then easily moved across PC platforms to an IBM compatible machine running Windows 95. The Paint Shop Pro package was used to convert the TIF images into Bitmap (BMP) type images. It was necessary to export the images yet again to another software package: Paintbrush. The image was then converted to a solely black and white (monochrome) format. This was necessary for binarisation to be completed in further steps.

5.2 Preprocessing

Binarisation was one of the first techniques employed. Binarisation is a technique that converts the black and white pixels of the BMP-type image into “1’s” and “0’s”, respectively. In this form, segmentation and other preprocessing techniques can be executed much more easily. Binarisation was performed using an already implemented program written in ANSI C. Segmentation has already been employed to develop a preliminary database of sample images. A simple segmentation program was implemented in C to segment larger images into smaller blocks ready for use in conjunction with a classification method.

5.3 Neural Network Algorithm

The choice of the neural network algorithm was of great importance to obtain accurate classification rates for the images acquired. A neural network using the Direct Solution Method was chosen due to its ability to train a neural network faster than existing ANN algorithms.

5.4 Implementation Platform and Language

The segmentation, preprocessing and classification programs were implemented and run using the SP2 Supercomputer. The operating system was UNIX and the chosen language for implementation was C.

5.5 Experimental Results

The data compression experiments were conducted using various images, varying in size. Most of the images were quite large, therefore as outlined in Section 4.2 the images were broken down into smaller windows. Window sizes between 8x8 and 16x16 pixels in dimension were chosen to limit the inputs into the network. The images and the settings for the ANN using the DSM are listed in Table 1.

Experiment 1 : Flower
For our first experiment we used a small simple bitmap image to test our data compression system. The "flower" image was very small (32x32) it was divided into smaller windows with dimensions : 16x16. The original and reconstructed images are shown below in Figures 1(a) and (b) respectively:

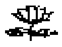

Figure 1(a). Original Image

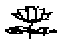

Figure 1(b) Reconstructed Image

Experiment 2: Lena
For our second experiment we used the benchmark image: "Lena" to test our system. The original and reconstructed images are shown below in Figure 2(a) and (b) respectively:

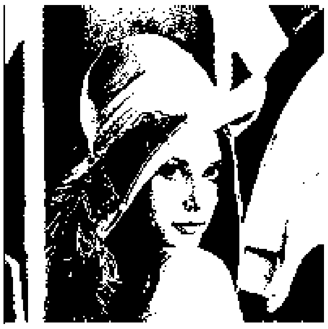

Figure 2(a). Original Image

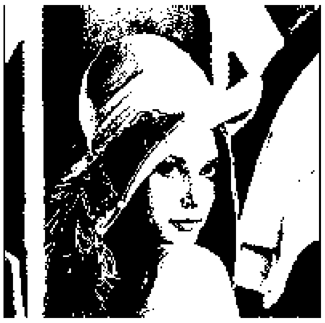

Figure 2(b). Reconstructed Image

Experiment 3 : Digital Mammograms
For our final experiment, we downloaded a large digital mammogram (largest of all three images) and we employed our data compression and decompression techniques. Below are the original and reconstructed images, Figures 3(a) and(b):

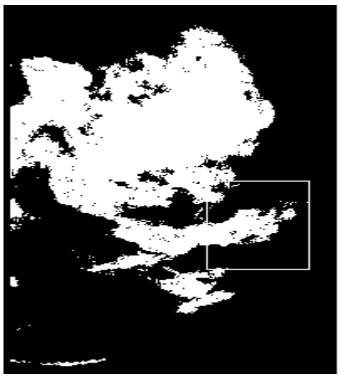

Figure 3(a). Original Image

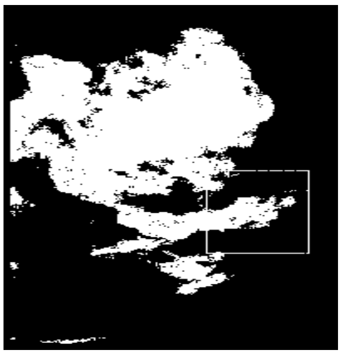

Figure 3(b). Reconstructed Image

Table 1. ANN Parameters

Image	Dimensions of Image	Dimensions of Window	No. of Hidden Units
Flower	32x32	16x16	4
Lena	200x200	8x8	60
Mammogram	336x416	16x16	100

6. Conclusion

In this paper, we have presented a new ANN based technique and we have also reviewed many traditional and intelligent techniques for data compression. Our approach using a fast training algorithm (DSM), has been implemented in the C programming language on the SP2 Supercomputer.
We segmented, compressed, decompressed and reconstructed three images using our proposed method. Results showed that a superior training time and compression could be achieved with our method. Further research is currently ongoing, and more experiments will be presented in the final paper.

References

[1] Jiang, J. A Neural Network Design for Image Compression and Indexing. International Conference on Artificial Intelligent Expert Systems and Neural Networks, Hawaii, USA, 1996, 296-299.

[2] Joutsensalo, J. Non Linear Data Compression and Representation by Combining Self Organising Map and Subspace Rule, Vol-2 IEEE International Conference on Neural Networks, 1994, 637-642.

[3] Blumenstein, M The Recognition of Printed and Handwritten Postal Address using Artificial Neural Networks. Dissertation, Griffith University, Australia, 1996

[4] Nelson M, The Data Compression Book M & T Publishing Inc. 1991

[5] Gelene R., and Sungar, M. Random Network Learning and Image Compression using Artificial Neural Networks, Vol-6 IEEE 1994, 3996-4001.

[6] Sicuranza, G.L, Ramponi G, and Marsi S. Artificial Neural Network for image compression, Electronic letters, 1990, 477-479.

[7] Ash, T. (1989) Dynamic Node Creation in Backpropagation Networks, 1989, 365-375.

[8] Kenue S. K. Modified Back-Propagation Neural Network with Applications to Image Compression, SPIE Vol.1709, Applications of Artificial Neural Networks, 1992, 394-401.

[9] Carbonara, M., Fowler J., and Ahalt, S, Compression of Digital Video Data using Artificial Neural Network Differential Vector Quantization, SPIE Vol. 1709, Applications of Artificial Neural Networks, 1992, 422-433.

[10] Min, K., and Min, H. Neural Network Based Image Compression using AMT DAP 610, SPIE Vol. 1709, Application of Artificial Neural Networks, 1992, 386-393.

[11] Panchanathan, S., Yeap, T., and Pilache, B. A Neural Network for Image Compression, SPIE Vol. 1709, Application of Artificial Neural Networks, 1992 376-385.

[12] Hambaba, M., Coffey B., and Khemlani, N. Image Coding using a Knowledge based Recognition System, SPIE Vol. 1709, 1992

[13] Verma, B., Fast Training of Multilayer Perceptrons, IEEE Trans. on Neural Networks, 1997