ISSN 2320-9984

Volume 2 Issue 2 | June 2014

ISSN: 2320-9984 (Online)

International Journal of Modern Engineering & Management Research

Website: www.ijmemr.org

Image Compression Technique

Shabbir Ahmad
Department of Computer Science and Engineering
Bhagwant University,
Ajmer (Rajasthan) [INDIA]
Email: ahmadbhopal281@gmail.com

M.R. Aloney
Assistant Professor
Department of Computer Science and Engineering
Bhagwant University,
Ajmer (Rajasthan) [INDIA]
Email: aloney12@gmail.com

Abstract**—***It is well known that the classic image compression techniques such as JPEG and MPEG have serious limitations at high compression rate, the decompressed image gets really fuzzy or indistinguishable. To overcome this problem, artificial neural networks ANNs techniques are used. This paper presents a neural network based technique that may be applied to data compression. This paper 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, LZ Method, Run Length Method, LZ-77 are discussed as well as more recent methods for the compression of data presents a neural network based technique that may be applied to data compression. The proposed technique and images. Intelligent methods for data compression are reviewed including the use of Back propagation and Kohonen neural networks. The proposed method includes steps to break down large images into smaller windows for Lossless image compression/ decompression processes. Results obtained with proposed technique leads to better compression ratio at the same time preserving the image quality.*

Keyword:*—Neural Network, Data Compression, Image Compression, ANN*

1. INTRODUCTION

Data compression is often referred to as coding, where coding is very general term encompassing any special representation of data which satisfies a given need. As with any communication, compressed data communication only works when both the sender and receiver of the Informating understand the encoding scheme. For example, this text makes sense only if the receiver understands that it is intended to be interpreted as characters representing the English language. Similarly, compressed data can only be understood if the decoding method is known by the receiver. It is useful because it helps to reduce consumption of expensive resources such as hard disk space or transmission bandwidth i.e. a data file that suppose to takes up 50 kilobytes (KB) could be downsized to 25 kilobytes (KB), by using data compression software. A simple characterization of data compression is that it Involves transforming a string of characters in some representation (such as ASCII) into a new string (of bits, for example) which contains the same information but whose length is as small as possible. Data compression has important application in the areas of data transmission and data storage. Compressed data required smaller storage size and reduce the amount of data that need to be transmitted. Hence, it increases the capacity of the communication channel. There are "lossless" and "lossy" forms of data compression. Lossless data compression is used when the data has to be uncompressed exactly as it was before compression. Text files are stored using lossless techniques, since losing a single character can in the worst case make the text dangerously misleading. Archival storage of master sources for images, video data, and audio data generally needs to be lossless as well. However, there are strict limits to the amount of compression that can be obtained with lossless compression. Lossless compression ratios are generally in the range of 2:1 to 8:1.

Lossy compression, in contrast, works on the assumption that the data doesn't have to be stored perfectly. Much information can be simply thrown away from images, video data, and audio data, and when uncompressed such data will still be of acceptable quality.

Compression ratios can be an order of magnitude greater than those available from lossless methods. Several techniques have been used for data compression. Each technique has their advantages and disadvantages in data compression.

2. RELATED WORK 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 Fano method, Statistical modeling and their variations [4]. 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 optimization is needed to get performance up to acceptable levels.

2.3 LZ-77

Another technique for data compression is LZ-77 encoding [7]. This technique is a simple, clever, and effective approach to compress text. This technique exploits the fact that words and phrases within a text stream are likely to be repeated. When they repeat, they can be coded as a pointer to an earlier occurrence, with the pointer accompanied by the number of characters to be matched. This technique is useful for compressing text because it able to reduce the file size and increase the compression ratio after compression. However, it is not efficient for image file format such bmp, gif, tif and tiff. Beside that, this technique will take several minutes to compress a data. Sometimes, the long processing time will cause the missing of some characters.

2.4 LZW Method

The most popular technique for data compression is Lempel Ziv Welch (LZW) [8]. LZW is a general compression algorithm capable of working on almost any type of data. It is generally fast in both compressing and decompressing data and does not require the use of floating-point operations. LZW technique also has been applied for text file. This technique is very efficient to compress image file such tiff and gif. However, this technique not efficient for compress text file because it require many bits and data dictionary.

2.5 Run Length Method

One of the techniques for data compression is “run length encoding”, which is sometimes knows as “run length limiting” (RLL) [5, 6]. Run length encoding is very useful for solid black picture bits. This technique can be used to compress text especially for text file and to find the repeating string of characters. This compression software will scan through the file to find the repeating string of characters, and store them using escape character (ASCII 27) followed by the character and a binary count of he number of items it is repeated. This compression software must be smart enough not to compress strings of two or three repeated characters but more than that. Instead, if the compression software is not smart, this technique will produce the bigger size than original size. First problem with this technique is the output file is bigger if the decompressed input file includes lot of escape characters. Second problem is that a single byte cannot specify run length greater than 256.

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 Back propagation algorithm and the Kohonen self-organizing maps.

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 generalization capability of the back propagation 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.

In general the images used for compression are of different types like dark image, high intensity image etc. When these images are compressed using Back-propagation Network, it takes longer time to converge. To achieve this, a cumulative distribution function is estimated for the image and it is used to map the image pixels. When the mapped image pixels are used, the Back-propagation Neural Network yields high compression ratio as well as it converges quickly.

3.1 Back propagation Neural Network

The Back propagation (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

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 organized 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 neighboring nodes are adjusted. This ensures that the output of these nodes is 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 Back propagation 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 finalized code-book and attend the competition governed by learning rules to obtain the optimized 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

4.1 Multilayer perceptron algorithm

This section presents the architecture of a feed word neural network that is used to compress image in the research works. Multilayer-perceptron algorithm [7, 8] is a widely used learning algorithm in Artificial Neural Networks. The Feed-Forward Neural Network architecture 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. In the forward pass the input vector is applied to the sensory nodes of the network and its effect propagates through the network layer by layer. Finally a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weight of the networks are all fixed. During the back pass the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from the desired response to produce an error signal. This error signal is then propagated backward through the network against the direction of synaptic conditions. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response.

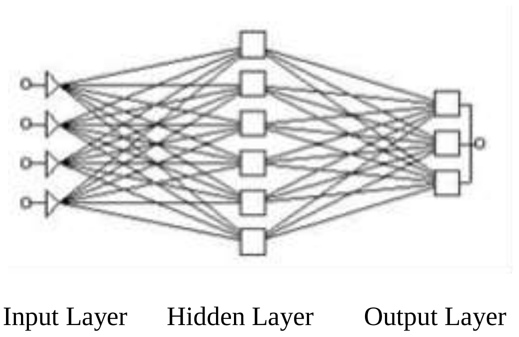

Figure 1. A Typical Feedforward Network

4.2 Algorithm:

The algorithm for Perceptron Learning is based on the back-propagation rule discussed previously. This algorithm can be coded in any programming language, and in the case of this tutorial, Java for the applets. In this case we are assuming the use of the sigmoid function *f(net)* described earlier in the tutorial. This is because it has a simple derivative.

Algorithm:

1. Initialize weights and threshold.

Set all weights and thresholds to small random values.

2. Present input and desired output

Present input $\boldsymbol{X_p} = x_0, x_1, x_2, \ldots, x_{n-1}$ and target output $\boldsymbol{T_p} = t_0, t_1, \ldots, t_{m-1}$ where n is the number of input nodes and m is the number of output nodes. Set w_0 to be $-ø$, the bias, and x_0 to be always 1. For pattern association, $\boldsymbol{X_p}$ and $\boldsymbol{T_p}$ represent the patterns to be associated. For classification, $\boldsymbol{T_p}$ is set to zero except for one element set to 1 that corresponds to the class that $\boldsymbol{X_p}$ is in. where the sum(in the [brackets]) is over the *k* nodes in the layer above node *j*.

4 .1 Training the Multilayer Perceptron

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, {wji}, 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

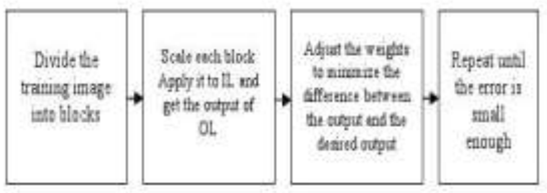
Divide the training image into blocks
Scale each block Apply it to IL and get the output of OL
Adjust the weights to minimize the difference between the output and the desired output
Repeat until the error is small enough

Figure 2: Block Diagram of MFFANN Training

3. Calculate the actual output

Each layer calculates the following:

$y_{pj} = f\,[w_0x_0 + w_1x_1 + \ldots. + w_nx_n]$

This is then passes this to the next layer as an input. The final layer outputs values o_{pj}.

4. Adapts weights

Starting from the output we now work backwards.

$w_{ij}(t+1) = w_{ij}(t) + ñ\text{Þ}_{pj}o_{pj}$, where $ñ$ is a gain term and Þ_{pj} is an error term for pattern *p* on node *j*.

For output units

$þ_{pj} = ko_{pj}(1 - o_{pj})(t - o_{pj})$ For hidden units

$þ_{pj} = ko_{pj}(1 - opj)[(þ_{p0}w_{j0} + þ_{p1}w_{j1} + \ldots + þpkwjk)]$

4.2 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. Figure 3 shows the encoding steps.

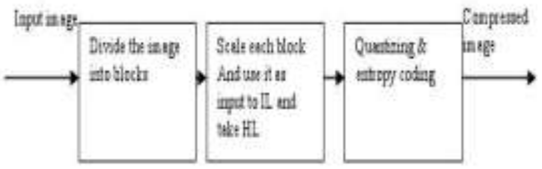
Input image
Divide the image into blocks
Scale each block And use it as input to IL and take HL
Quantizing & entropy coding
Compressed image

Figure 3: encoding steps

4.3 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. Figure 4 show the decoder block diagram.

4.4 Weight Adjustment

The networks weights need to be adjusted in order to minimize 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 ith neuron at iteration n is given by

$$e_i (n) = X_i (n) - X'_i (n)$$

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.

$$E_l (n) = \sum e_i^{\,2}(n)$$

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

$$E = \sum E_l\,,\; E_l = (\,X'\,,\,w)$$

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 minimize the error function with respect to weight vector (w) it is necessary to find an optimal solution (w*) that satisfy the condition

$$E\,(\,w^*) \leq E\,(\,w)$$

The necessary condition for the optimality is

$\Delta E\,(\,w) = 0$ where Δ is gradient operator

$\Delta = [\partial\,/\,\partial w]$ and ΔE (w) is gradient vector (g) of error function is defined as follows

$$\Delta E\,(\,w) = \partial E\,/\,\partial w$$

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 eight vectors w(1), w(2) … such that the error function is reduced for each iteration

$$E\,(\,wn + 1\,) \leq E\,(\,wn\,)$$

5. CONCLUSION

Our main work is focused on the image compression by using multilayer perceptron, which exhibits a clear-cut idea on application of multilayer perceptron with special features compare to Hoffman code. By using this algorithm we can save more memory space, and in case of web applications transferring of images and download should be fast. We can develop this project for image encryption and decryption. In this process compression of image is as similar data encryption, and decompression as decryption. Only produce a secret key to thwart the attacker.

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.