Volume 3, Issue 2, February 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Image Compression using Growing Self Organizing Map

Aslam Khan
RGPV Bhopal, India

Sanjay Mishra
RGPV Bhopal, India

Abstract: Image compression algorithm is needed that will reduce the amount of Image to be transmitted, stored and analyzed, but without losing the information content.This paper presents a neural network based technique that may be applied to image compression. Conventional techniques such as Huffman coding and the Shannon Fano method, LZ Method, Run Length Method, LZ-77 are more recent methods for the compression of data. A traditional approach to reduce the large amount of data would be to discard some data redundancy and introduce some noise after reconstruction. We present a neural network based self-organizing Kohonen map technique that may be a reliable and efficient way to achieve vector quantization. Typical application of such algorithm is image compression. Moreover, Kohonen networks realize a mapping between an input and an output space that preserves topology. This feature can be used to build new compression schemes which allow obtaining better compression rate than with classical method as JPEG without reducing the image quality.

Key words: Neural Network, Image Compression, ANN, Kohonen Network, Image

1. INTRODUCTION

Image compression [13] 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 information 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. Image 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. The purpose of this research is to investigate a method for sending data speedily across the Internet using a 'predictive' data compression model. This method adopts an artificial neural network technique coupled with a data encoder-decoder engine, in an attempt to provide that solution.

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. NEURAL NETWORK BASED METHOD FOR IMAGE COMPRESSION

Artificial Neural Networks have been applied to many problems [3][11], 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.

3.1 Back propagation Neural Network

The Back propagation (BP) algorithm[12] 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]. 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].

3.2 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 [9]. In fact it was found that the signal to noise ratio was quite poor when using a BP neural network [9]. 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.

4. PROPOSED TECHNIQUES FOR IMAGE COMPRESSION

4.1 Growing Self Organizing Map Algorithm

A growing self-organizing map (GSOM) is a growing variant of the popular self-organizing map (SOM). The GSOM was developed to address the issue of identifying a suitable map size in the SOM. It starts with a minimal number of nodes (usually 4) and grows new nodes on the boundary based on a heuristic. By using the value called Spread Factor (SF), the data analyst has the ability to control the growth of the GSOM.

All the starting nodes of the GSOM are boundary nodes, i.e. each node has the freedom to grow in its own direction at the beginning. New Nodes are grown from the boundary nodes. Once a node is selected for growing all its free neighboring positions will be grown new nodes. The figure shows the three possible node growth options for a rectangular GSOM.

In GSOM, 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

4.2 The Learning Algorithm of the GSOM:

The GSOM process is as follows:

1. Initialization phase:

1. Initialize the weight vectors of the starting nodes (usually four) with random numbers between 0 and 1.

2. Calculate the growth threshold (GT) for the given data set of dimension D according to the spread factor (SF) using the formula $GT = -D \times \ln(SF)$

2. Growing Phase:

1. Present input to the network.
2. Determine the weight vector that is closest to the input vector mapped to the current feature map (winner), using Euclidean distance. This step can be summarized as: find q' such that $|v - w_{q'}| \leq |v - w_q| \, \forall q \in \mathbb{N}$ where v, w are the input and weight vectors respectively, q is the position vector for nodes and $\mathbb{N}$ is the set of natural numbers.
3. The weight vector adaptation is applied only to the neighborhood of the winner and the winner itself. The neighborhood is a set of neurons around the winner, but in the GSOM the starting neighborhood selected for weight adaptation is smaller compared to the SOM (localized weight adaptation). The amount of adaptation (learning rate) is also reduced exponentially over the iterations. Even within the neighborhood, weights that are closer to the winner are adapted more than those further away. The weight adaptation can be described by $w_j(k+1) = \begin{cases} w_j(k) & \text{if} j \notin N_{k+1} \\ w_j(k) + LR(k) \times (x_k - w_j(k)) & \text{if} j \in N_{k+1} \end{cases}$ where the Learning Rate $LR(k)$, $k \in \mathbb{N}$ is a sequence of positive parameters converging to zero as $k \to \infty$. $w_j(k)$, $w_j(k+1)$ are the weight vectors of the node j before and after the adaptation and N_{k+1} is the neighbourhood of the winning neuron at the $(k+1)$th iteration. The decreasing value of $LR(k)$ in the GSOM depends on the number of nodes existing in the map at time k.
4. Increase the error value of the winner (error value is the difference between the input vector and the weight vectors).
5. When $TE_i > GT$ (where TE_i is the total error of node i and GT is the growth threshold). Grow nodes if i is a boundary node. Distribute weights to neighbors if i is a non-boundary node.
6. Initialize the new node weight vectors to match the neighboring node weights.
7. Initialize the learning rate (LR) to its starting value.
8. Repeat steps 2 – 7 until all inputs have been presented and node growth is reduced to a minimum level.

3. Smoothing phase.

1. Reduce learning rate and fix a small starting neighborhood. Find winner and adapt the weights of the winner and neighbors in the same way as in growing phase.

5. PERFORMACE EVOLUTION

In order to test the compression engine, several files of various formats were run through the scheme. This system uses three metrics such as compression ratio, transmission time. Compression ratios are defined as

$$Z = \frac{\text{Compressed Length (in bytes) x 100}}{\text{Total Length (in bytes)}}$$

$$\text{Compression Ratio} = 100 - Z$$

$$\text{Difference ratio} = \frac{\text{old method ratio} - \text{Proposed method ratio}}{\text{Old method ratio}}$$

The results below would then agree with the above. Equation

6. CONCLUSION

Our main work is focused on the data compression by using Growing Self Organizing Map algorithm, 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 data 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.

REFRENCES: -

1. R. C.Gonzales, R. E. Woods, Digital Image Processing, Second Edition, Prentice-Hall, 2002.
2. Veisi H., Jamzad M., Image Compression Using Neural Networks,Image Processing and Machine Vision Conference (MVIP), Tehran, Iran, 2005.

3. Missing Data Estimation Using Principle Component Analysis and Autoassociative Neural Networks. Jaisheel Mistry, Fulufhelo V. Nelwamondo, Tshilidzi Marwala. 3, s.l. : iiisci.org, Journal of Systemics, Cybernatics and Informatics, Vol. 7, pp. 72-79, 2009,
4. M. Egmont-Ptersen, D. de Ridder, H. Handels, "Image processing with neuralnetworks- a review", Pattern Recognition , 35, pp. 2279- 2301, 2002.
5. B. Verma, M. Blumenstein, S. Kulkarni, "A new compression technique using an artificial neural network". Faculty of Information and Communication Technology, Griffith University, Australia, 2004.
6. J. Jiang, "Image compression with neural networks – A survey, Signal Processing": Image Communication, 14, pp. 737 – 760, 1999.
7. S. Anna Durai and E. Anna Saro "Image Compression with Back-Propagation Neural Network using Cumulative Distribution Function" World Academy of Science, Engineering and Technology 17, 2006.
8. Martin Riedmiller and Heinrich Braun "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm" Institut fur Logik, Komplexitat und Deduktionssyteme, University of Karlsruhe, W-7500 Karlsruhe, FRG, 1993.
9. Md. I. Bhuiyan, Md. K. Hassan, "Image compression with neural network using DC algorithm", Journal of signal processing, vol. 5, no. 6, pp. 445 -459, 2001
10. S.S Panda,M.S.R.S Prasad Ch.SKVR Naidu"Image compression using backpropagation neural network",International journal of engineering science and advance technology volume -2,issue-1,2012 pp:74-78 ISSN:2250-3676.
11. Neelmani chhedaiya,prof.Vishal Moyal "Implementation of Backpropagation algorithm in Verilog",International Journal Computer Technology and application volume-3,issue-1,2012,pp:340-343:ISSN:2229-6093.
12. K.Siva Nagi Reddy,Dr. B.R.Vikram,L.Koteswara Rao"Image Compression and reconstruction using a new approach by artificial neural network",International journal of image processing ",volume -6 ,Issue-2,2012,pp:68-85.
13. Anuj Sharma & Mahendra Pratap Panigraphy" neural network and image compression",VSRD-International journal of computer science and information technoology",volume-2,Issue-9,2012,pp:746-755.

