Using Perceptron Neural Network and Genetic Algorithm for Image Compression and Decompression

Mohammed Mustafa Siddeq
College of Technology /Kirkuk
Software Engineering Depart.
e-mail: mamadmmx76@yahoo.com

Dalya Abdullah Anwar
University of Salahhaden/ Erbil
College of Science Education/
Computer Depart.
e-mail: dalea84@yahoo.com

Abstract

This paper introduces an idea for image compression by using *Genetic Algorithm and Arithmetic Coding*. First stage is by using perceptron neural network to compress each three-pixels into single value, this value is called *Compression value*. In this paper the neural network did not need weights for training at compression part, the weights used in this paper are of one dimensional array containing floating point values. The total of weights values equal one. In the decompression part we use Genetic Algorithm for return the pixels, by using Crossover operation and the *Fitness Value*. The fitness value represented the error between *Compression value* and *Desired output* for each generated string by GA to get approximately original three-pixels. The second stage is Arithmetic coding algorithm uses to convert vector of compression values into a single floating point number. Our approach tested with color image, also in this paper the performance of the algorithm is computed and compared wit PNG and TIFF algorithm.

1. Introduction

The transport of images across communication paths is an expensive process. Image compression provides an option for reducing the number of bits in transmission. This in turn helps increase the volume of data transferred in a space of time, along with reducing the cost required. It has become increasingly important to most computer networks[9,6], as the volume of data traffic has begun to exceed their capacity for transmission. Traditional techniques that have already been identified for data compression include: Predictive coding, Transform coding and Vector Quantization. [9,6,10,8].In brief, predictive coding refers to the decorrelation of similar neighboring pixels within an image to remove redundancy. Following the removal of redundant data, a more compressed image or signal may be transmitted [10,8]. The most common general-purpose, lossless compression algorithm used with TIFF is LZW, which is inferior to PNG [5]. There is a TIFF variant that uses the same compression algorithm as PNG uses, but it is not supported by many proprietary programs. TIFF also offers special-purpose lossless compression algorithms like CCITT Group IV, which can compress bi-level images (e.g., faxes or black-and-white text) better than PNG compression algorithm [5]. Transform-based compression techniques have also been commonly employed. These techniques execute transformations on images to produce a set of coefficients. A subset of coefficients is chosen that allows good data representation (minimum distortion) while maintaining an adequate amount of compression for transmission. The results achieved with a transform-based technique are highly dependent on the choice of transformation used (cosine, wavelet, Karhunen-Loeve etc.) [1,4]. Finally, vector quantization techniques require the development of an appropriate codebook to compress data. Usages of codebooks do not guarantee convergence and hence do not necessarily deliver infallible decoding accuracy. Also the process may be very slow for large codebooks as the process requires extensive searches through the entire codebook [7,3]. Following the review of some of the traditional techniques for image compression, it is possible to discuss some of the more recent techniques that may be employed for data

compression. Artificial Neural Networks (ANNs) have been applied to many
problems [3,5], and have demonstrated their superiority over traditional methods when dealing with noisy or incomplete data. One such application is for image compression. Neural networks seem to be well suited to this particular function, as they have the ability to preprocess input data to produce simpler data with fewer components [10,8,1]. This compressed information (stored in a hidden layer) preserves the full information obtained from the external environment. ANN based techniques not only can provide sufficient compression rates of the data in question, but 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. There have already been an exhaustive number of papers published applying ANNs to image compression [1-3]. Many different training algorithms and architectures have been used [5].

2. Proposed Compression Algorithm

In this paper we introduce an idea for image compression by using Perceptron Neural Network and decompression by Genetic algorithm and Arithmetic coding. This approach reduces image size to half, reduce each three image pixels into integer number. The weights values are constant for using them in compression and decompression; this leads the neural network do not need to store information (i.e. Header file) about compressed image.

2.1 Compression image by Linear Perceptron Neural Network (LPNN)

This approach is faster for compression, because the number of data are used in this neural network is few, also the neural network consists of input layer and output layer as shown in Figure-1.

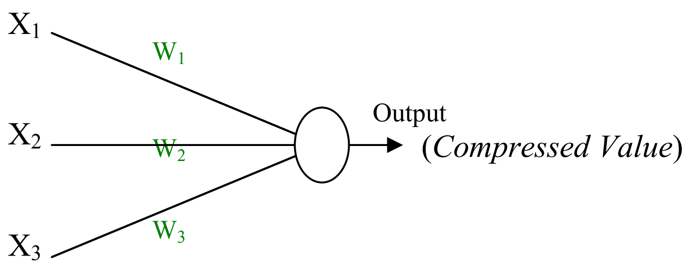


Figure – 1. Perceptron Neural network

The numbers of data are used for compression, three input data (X_1, X_2, X_3) and the final output of this neural is single value. This value called *Compression Value* and represented as (8-Bit), as shown in the following equation [1]:-

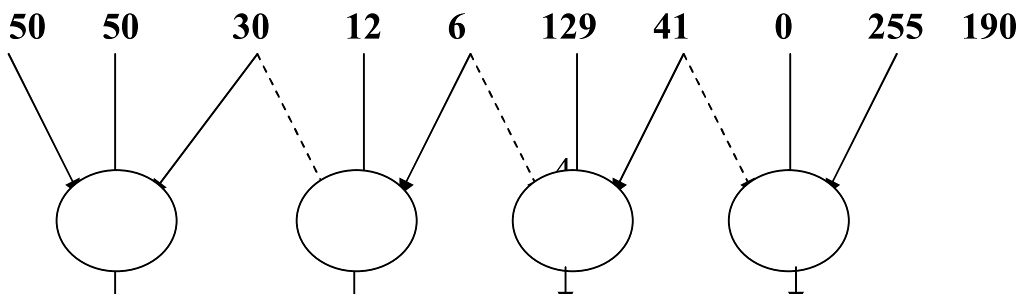
$$\textit{Compression Value} = \sum_{i=1}^{n} W_i X_i \qquad (1)$$

Where i =1,2,3

In equation (1), the weight values are constant for image compression, this means that the weights do not need for learning. The weights values can represent one-dimensional array containing floating point numbers, and total of weights values equivalent to one. Assume weights values W= [0.2 , 0.3 , 0.5]. The idea of weights values is similar to (Mask Filter $_{3x3}$) used in image enhancement and image restoration, and total of (Mask Filter $_{3x3}$) equivalent to one[1,4]. To explain image compression by using neural network in detail, Figure – 2 shows pixels compression by neural networks:

[50 , 50 , 30 , 12 , 6 ,129 ,41 , 0 ,255, 190]

Where W_1= **0.2**; W_2= **0.3**; W_3 =**0.5**;



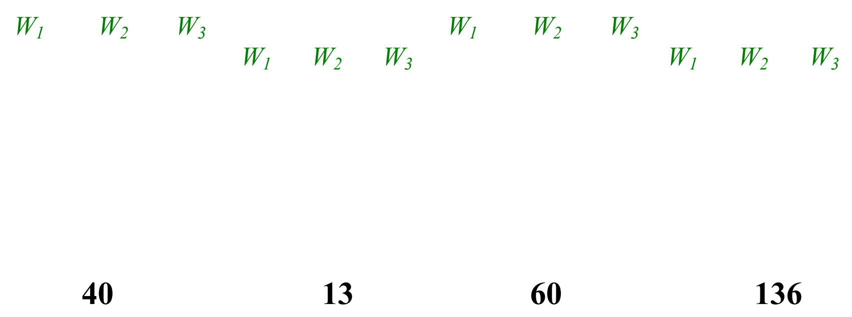


Figure – 2 neural network used for image compression

In Figure - 2 the input data {"50 , 50 , 30"} are compressed by using neural networks to produce output value ("40"), and ("12 , 6") shared with 3rd input data ("30") to produce 2nd output value ("13"). This process continues until compress all data. The last input data ("190") will be ignored, because cannot use ("190") just for compression, for this reason the last element is ignored; this leads in decompression getting nine data.

2.2. Decompression image by Genetic Algorithm (GA)

The genetic algorithm is a parallel search algorithm by using a number of strings and computing fitness value for each string, these strings are shared with each other by crossover function until reached to the result [2]. The fitness function for each string could be computed as:

$$\textit{Desired Output} = \sum_{i=1}^{n} Wi\, Pi \qquad (2)$$

Where \ i =1,2,3

In equation (2) the vector $\underline{P}$ = [P_1 , P_2 , P_3] represents the estimated data generated by Genetic Algorithm, and the *Wi* represents weights values (See Figure-2) the GA generates many strings randomly then using the crossover function to find the suitable vector. The crossover operation divides any two vectors randomly from selected point, and then makes exchange between these strings to generate new strings, as shown in the following representation:

Before Crossover	After Crossover
S1: 255, 34, 67	S1: 12, 34, 67
S2: 12, 50, 100	S2: 255, 50, 100

After Generating new strings, the fitness function is used to compute the fitness value for each string. The fitness function computed by using; the difference between compressions value (See equation 1) and desired output (See equation 2). The vector $\underline{P}$ is selected according to minimum fitness value:-

$$\textit{Fitness Value}(k) = (\textit{Compression Value} - \textit{Desired Output}) \qquad (3)$$

Where\ i =1,2,3.

k= number of string generated by GA.

Equation (3) is used to generate vector $\underline{P}$, if the desired output is equivalent to the *compression value,* this means *Fitness Value(k)= 0*, in this case the GA stops from iterations, and vector $\underline{P}$ represents approximately original data. Figure – 3 illustrates the decompression by GA for the compression values **{ 40 , 13 }.** The GA generates five pixels from two compression values.

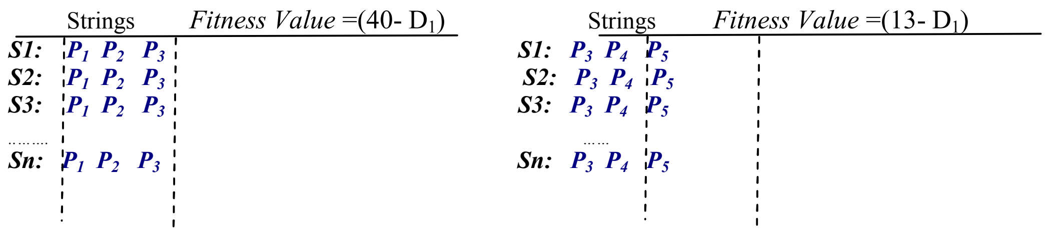


Figure – 3 Decompressions by GA

3. Arithmetic Coding

The second part in this paper, by using the arithmetic coding we have taken a stream of data (i.e. compressed data from neural network) and convert it to a single floating point value. This output value in range less than 1 and greater than 0, when decoded this single value getting exact stream of data, see the reference[9]. The arithmetic coding needs to compute the probability of all data and assign range for each data, the range value consists of Low and High value.

4. Computer Test

The compression and decompression applied on the color "Cat" image. The weights values: "**W = [0.2 , 0.3 , 0.5]**" are using for the image and the number of strings generated by the GA is 256 strings. The language used is (**VISUAL C++.NET**) on **Dual-Core Pentium 1.8GHz**.

The color image consists of three layers (i.e. Red, Green and Blue), our approach compresses each layer independently, and the total time for decompression by GA is 5 seconds and the average iterations taken for decompression by GA are 126 iterations.

(a) Original **Cat** image 350 x 350 (b) Decompressed **Cat** image

Figure – 4 Decompressed images by our approach.

To compute the efficiency of our approach we used *Compression Performance* as shown in the following equation [6]:-

$$C.R. = \frac{Size\ after\ compression}{Size\ before\ compression} \tag{4}$$

$$Compression\ Performance = [100(1 - C.R.)]\% \tag{5}$$

Our approach as compared with The TIFF and PNG, these algorithms represented lossless data compression methods, which are used in compression of library. Table 1 has shown the comparison between our approach and both, PNG and TIFF.

Table 3 Comparison with PNG and TIFF

Algorithm	**Before Compression**	**After Compression**	**Compression Performance**
GA with Arithmetic	358-KByte	120-KByte	66.4%
PNG	358-KByte	280-KByte	8%
TIFF	358-KByte	380-KByte	NON

5. Conclusion

The advantage of this approach can be illustrated in the following steps:-

1– Our algorithm gives better image compression ratio, than PNG and TIFF. This because combine each three-bytes from an image to be single byte by using LPNN.

2– The GA computes the fitness values by equation (3) for each $\underline{P}$ vector (i.e. vector of pixels) after making crossover randomly between any two vectors. All new vectors are transferred into next generation without removing any string. Moreover, in this algorithm we did not use the mutation. This operation makes our algorithm finding pixels after a few iterations.

3– In our approach, the LPNN does not need to compute the probability of image file; also the weights values used in the neural network are constant at compression and decompression operations.

The disadvantage of our approach, that it needs more computations (arithmetic coding) and recurrence calculating which may be leading to increase time execution for compression and decompression. Also the image quality for our approach is less than PNG and TIFF. This is because the GA can not find exact original value for some compressed data.

RERERENCES

[1] **Abhijit S. Pandya and Robert B. Macy,** *"Pattern Recognition with Neural Networks in C++"*, IEEE Press 1995.

[2] **Dalya A. Anewar,** *"Qualified Genetic Algorithms for Image Smoothing*", M. Sc. – Thesis, University of Mousl, Collage of Computer Sciences and Mathematics, Mathematic Dept. 2006.

[3] **Dony, R. D., and Haykin**, S., *"Neural Network Approaches to Image Compression"*, Proceedings of the IEEE, Vol. 23, No. 2, pp 289-303., 1995.

[4] **Hambaba, M., Coffey B., and Khemlani, N.,** "*Image Coding using a Knowledge based Recognition System"*, SPIE Vol. 1709., Application of Artificial Neural Networks, 1992.

[5] **I. Kontoyainnis**, "Pattern matching and lossy data compression on random field", *IEEE Trans. on Inform. Theory,* 49, pp. 1047-105 1. April, 2003.

[6] **Mohammed M. Siddeq**, "SEQUNCE DYNAMIC CODE FOR STREAM TWO BYTE DATA (16-Bit) COMPRESSION", *Al-Taqani Journal*, Vol. 19,No.2,pp 85-99, 2006.

[7] **Oja, E.,** "*Data Compression, Feature Extraction, and Autoassociation in Feed forward Neural Networks in Artificial Neural Networks*", (Eds) Kohonen et al., Elsevier Science Publishers, pp 737-745., 1991.

[8] **Sarmad M. Al-Kaysi,** *"Image Restoration using Neural Networks*", M. Sc. – Thesis, University of Baghdad, Collage of Science, Computer Science Dept. 2000.

P_1

Desired

[9] Witten, Ian H., Neal, Radford M., and Cleary, John G. *"Arithmetic Coding for Data Compression"*, Communications of the ACM, June, pp 520-540, 1987.

[10] Z Xiong, X. Wu, S. Cheng, and J. Hua, Loss-to-lossless compression of edical volumetric data using three-dimensional integer wavelet transforms, IEEE Trans. on Medical Imaging, Vol. 22, No. 3, March 2003.