The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 12 || Pages || 10-17 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Modeling of neural image compression using gradient decent technology

Anusha K L[1], Madhura G[2], Lakshmikantha S[3]
[1,] *Asst.Prof, Department of CSE,EWIT, Bangalore*
[2,] *Asst.Prof, Department of CSE,EWIT, Bangalore*
[3,] *Asst.Prof, Department of CSE,EWIT, Bangalore*

----------**ABSTRACT**----------
In this paper, we introduce a new approach for image compression. At higher compression rate, the decompressed image gets hazy in GIF and PNG, an image compression technique. In order to overcome from those problems artificial neural networks can be used. In this paper, gradient decent technology is used to provide security of stored data. BP provides higher PSNR ratio, with fast rate of learning. Experiments reveal that gradient decent technology works better than Genetic algorithm where the image gets indistinguishable as well as not transmitted in a secured manner.

Keywords: *Image-compression, back propagation, gradient decent, artificial neural network, genetic algorithm*

Date of Submission: 19 June 2014 Date of Accepted: 20 December 2014

I. INTRODCUTION

Artificial neural network(ANN) are computational models inspired by animals 'central nervous system'(particular brain)[5,6,8]which are capable of machine learning and pattern recognition .Like the brain, this computing model consist of many small units or neurons or node (the circle) are interconnected .There are several different neural networks models that differ widely in functions and applications. In [1,2,3,4,7]paper we discuss the feed-forward network with back propagation learning .From mathematical view; a feed-forward neural network is function. It takes an input and produces an output. The input and output are represented by real numbers. A simple neural network may be illustrated like in Fig.1.

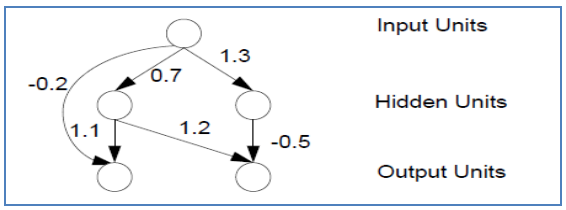

Fig 1: Simple Neural Network

This network consists of 5 units or neurons or nodes (the circle) and 6 connections (the arrows). The number next to each connection is called weight; it indicates the strength of the connection .Positive weight excitatory. Negative weight inhibitory. This is a feed forward network, because the connections are directed in only one way from top to bottom. There are no loops or circles.

II. FEED FORWARD ARTIFICIAL NEURAL NETWORKS

Information processing in neural network. The node receives the weighted activation. First these are added up (summation). The result is passed through an activation function. The outcome is the activation of the node. For each of the outgoing connections, this activation value is multiplied with the specific weight and transferred to the next node. It is important that a threshold function is non-linear. Otherwise a multilayer network is equivalent to a one layer network. The most applied threshold function is sigmoid.

However, the sigmoid is the most common one. It has some benefits for back propagation learning, the classical training algorithm for feed forward neural network.

A. Evolution of weights in neural network using GA.

Genetic algorithm starts with the random generation of an initial set of individuals, the initial population. The individuals are evaluated and ranked. It follows Darwin "survival of fittest rule" one with worst fitness value is discarded. There are 2 basic operators to generate new individual.Mutation is simpler one. Crossover is done by taking parts of the bits string of one of the parents and the other parts from the other parent and combining both in the child. Three kinds of crossover, one-point, two-point and uniform crossover.

$$f(v) = \frac{1}{1 + e^{-v}}$$

B. Weight optimization with back propagation algorithm

Main idea of back propagation is to distribute the error function across the hidden layers, corresponding to their effect on the output. Works on feed-forward network. Back-propagation is one method to train the network. The training is performed by one pattern at a time. Back propagation is used for finding the optimal set of weights. It performs gradient decent and improves the performance of the neural network by reducing its error along its gradient. The error is expressed by the root-mean square (RMS)

$$E = \frac{1}{2}\sum_{p} \|t_p - o_p\|^2$$

P is vector over all patterns, tp is projected target, and o is actual output. Back propagation is done using delta rule.

Delta rule:

$\Delta Wji = \eta * \delta j * xi$

$\delta j = (tj - yj) * f'(hji)$

n is learning rate of the neural network, tj is targeted output of jth neuron, and yj is actual output.

Delta Rule for Multilayer Neural Networks, problem with Multilayer Network is that we don't know the targeted output value for the Hidden layer neurons.
This can be solved by a trick:

$\delta i = \Sigma(\delta k * Wki) * f'(hi)$

The first factor in parenthesis involving the sum over k is an approximation to (ti-ai) for the hidden layers when we don't know ti.

C. Algorithm for weights evolution by GA in ANN

1) Chromosome Encoding: The weights (and biases) in the neural network are encoded as a list of real numbers (see Fig. 2a).

2) Evaluation Function: Assign the weights on the chromosome to the links in a network of a given architecture, run the network over the training set of examples, and return the sum of the squares of the errors.
3) Initialization Procedure: The weights of the initial members of the population are chosen at random with a probability distribution given by *t*l rl . This is different from the initial probability distribution of the weights usually used in back propagation, which is a uniform distribution between -1.0 and 1.0. Our probability distribution reflects the empirical observation by researchers that optimal solutions tend to contain weights with small absolute values but can have weights with arbitrarily large absolute values. We therefore seed the initial population with genetic material which allows the genetic algorithm to explore the range of all possible solutions but which tends to favour those solutions which are a priori the most likely.

4) Operators: We created a large number of different types of genetic operators. The goal of most of the experiments we performed was to find out how different operators perform in different situations and thus to be able to select a good set of operators for the final algorithm. The operators can be grouped into three basic categories: mutations, crossovers, and gradients (see Fig.2b). A mutation operator takes one parent and randomly changes some of the entries in its chromosome to create a child. A crossover operator takes two parents and creates one or two children containing some of the genetic material of each parent. A gradient operator takes one parent and produces a child by adding to its entries a multiple of the gradient with respect to the evaluation function.

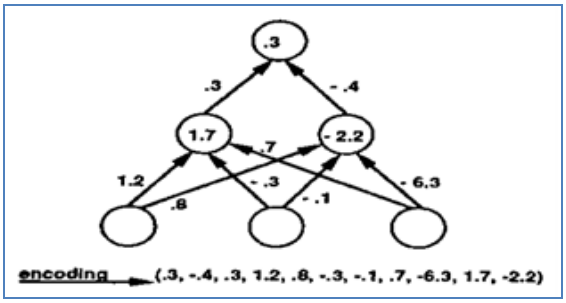

Fig 2a: Encoding Network on a Chromosome

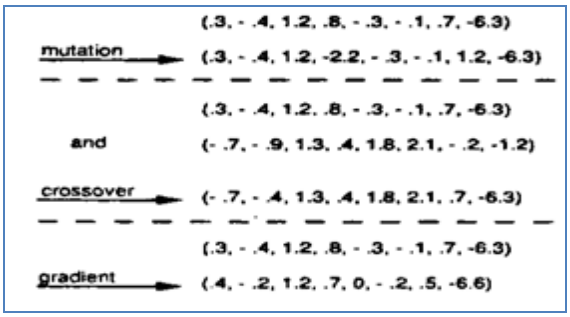

Fig 2b: Operation of the Operators

III. IMAGE COMPRESSION STRUCTURE USING NEURAL NETWORK

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the graphic below Fig 3:

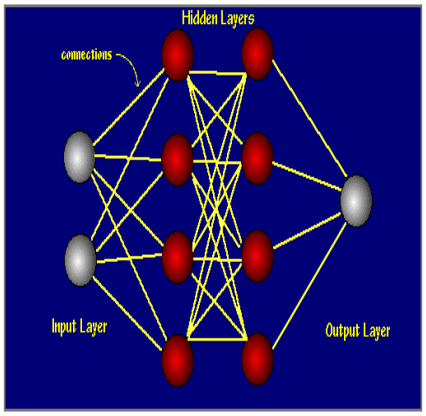

Fig 3: Output Graphic

Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The delta rule is often utilized by the most common class of ANN called ‘Back Preoperational Neural Networks' (BPNNs). Back propagation is an abbreviation for the backwards propagation of error. With the delta rule, as with other types of back propagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments.

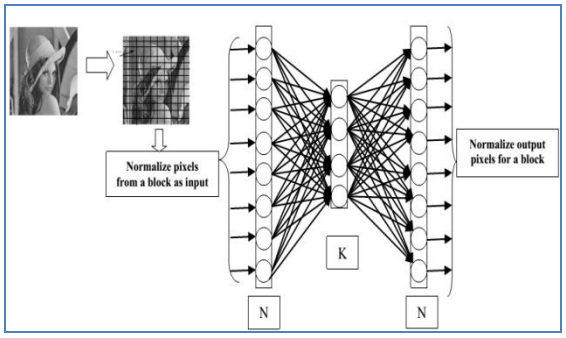

Fig 4: Architecture of neural network at time of training

More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights as shown in fig. 4. Compression process is defined by taking the half of the trained architecture which has been utilize at the time of training ,i.e. input layer along with the hidden layer as shown in Fig.5. Remaining half of the trained architecture i.e. hidden layer along with output layer is utilized to setup the decompression, as shown in Fig.6.

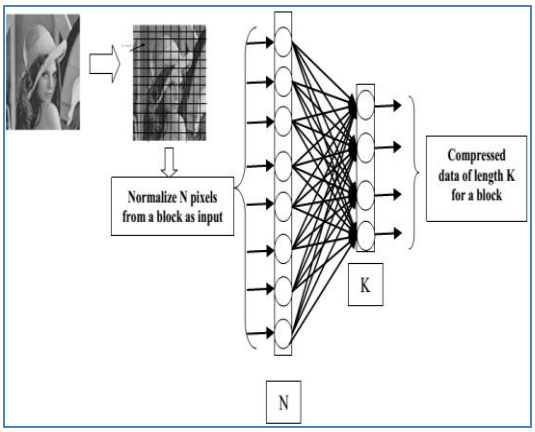

Fig 5: Compression module

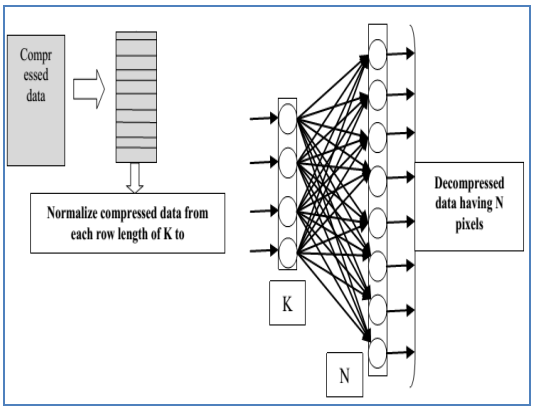

Fig 6: Decompression module

IV. PERFORMANCE PARAMETERS

Evaluation criteria used for comparison in this paper, is compression ratio (CR) and the peak signal-to-noise ratio (PSNR). For an Image with R rows and C columns, PSNR is defined as follows:

$$PSNR = 10\ log_{10}\left(\frac{255^2}{\frac{1}{RC}\sum_{I=1}^{R}\sum_{J=1}^{C}(X_{IJ}-\bar{X}_{IJ})^2}\right)$$

Compression ratio (CR) which is a criterion in compression problems is defined as the number of bits in original image to number of bits in the compressed image. This criterion in the sense of using neural net structure is defined as follow:

CR = N. B_I /B_H .K

In this equation N and K are the neurons/pixels available in the input and hidden layer respectively and B_I and B_H are the number of bits needed to encode outputs of input and hidden layer. If the number of bits needed to encode the input layer and the number of bits needed to encode the hidden layer be the same, the compression ratio will be the number of neurons in the input layer to hidden layer. As an example for the gray level images which are 8 bits long if we encode the compressed image with the same number of bits in a block of 8×8 and the network of with 16 neurons at the hidden layer, the compression ratio will be 4:1. And for the same network with floating point used to encode the hidden layer, the compression ratio will be 1:1 which means no compression.

To verify the developed design for evolving the weights of neural network two different experiments are considered as explained in the following section. This will give the confidence to apply the developed method for image compression.

V. IMAGE COMPRESSION WITH GA AND ANN

A population of 50 chromosomes is applied for evolving the weights up to 200 generations. Compression ratio for this experiment defined as 4:1. Performance plot for compression is shown in Fig.7 and in Fig.8 .Decompression result of Lena image is shown in Fig.9. Table (3) shows the parameter values. From the result it is very clear that the process is taking very long time for completing the cycle of generation. Even convergence is not proper and the result of compression performance is very poor and cannot be consider for practical purpose.

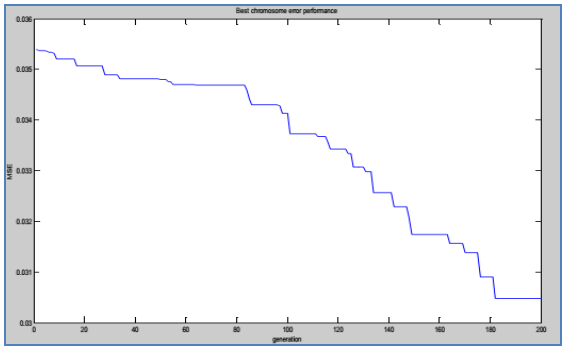

Fig 7: Best chromosome Error Plot

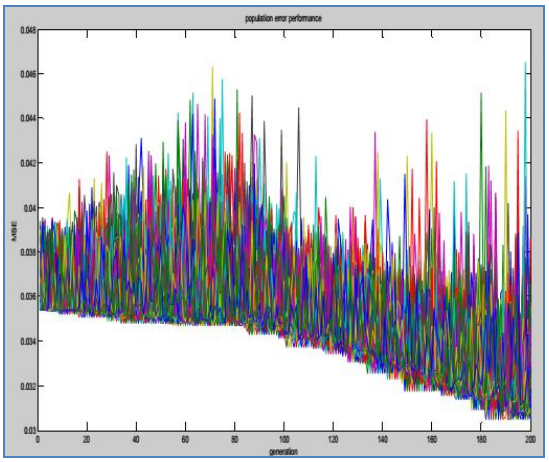

Fig 8: Population Error plot

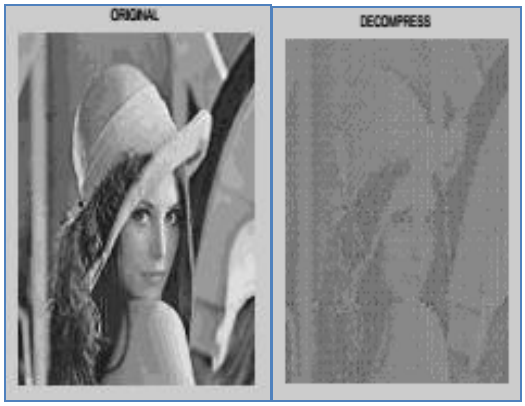

Fig 9: Decompression by GA with ANN

VI. IMAGE COMPRESSION USING ANN AND STANDARD BACK PROPAGATION ALGORITHM

Standard back propagation is one of the most widely used learning algorithms for finding the optimal set of weights in learning. A single image "Lena" is taken as training data. The error curve of learning is shown in Fig.10 for below defined set of parameters. Further, different images are tested to generalize the capability of compression. The processes repeated for two different compression ratios by changing the number of hidden nodes in neural architecture. The performance observed during the time of training and testing is shown in table 4, table 5 for compression ratio 4:1 and in table 6, table 7 or 8:1, respectively. Table 8 given the comparison with [9]

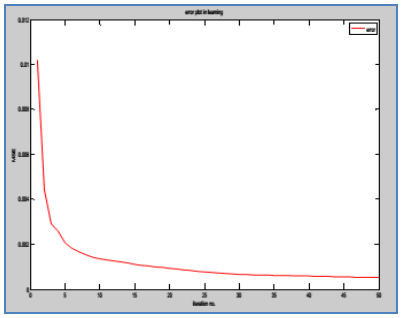

Fig 10: Error plot in back propagation

Parameter setting for back propagation learning:

Initial random weights value taken from uniform distribution in range of [-0.0005 +0.0005]. Learning rate:0.1Momentum constant: 0.5;Bias applied to hidden and output layer nodes with fixed input as (+1). Allowed number of iteration : 50

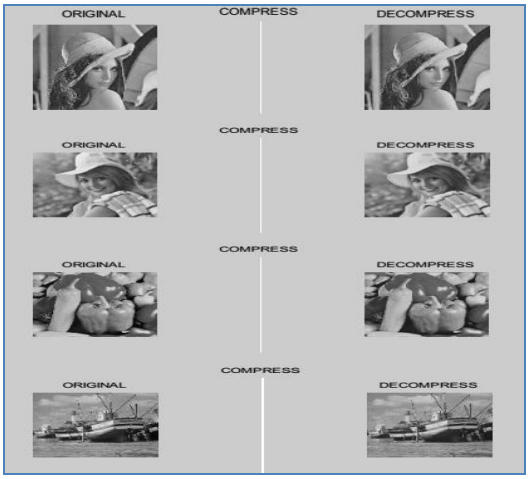

Fig 11: Performance by Gradient Decent

Table 3

Training image	Image format	Learning Time	No. of epoch.	Error	CR	PSNR(db)	Comp. Time(sec.)	Decom. Time(sec.)
Lena	BMP	1.771	200	0.035	4:1	15.15	0.0692	0.1665

Table 4

Training image	Image format	Learning Time	No. of epoch.	Error	CR	PSNR(db)	Comp. Time(sec.)	Decom. Time(sec.)
Lena	BMP	66.69	50	0.005	4:1	32.6017	0.0686	0.1881

Test image	Image format	CR	PSNR(db)	Comp. Time	Decom.Time
Elaine	TIF	4:1	31.0652	0.0691	0.1754
Boat	TIF	4:1	28.5801	0.0700	0.1844
Pepper	TIF	4:1	29.5953	0.0696	0.1875

Table 5

Training image	Image format	Learning Time	No. of epoch.	Error	CR	PSNR(db)	Comp. Time(sec.)	Decom. Time(sec.)
Pepper	TIF	61.32	50	0.005	4:1	31.4338	0.0692	0.1845

Test image	Image format	CR	PSNR(db)	Comp. Time	Decom.Time
Elaine	TIF	4:1	30.3149	0.0689	0.1834
Boat	TIF	4:1	28.4153	0.0693	0.1864
Pepper	BMP	4:1	31.4238	0.0696	0.1928

Table 6

Training image	Image format	Learning Time	No. of epoch.	Error	CR	PSNR(db)	Comp. Time(sec.)	Decom. Time(sec.)
Lena	BMP	43.03	50	0.0010	8:1	29.7739	0.0665	0.1799

Test image	Image format	CR	PSNR(db)	Comp. Time	Decom.Time
Elaine	TIF	8:1	29.4417	0.0336	0.1833
Boat	TIF	8:1	26.0333	0.0578	0.1790
Pepper	TIF	8:1	27.3020	0.0576	0.1821

Table 7

Training image	Image format	Learning Time	No. of epoch.	Error	CR	PSNR(db)	Comp. Time(sec.)	Decom. Time(sec.)
Pepper	TIF	47.4454	50	0.0012	8:1	28.6450	0.0675	0.1857

Test image	Image format	CR	PSNR(db)	Comp. Time	Decom.Time
Elaine	TIF	8:1	28.2311	0.0581	0.1795
Boat	TIF	8:1	26.2294	0.0583	0.1855
Lena	BMP	8:1	28.5439	0.0570	0.1871

Table 8

[Durai & Sarao]				Proposed method	
Image	*CR*	*PSNR(db)*	*Time(sec)*	*PSNR(db)*	*Time(sec)*
Lena	4:1	28.91	182	**32.60**	66.69
Pepper	4:1	29.04	188	**31.43**	61.32
Boat	4:1	29.12	178	**29.68**	64.75

Fig: 12 Compression ratios: 8:1

VII. CONCLUSION

There is increasing demand of image compression based processing in various applications. Compression techniques based on neural network have good scope. The GA performs better compared to back propagation algorithm. Security of compressed data is another inherent advantage available if compression happens by neural network i.e...; until weights are not available it is nearly impossible to find the contents of the image from compressed data.

REFERENCES

[1]. J.Jiang, "Image compression with neural networks: A survey".Signal Processing: Image Communication 14 (1999), 737-760
[2]. Moghadam, R.A. Eslamifar, M. "Image compression using linear and nonlinear principal component neural networks" .Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the London, Issue Date: 4-6 Aug. 2009, pp: 855 – 860.
[3]. Palocio, Crespo, Novelle,"image /video compression with artificial neural network".Springer-Verlag, IWANN 2009, part ii, LNCS 5518, pp: 330- 337.2009
[4]. Bodyanskiy,grimm,mashtalir,vinarski,"fast training of neural network for image compression".Springer-Verlag ,ICDM 2010,LNAI 6171, PP;165- 173,2010.
[5]. Liying Ma, Khashayar Khorasani," Adaptive Constructive Neural Networks Using Hermite Polynomials for Image Compression ",Advances in Neural Networks, springer, Volume 3497/2005, 713-722, DOI: 10.1007/11427445_115
[6]. Dipta Pratim Dutta, Samrat Deb Choudhury, Md Anwar Hussain, Swanirbhar Majumder, "Digital Image Compression Using Neural Networks," act, pp.116-120, 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009
[7]. B. Karlik, "Medical Image Compression by Using Vector Quantization Neural Network", ACAD Sciences press in Computer Science,vol. 16, no. 4, 2006 pp., 341-348.
[8]. Y. Zhou., C. Zhang, and Z. Zhang, "Improved Variance-Based Fractal Image Compression Using Neural Networks", Lecture Notes in Computer Science, Springer-Verlag, vol. 3972, 2006, pp. 575-580