Journal of Computer Science 6 (11): 1347-1354, 2010
ISSN 1549-3636

Improving the Performance of Backpropagation Neural Network Algorithm for Image Compression/Decompression System

Omaima N.A. AL-Allaf
Department of Computer Information Systems,
Faculty of Sciences and Information Technology,
AL-Zaytoonah Private University of Jordan, P.O. Box 130,
Amman (11733), Jordan

Abstract: Problem statement: The problem inherent to any digital image is the large amount of bandwidth required for transmission or storage. This has driven the research area of image compression to develop algorithms that compress images to lower data rates with better quality. Artificial neural networks are becoming attractive in image processing where high computational performance and parallel architectures are required. **Approach:** In this research, a three layered Backpropagation Neural Network (BPNN) was designed for building image compression/decompression system. The Backpropagation neural network algorithm (BP) was used for training the designed BPNN. Many techniques were used to speed up and improve this algorithm by using different BPNN architecture and different values of learning rate and momentum variables. **Results:** Experiments had been achieved, the results obtained, such as Compression Ratio (CR) and peak signal to noise ratio (PSNR) are compared with the performance of BP with different BPNN architecture and different learning parameters. The efficiency of the designed BPNN comes from reducing the chance of error occurring during the compressed image transmission through analog or digital channel. **Conclusion:** The performance of the designed BPNN image compression system can be increased by modifying the network itself, learning parameters and weights. Practically, we can note that the BPNN has the ability to compress untrained images but not in the same performance of the trained images.

Key words: Image compression, artificial neural networks, backpropagation neural network, backpropagation algorithm

INTRODUCTION

Artificial Neural Networks (ANN) models have been studied for many years in the hope of achieving human-like performance in the fields of pattern recognition, speech and image recognition where high computational rates are required. The Backpropagation Neural Network (BPNN) is the most widely used multi layer feed-forward ANN. The BPNN consists of three or more fully interconnected layers of neurons which can be trained by using the Backpropagation Algorithm (BP). The BP training can be applied to any multilayer NN that uses differentiable activation function and supervised training (Wasserman, 1989).

Image compression is a process of: representing an image with fewer bits while maintaining image quality (Gonzales and Wintz, 1987); saving cost associated with sending less data over communication lines and finally reducing the probability of transmission errors. In the literature, different ANN architectures and training algorithms have been developed for image compression which provided high Compression Ratio (CR) and high Signal to Noise Ratio (SNR).

The BPNN has the simplest architecture of ANN that has been developed for image compression but its drawback is very slow convergence. Many approaches have been carried out to improve the speed of convergence. These approaches are computationally complex in nature and applied only to limited patterns. At the same time, the images used for compression are of different types like dark image and high intensity image. When these images are compressed using BPNN, it takes longer time to converge because any given image may contain a number of distinct gray levels with narrow difference with their neighborhood pixels. Therefore, Durai and Saro (2006) suggested mapping the gray levels of the image pixels and their neighbors in such a way that the difference in gray levels of the neighbors with the pixel is minimized and then the CR and network convergence can be improved.

They achieved this by estimating a cumulative Distribution Function (CDF) for the image. They used CDF to map the image pixels, then, the BPNN yields high CR and converges quickly. Their experiments achieved CR equal 4:1 and Peak Signal to Noise Ratio (PSNR) equal 28.91 when applied this approach on (256×256) Lena image. While, Roy *et al.* (2005) developed an edge preserving image compression technique using one hidden layer feed forward BPNN of which the neurons are determined adaptively. Edge detection and multi-level thresholding operations are applied to reduce the image size. The processed image block is fed as single input pattern while single output pattern has been constructed from the original image unlike other NN based techniques where multiple image blocks are fed to train the network. Their experiment achieved CR (30:1) and SNR (0.3013) when they applied their proposed approach on Lena image.

Khalil (2007) proposed a bipolar sigmoidal BP (PPB) to train a feed forward auto associative NN. The proposed method includes steps to break down large images into smaller windows for image compression process. Experiments have been achieved CR (8:1) and PSNR (29.0) on applying PPB with number of hidden units equal 16 on (256×256) Lena image.

Xianghong and Yang (2008) used BPNN for image compression and developed algorithm based on improved BP. The blocks of original image are classified into three classes: background blocks, object blocks and edge blocks, considering the features of intensity change and visual discrimination. Experiments have been achieved CR (3.156:1) and PSNR (41.209) on applying this approach with number of hidden units equal 8 on (256×256) Lena image.

Finally, Veisi and Jamzad (2009) presented an adaptive method based on BPNN for image compression/decompression based on complexity level of the image by dividing image into blocks, computing the complexity of each block and then selecting one network for each block according to its complexity value. They used three complexity measure methods such as: entropy, activity and pattern-based to determine the level of complexity in image blocks. They used best-SNR approach in selecting compressor network for image blocks which chooses one of the trained networks such that results best-SNR in compressing the input image block. Experiments have been achieved CR (3.156:1) and PSNR (34.92) on applying this approach with number of hidden units equal to 8 on (256×256) Lena image.

According to literature studies, we need compression technique that leads to less storage requirements and best CR. The two most important problems that must be solved by using ANNs are the definition of network architecture, learning parameters and the weights on the connecting arcs between the neurons. In this research, we present and improve BPNN image compression/decompression system which is trained by BP.

MATERIALS AND METHODS

BPNN for image compression: In this research, a BPNN is designed for image compression and the architecture of it is developed using the idea of encoder problem. The BPNN is fed by analog gray level of image (ranged between 0 and 255) as an input and produces an appropriate compressed code at outputs of hidden layer units. In the reconstruction process, this network would produce an analog gray level image by the outputs of output layer units.

The design of BPNN for image compression system involves determining the number of network's layers. The complexity of decision regions formed by network can be increased by increasing the number of layers. BPNN with three layers (input, hidden and output layers) as shown in Fig. 1 and a suitable number of hidden layer units is a good choice because one layer of sigmoidal hidden units is sufficient to approximate any continuous function (Jagesh and Poon, 1999). It is noted practically that when BPNN architecture is designed with more than one hidden layer, the probability of occurring BP problems such as trapping in local minimum and slow convergence would increase very much than when we design BPNN with three layers only. As shown in Fig. 1, the number of input layer units (Ni) is equal to the number of output layer units (N). The number of neurons in input and output layers is governed by the dimension of image sub block (P×P).

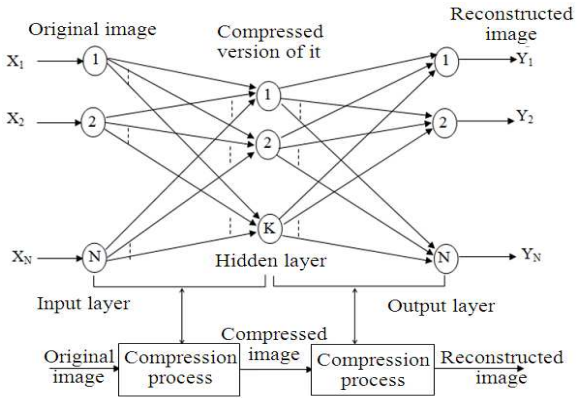

Fig. 1: BPNN image compression system

Whereas the number of hidden layer units with K units (K<N) is usually less than the number of input layer units, no rule exists to select the best number of units in the hidden layer. Therefore, the number of hidden layer units is determined empirically and it is affects on compression performance of BPNN as shown later in results.

Image normalization and segmentation must be applied to that image to simplify using it by the BPNN. The image data is represented by the pixel value function f(X,Y) where X and Y correspond to the spatial coordinates within the image with pixel values between 0 and 2^k-1, where k is the number of bits which represent each pixel in the image, usually k = 8, then the pixel values would lie within the range [0-255]. The BPNN requires inputs with real type and the sigmoid function of each BPNN neuron requires the input data to be in the range [0-1]. For this reason the image data values must be normalized. The normalization is the process of linearly transformation of image values from the range [0-255] into another range that is appropriate for BPNN requirements to obtain valuable results and to speed up the learning (Kursk *et al*., 2006). In this research, the image is linearly transformed from range [0-255] to range [0-1].

Image segmentation is the process of dividing the image into sub images, each of which is considered to be a new separate image. In this research, the image is segmented by dividing it into non overlapping blocks with equal size to simplify the learning/compressing processes. The input layer units of BPNN are represented by one-dimensional vector. Image rasterization is the process of converting each sub image from a two-dimensional block into a one-dimensional vector.

Initialization BPNN learning parameters: The weight connections of the BPNN are represented by two weight matrices. The first matrix is V which represents the weight connections between the input and hidden layer units and the second matrix is W which represents the weight connections between the hidden and output layer units. These two weight matrices must be initialized to small random numbers because the network may be saturated by large values of the weights. The learning rate value usually reflects the rate of network learning and its value (between 0.1 and 0.9) is chosen by the user of the network. Values that are very large can lead to instability in the network and unsatisfactory learning, while values that are too small can lead to excessively slow learning (Wasserman, 1989).

Preparation of BPNN training/testing set: A testing set consists of sub images that are not included in the training set and it is used to assess the BPNN performance after training. The preparation of training/testing set includes the following steps:

Step 1: Apply the segmentation process on the image to be used in learning/testing processes.
Step 2: Apply rasterization and normalization on every block segment.
Step 3: Store the results in the training set file.
Step 4: Repeat from Step 1 while there are more images to be used in training process.

Simulation of BPNN learning: a simulation program to implement BP was built to include the steps:

Step 1: Initialization of network weights, learning rate (η) and Threshold error. Set iterations to zero.
Step 2: Open file which contains the image training set.
Step 3: Total_error = zero; iterations $\rightarrow$ iterations+1.
Step 4: Get one vector and feed it to input layer.
Step 5: Initialize the target output of that vector.
Step 6: Calculate the outputs of hidden layer units.
Step 7: Calculate the outputs of output layer units.
Step 8: Calculate error (desired output - actual output) and calculate total_error $\rightarrow$ total_error + error.
Step 9: Calculate delta sigma of output layer units and adjust weights between output and hidden layer.
Step 10: Calculate delta sigma of hidden layer units and adjust weights between hidden and input layer.
Step 11: While there are more vectors, go to Step 4.
Step 12: If Threshold error >= Total_error then stop, otherwise go to Step 3.

BPNN compression process: The number of connections between each two layers in BPNN is calculated by multiplying the total number of neurons of the two layers, then adding the number of bias neurons connections of the second layer (bias connections of a layer is equal to the number of layer neurons). If there are Ni neurons in the input layer, Nh neurons in the hidden layer and No neurons in the output layer, the total number of connections is given by equation:

$$\text{Network Size(Nw)} = [(Ni \times Nh)+Nh]+[(Nh \times No)+No] \quad (1)$$

The block diagram of the BPNN image compression/decompression is shown in Fig. 2. The

compression process can be summarized by the following steps:

Step 1: Read image pixels from file and then normalize it by converting it from range [0-255] to range [0-1].
Step 2: Divide the image into non-overlapping blocks.
Step 3: Rasterizing the image blocks.
Step 4: Apply the rasterized vector into input layer units
Step 5: Compute the outputs of hidden layer units by multiplying the input vector by the weight matrix (V).
Step 6: Store the outputs of hidden layer units after denormalizing them in a compressed file.
Step 7: While there are more image vectors go to Step 4.

At the beginning of NN compressed file, there is a header block that contains information about the source image and the compression process parameters.

BPNN decompression process: The implementation of BPNN decompression includes the following steps:

Step 1: Open the compressed file.
Step 2: Take one vector from the file.
Step 3: Normalize this vector (it represents the outputs of hidden layer units).
Step 4: Compute the outputs of output layer units by multiplying outputs of hidden layer units by the weight matrix (W).
Step 5: Derasterize the outputs of output layer units to build the sub image of size P×P.

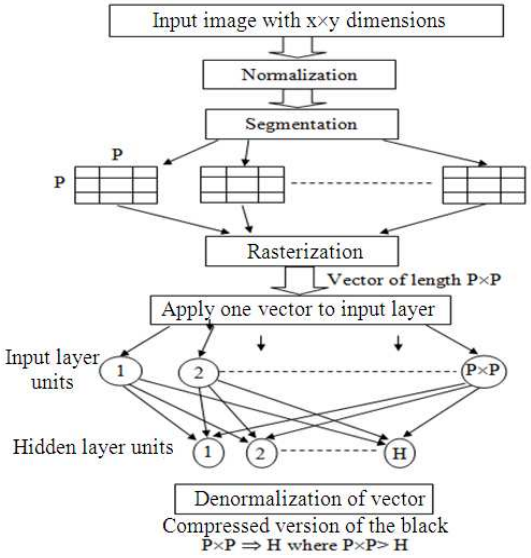

Fig. 2: BPNN image compression/decompression

Step 6: Return this sub image to its proper location
Step 7: Denormalize this block and store it in the reconstructed file.
Step 8: While there are more vectors in compressed file go to Step 2.

Improving performance of BP: The performance of BPNN can be improved by applying many approaches as follows.

Adding bias unit: A bias unit is added as a part of every BPNN layer but not the output layer. This unit has a constant value of 1 and it is connected to all units in the next layer. The weights on these connections can be trained in just the same way as other weights. The bias units provide a constant term in the weighted sum of the units in the next layer. The result is sometimes an improvement on convergence properties of the network. The bias unit also provides a threshold effect on each unit it targets. It contributes a constant term in summation of products (NETj) which is the operand in sigmoid function. Thus, we used the equation:

$$NET_j = \sum_{i=1}^{N} X_i W_{ji} + \theta_j \quad (2)$$

This is equivalent to translating the sigmoid curve to the left or to the right. In this way, the bias unit provides an adjustable threshold for each target unit. The threshold for unit j then comes from the value of Wj0 (considering θ_j as X_0 unit) the weight of interconnection for bias unit.

Using momentum variable (θ): Momentum variable improves the training time of BP and enhancing the stability of the process. It involves adding a term to the weight adjustment that is proportional to the amount of the previous weight change. Once an adjustment is made, it is "remembered" and serves to modify all subsequent weight adjustments (Wasserman, 1989). We used the following equation for weight adjustment:

$$W_{ji}^{new} = W_{ji}^{old} + [\Delta W_{ji}^{q}]^{new} \quad (3)$$

and also, we used the following equation (Wasserman, 1989):

$$[\Delta W_{ji}^{q}]^{new} = \eta \delta_i^q O_j^{q-1} + \alpha [\Delta W_{ji}^{q}]^{old} \quad (4)$$

where, α is the momentum variable in the range 0.0-1.0, but it is commonly set to around 0.9. By using momentum, the network tends to follow the bottom of

narrow gullies in error surface (if they exist) rather than crossing rapidly from side to side. If α is 0.0, then the smoothing is minimum; the entire weight adjustment comes from the newly calculated change. If α is 1.0, the new adjustment is ignored and previous one is repeated. Between 0 and 1 is a region where weight adjustments are smoothed by amount proportional to α (Wasserman, 1989).

Using beta (β): The variable Beta can be used in the sigmoidal function during BP to determine the steepness of the shape of sigmoid function. When β is used in BP training it lies in the range [0.1-1], so when β has a value of 0.1, learning is slowly converge, but when the value of $\beta = 1$, instability may occur. When β is used in sigmoid function, we used the following equation:

$$OUT = F(NET_j) = 1 / (1 + e^{-\beta \times NET_j}) \quad (5)$$

Also we used the following equation:

$$F'(NET_j) = \beta \times (OUT\ (1 - OUT)) \quad (6)$$

Changing the learning rate value (η): an attempt to produce more efficient BPNN learning, by allowing the value of η to begin at a high value and to decrease during the learning process. At the same time, allowing α to begin with low value and to increase during learning process to improve learning performance.

Introduction of random noise: One of the most important problems of BP training is the local minimum problem. Local minimum traps can be avoided with the introduction of random noise during the training process. Introducing random noise is another way of shifting the location of error function thereby permitting the descent process to escape from local minimum and continue on its downward search for a global minimum. The introduction of noise can also improve the network's ability to generalize. Introducing noise into the training patterns can be done by adding small random perturbations to the training patterns during training process.

Effects of channel errors: The fundamental problem in image transmission is the image reproduction at receiver with acceptable image quality. ANN has a strong immunity against noise and this can be examined by simulating image transmission on analog or digital channel with white noise added to transmitted image. In analog transmission, noise is added to image intensity value, while in digital transmission, noise is added to each bit of image intensity as suggested by Melesse *et al.* (2005). To simulate transmission process, we take the compressed file of an image that compressed using BPNN and then, a random noise is added to this compressed file.

The transmission on analog channel is simulated as follows: each index in the compressed image file is converted into a decimal code and then a random number of Gaussian distribution is generated for that code. If the random number is greater than 0.5, the code is incremented by one, but if it is less than -0.5 the code is decremented by one. The Probability of error of changing the code can be varied by altering the value of the variance during the generation of the Gaussian distributed numbers. Whereas the transmission on a noisy digital channel is simulated as follows: Each index in the compressed image file is converted into a binary code, then for this code a random number of Gaussian distribution is generated, if it is greater than 0.5 or less than -0.5, then one bit is inverted.

RESULTS

The CR is the degree of data reduction obtained as a result of compression process. In BPNN image compression system, the CR is defined by the ratio of the data fed to the input layer Neurons (Ni) to the data out from the hidden layer neurons (Nh). Also the CR can be computed by the equation:

$$CR = (1- (Nh/Ni)) \times 100\% \quad (7)$$

After completing the decoding process, the SNR and PSNR and NMSE (Normalized Mean Squared Error) should be calculated between the reconstructed image and original image to verify the quality of the decoded image with respect to the original one. To check the compression performance, the CR and Bit Per Pixel rate (BPP) are calculated. The CR is the amount of compression, while BPP is the number of bits required to represent each pixel value of compressed image. The better compression performance is with the highest CR, the least BPP rate and highest PSNR (Gonzales and Wintz, 1987). Simulations were conducted to evaluate the compression and generalization performances of the proposed BPNN image compression system. The efficiency of this BPNN was tested by several experiments using real world images.

BPNN size, capacity and generalization: The network capacity quantifies the learning capabilities of ANN architecture, which is a measure of the number of

training patterns that ANN can correctly identify after training has been completed. Generalization is the network's ability to correctly identify a pattern on other parts of the domain that the network did not access during the training phase (Alshoaibi *et al*., 2009). Referring to BPNN in Fig. 1, let Ni represents the number of input layer neurons, No represents the number of output layer neurons, Nh represents the number of hidden layer neurons and Nw represents the total number of weights in BPNN including bias weights, the Capacity (C) is defined as (Nw/No). Assume that, training patterns are taken from an N×N pixel image which is partitioned into P×P pixel patches for training, where P is the block dimension. Let Np represents the total number of patterns in training set:

$$Np = (N \times N) / (P \times P) \quad (8)$$

To expect good generalization, it is necessary that the Np be several times larger than the network capacity, that is, Np>C, then Generalization (G) is defined as (Np/(Nw/No)) and it is equal also (Np/C).

Effectiveness of hidden layer neurons: In suggested BPNN image compression, if we take an image of 256×256 dimension (N=256) and block dimension (P = 8), then Np was computed using Eq.8 and it is equal to 1024 patterns. Table 1 list various network architectures where Nw was computed using Eq. 1.

Effectiveness of input layer neurons: Ni depends on the block dimension (P). If the P increases this would result in increasing the Ni and this also would result in decreasing the G and decreasing the C. Table 2 shows the effect of P on G and C for images of size 256×256 using BPNN of Nh = 2. Increasing P decreases Np which results in decreasing G and C.

Table 1: Effectiveness of Nh on BPNN Size, C and G

Nh	Network size (Nw)	Capacity	Generalization
2	322	5.0	203.53
4	580	9.0	113.00
8	1096	17.0	59.80
16	2128	33.0	30.80
32	4192	65.5	15.60
64	8320	130.0	7.90

Table 2: Effectiveness of Ni on BPNN size, G and C

Block (P)	Ni	Np	Nw	G	C
2	4	16384	22	2978.90	5.50
4	16	4096	82	799.20	5.12
8	64	1024	322	203.50	5.03
16	256	256	1282	51.12	5.00

Hidden layer neurons Vs BPNN convergence time: To evaluate the relation between the Nh, convergence time and reconstructed image error, the BPNN was trained on the image (boy16.dat) of dimension 16×16, with training set of 4 blocks (block size 8×8) and with tolerance value equal 0.001 for various Nh. We fixed all other network parameter values except Nh. We recorded in Table 3 convergence time, PSNR and RMSE error for each value of Nh. From Table 3, we can see that minimum convergence time, minimum error, maximum SNR and PSNR occur when Nh = 8.

Number of input layer neurons vs. CR: The block dimension (P) plays a role in determining Ni, where Ni equals P×P. When the P is increased then Ni is increased also, this would result in increasing the CR. Table 4 shows various BPNN architectures (for various P and Nh = 2) Vs their corresponding CR and bpp.

Learning rate Vs convergence time: The convergence time required for BPNN training depends on the value of learning rate (η) because this value is used during the weights update. Values that are very large (0.9) can lead to fast learning but instability in the network and unsatisfactory learning. Values that are too small can lead to excessively slow learning. Table 5 shows the effectiveness of η value on convergence time and the number of iterations when the value of momentum variable is equal to 0.1 and for a fixed BPNN architecture. To produce more efficient BPNN learning, training the network can be started with large value of η and then decreasing it during learning.

Ability to compress untrained images: The BPNN image compression system has the ability to compress untrained image but with lower compression performance than when using trained image, because given a trained BPNN, it is not possible to guarantee a particular level of performance on unseen images.

Table 3: Effectiveness of Nh on BPNN convergence time and objective fidelity criteria

K	Nh	Convergence time (sec)	SNR (dB)	PSNR (dB)	RMSE	Iterations
1	2	31923.0	35.20	44.50	0.90	93549
2	4	145.0	41.70	51.10	0.70	2545
3	8	33.0	43.40	52.90	0.57	778
4	16	51.0	43.20	52.60	0.59	670
5	32	77.0	43.00	52.40	0.60	526
6	64	203.0	41.20	50.70	0.74	692

Table 4: Input layer neurons vs. CR and bpp

Block dimension	Ni	CR:1	BPP
2	4	2:1	4.0000
4	16	8:1	1.0000
8	64	32:1	0.2500
16	256	128:1	0.0625

Table 5: The learning rate Vs convergence time

Learning rate	Iterations	Convergence time (sec)	MSE
0.1	5449	874.00	0.579
0.2	3252	394.99	0.643
0.3	1872	122.00	0.637
0.4	1371	101.99	0.579
0.5	1207	92.99	0.612
0.6	1033	87.00	0.555
0.7	1046	86.00	0.618
0.8	745	32.00	0.628
0.9	649	27.00	0.618
1.0	594	25.99	0.602

Table 6: SNR, PSNR and RMSE for three images

Trained image or not trained	SNR (dB)	PSNR (dB)	RMSE	Figure name
Trained image	28.13	32.35	6.17	(3b)
Not trained	22.00	30.00	8.28	(3d)
Not trained	21.10	28.30	9.75	(3f)

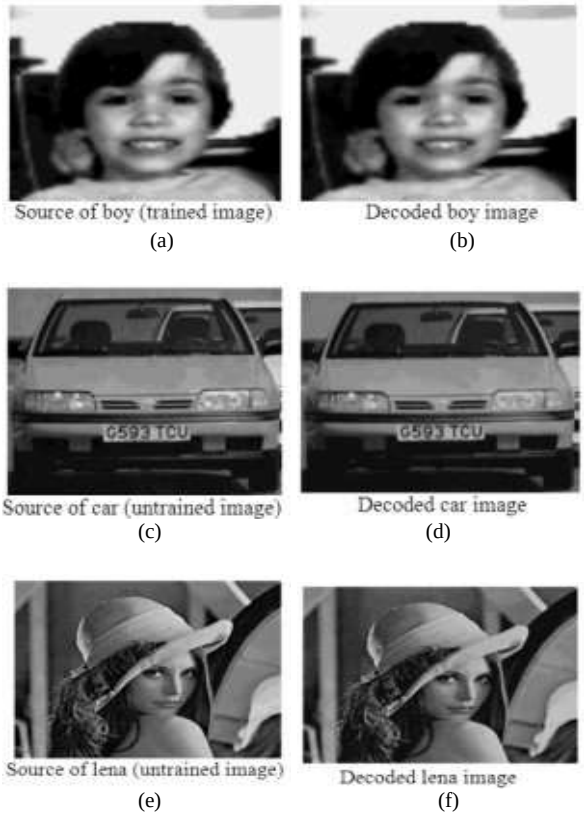

Fig. 3: BPNN ability to compress untrained images

It is in fact common for the generalization performance of NN to become sub-optimal if training is allowed to continue indefinitely as the model over fits training data (Souiyah *et al*., 2009). Figure 3a and 3b show original and decoded trained image (256×256 pixels, 256 gray levels).

Table 7: SNR and PSNR of decoded images when transmission through analog channel (Aows image)

Probability of error	SNR (dB)	PSNR (dB)
0	20.00	28.00
0.6	19.71	27.20
0.9	19.58	27.17

Table 8: SNR and PSNR of decoded images when transmission through digital channel (girl image)

Probability of error	SNR (dB)	PSNR (dB)	Figure name
0	21.0	31.0	(3b)
0.05	13.4	22.7	(3c)
0.2	10.0	18.5	(3d)

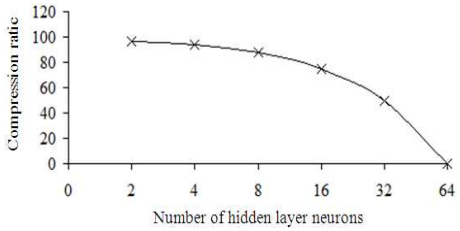

Fig. 4: Hidden layer neurons Vs CR

Figure 3c and 3d show original and decoded untrained image. Figure 3e and 3f show another example of original and decoded untrained image. Table 6 shows SNR, PSNR and RMSE of these decoded images.

Hidden layer neurons Vs CR: When the BPNN image compression system is used for image compression problem, the Nh represents the compression version of image blocks. CR is inversely proportional to Nh. Let Ni equal No. We select $Nh = 2^k$ where $1 \leq k \leq \text{Log}_2 Ni$ and the CR is calculated using the Eq. 7: $CR = (1-(Nh/Ni))\times 100$. When Ni = 64; $CR = (1-(Nh/64)) \times 100$; Fig. 4 shows the various network architectures (by changing Nh) Vs their corresponding CR. It is obvious that the Nh is responsible for determining the ratio of achieved CR. Also, by providing too many hidden layer neurons to the BPNN, the number of connection weights will increase, which increases the number of solutions available to the network. Thus, it will take a longer amount of time to find the correct set of weights, or only a local solution will be found.

BPNN for removing channel errors: When BPNN image compression system is used for image compression, it has the ability to remove errors that have occurred during compressed image transmission through analog or digital channel. This ability comes from the BP that is used to train the BPNN.

The decoded images (BMP image named Aows with 256×256 pixels and 256 gray levels) are transmitted respectively through a very noisy analog channel with probabilities of error p = 0.6 and p = 0.9 respectively. Table 7 shows SNR and PSNR of these decoded images with respect to their probabilities of errors during transmission through noisy analog channel.

Table 8 shows the SNR and PSNR of the decoded Girl image with 256×256 pixels and 256 gray levels with respect to their probabilities of errors when transmission through a noisy digital channel (when the BPNN image compression system is used).

CONCLUSION

In this research, a simulation program for BP was developed on a single processor computer. This leads to the problem of long execution times to train an image. Many methods have been carried out to improve the speed of convergence. All of these methods are computationally complex in nature and applied only to limited patterns. This research has successfully applied the BP for training the BPNN image compression system. But this BP has many problems. We suggested improving the BP for a better convergence, CR and PSNR by considering the drawbacks of BP. These drawbacks can be avoided by: (1) monitoring the total error value during the training process and changing the values of learning rate (η) and momentum variable (α) depending on this error; (2) by using the beta term ($\beta\alpha$) in the sigmoid function and finally (3) by adding small random noise to BPNN weights.

The performance of BPNN image compression system has been tested in various types of images. Experiments were conducted by varying the sub-image size namely 4×4, 8×8, 16×16 and 32×32. The BPNN was also tested by varying the number of neurons in the hidden layer, resulting in CRs ranging from 4 to 64. The PSNR values for various combinations of the above were obtained. The Quality of restored image (PSNR, CR and the speed of convergence of the BPNN) were compared for both conditions. From the results, one can clearly see that the performance of the designed BPNN image compression/decompression system can be increased; this may be accomplished by modifying the network itself.

Practically, we can note that the BPNN has the ability to enhance any noisy compressed image that had been corrupted during compressed image transmission through a noisy digital or analog channel. Practically, we can note that the BPNN has the ability to compress untrained images but not in the same performance of the trained images. This can be done especially when using small number of image block dimension (P).

REFERENCES

Alshoaibi, A.M., A.K. Ariffin and M.N. ALmaghribi, 2009. Development of efficient finite element software of crack propagation simulation using adaptive mesh strategy. Am. J. Applied Sci., 6: 661-666. DOI: 10.3844/.2009.661.666

Durai S.A. and E.A. Saro, 2006. Image compression with back-propagation neural network using cumulative distribution function. World Acad. Sci. Eng. Technol., 17: 60-64. http://www.waset.org/journals/waset/v17/v17-12.pdf

Souiyah, M., A. Muchtar, A. Alshoaibi and A.K. Ariffin, 2009. Finite element analysis of the crack propagation for solid materials. Am. J. Applied Sci., 6: 1396-1402. DOI: 10.3844/.2009.1396.1402

Gonzales R.C. and P. Wintz, 1987. Digital Image Processing. 2nd Edn., Addison-Wesley, Boston, MA., USA., ISBN: 10: 0201110261, pp: 503.

Jagesh, V.S. and C.S. Poon, 1999. Linear independence of internal representations in multi layer perceptrons. IEEE Trans. Neural Networks, 10: 10-18. DOI: 10.1109/72.737489

Melesse, A.M. and R.S. Hanley, 2005. Energy and carbon flux coupling: multi-ecosystem comparisons using artificial neural network. Am. J. Applied Sci., 2: 491-495. DOI: 10.3844/.2005.491.495

Khalil, R.A., 2007. Digital image compression enhancement using bipolar backpropagation neural networks. Al-Rafidain Eng., 15: 40-52. http://alrafidain.engineering-coll-mosul.com/files/4ab25.pdf

Kursk, S.M., R.J. Rasras and D. Skopin, 2006. The artificial neural network based approach for mortality structure analysis. Am. J. Applied Sci., 3: 1698-1702. DOI: 10.3844/.2006.1698.1702

Roy, S.B., K. Kayal and J. Sil, 2005. Edge preserving image compression technique using adaptive feed forward neural network. Proceeding of the 9th IASTED International Conference on internet and Multimedia Systems and Applications, Feb. 21-23, DBXLAB, Grindelwald, Switzerland, pp: 467-471. http://dbxlab.uta.edu/dbxlab/EURO.pdf

Veisi, H. and M. Jamzad, 2009. A complexity-based approach in image compression using neural networks. Int. J. Sign. Process., 5: 82-92. http://www.akademik.unsri.ac.id/download/journal/files/waset/v5-2-11-5.pdf

Wasserman, P.D., 1989. Neural Computing: Theory and Practice. Coriolis Group, New York, USA., ISBN: 10: 0442207433, pp: 230.

Xianghong, T. and L. Yang, 2008. An image compressing algorithm based on classified blocks with BP neural networks. Proceeding of the International Conference on Computer Science and Software Engineering, Dec. 12-14, IEEE Computer Society, Wuhan, Hubei, pp: 819-822. DOI: 10.1109/CSSE.2008.1357