Fast Backpropagation Neural Network for VQ-Image Compression

Basil S. Mahmood
Dept. of engineering Computers
College of engineering

Omaima N. AL-Allaf
Dept. of Computers
College of Computer and Mathematical Science

University of Mosul

Received on: 20/8/2002 Accepted on : 4/1/2003

الملخص

إن مشكلة التعامل مع الصور الرقمية هي كمية البيانات الهائلة اللازمة لعملية التخزين أو النقل، الأمر الذي أدى إلى ابتكار طرائق مختلفة لكبس الصور و في الوقت نفسه الإبقاء على وضوح جيد للصور . ومن التقنيات الجذابة في هذا المجال هو استخدام الشبكات العصبية التي تمتاز بقابليتها العالية في الحسابات السريعة التي تعتمد على استخدام المعالجات المتوازية.

في هذا البحث تم استخدام شبكة عصبية بثلاث طبقات لكبس الصور باستخدام طريقة التكميم الاتجاهي (VQ). لقد ثبت لدينا بأن النتائج الخارجة من الطبقة الوسطية (أو المخفية) للشبكة تمثل كتاب الشفرة (code book) المستخدم في التكميم الأتجاهي. ولذلك فهي طريقة جديدة لتوليد كتاب الشفرة. كما أن خوارزمية الانتشار العكسي السريعة المبينة و المطبقة على الشبكة المصممة أثبتت كفاءتها بالحصول على نفس نسبة الإشارة إلى الضوضاء و نفس نسبة الكبس التي تنتجها طريقة الانتشار العكسي الاعتيادية و لكن بزيادة سرعة تنفيذ مقدارها ٥٠ ضعفاً.

Abstract

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 very attractive

in image processing where high computational performance and parallel architectures are required.

In this work, a three layered backpropagation neural network (BPNN) is designed to compress images using vector quantization technique(VQ).The results coming out from the hidden layer represent the codebook used in vector quantization, therefore this is a new method to generate VQ-codebook. Fast algorithm for backpropagation called

(FBP) is built and tested on the designed BPNN. Results show that for the same compression ratio and signal to noise ratio as compared with the ordinary backpropagation algorithm, FBP can speed up the neural system by more than 50. This system is used for both compression/decompression of any image. The fast backpropagation (FBP) neural network algorithm was used for training the designed BPNN. The efficiency of the designed BPNN comes from reducing the chance of error occurring during the compressed image transmission through analog channel (BPNN can be used for enhancing any noisy compressed image that had already been corrupted during transmission through analog channel). The simulation of the BPNN image compression system is performed using the Borland C^{++} Ver 3.5 programming language. The compression system has been applied on the well known images such as Lena, Carena, and Car images, and also deals with BMP graphic format images.

Keywords: Artificial Neural Networks, Image Compression, Backpropagation algorithm (BP), Fast Backpropagation algorithm (FBP), BPNN, FBPNN, VQ.

1. Introduction

Artificial neural network models have been studied for many years in the hope of achieving human-like performance in the fields of difficult real world problems such as pattern recognition, speech and image recognition, and so on. These models are composed of many interconnected non-linear computational elements that operate in parallel and connected together via weights much the same way as the brain's neurons are. These neurons are able to associate and generalize without rules [1 ; 2 ; 3].

The digital representations of images usually require a very large number of bits. In many applications, it is important to consider techniques for representing an image, or the information contained in the image, with fewer bits while maintaining an acceptable fidelity of image quality. In the terminology of information theory this is referred to as image encoding, or as image compression [4]. The benefity of image compression would result in reducing storage requirements, cost, time, the probability of transmission error occurring since fewer characters are transmitted when data is compressed while the probability of an error occurring remains constant, and compression algorithm may provide a level of security illicit monitoring [5].

Artificial neural networks (ANNs) are increasingly being examined and considered as possible solutions for many problems and applications where high computation rates are required. The main advantages of an ANN are its concurrent parallel processing capability, and its capability to extract a desired transformation function from instances. It is pointed out that the following three capabilities are important for neural network models to function as learning machines:

1- Data compression capability of mapping a set of original data to feature a space of reduced dimensionality.
2- Generalization capability of compressing unlearned data as well as learned data.

3- Neural network for image compression can limit the effects of channel errors and circuit faults [6].

Transmission of Compressed Image

The fundamental problem in the transmission of images is the reproduction of image at the receiver with an image accuracy and quality acceptable to the viewer. For economy, this must be done using the minimum possible number of transmission bits. Many transform coders are used for coding pictures at rates of less than 1bit/pixel in order to produce more subjectively pleasing reconstructed images (i.e. in order to achieve error protection), this results in reducing the presence of channel errors but does not remove it totally. Although this method of error protection gives a much improved error protection, its main disadvantage is that it requires considerable data storage and significant search times when decoding the data [7].

2. Artificial Neural Networks

Artificial neural network models or simply "neural networks" go by many names such as "connectionist models", "parallel distributed processing models", and "neuromorphic systems". Whatever the name, all these models attempt to achieve good performance via dense interconnection of simple computational elements [1]. Instead of programming a neural network, you "teach it" to give acceptable answers. You input known information, assign weighted values to the connections within architecture, and run the network (which adjusts those weights by using several criteria) over and over until the output becomes

satisfactorily accurate. A weight matrix of interconnections allows neural networks to learn and remember.

As a result of the way they work, even when you enter new information that is not stored in the network, they can still provide adequate responses. When they work correctly, neural networks provide some major benefits, such as the ability to take incomplete data and produce approximate results. Their parallelism, speed, and trainability make them fault-tolerant, as

well as fast and efficient for handling large amounts of data [2].

While a digital computer's memory is measured in bytes, a neural network's memory is judged by interconnections. Likewise, while the speed of a digital computer is expressed in instruction per second, the neural network's speed is measured in interconnections per second. There are many types of neural networks, but all have three things in common: characteristics of distributed processing elements, the connections between them (network topology), and the learning rule (training method). These three aspects together constitute the neural network paradigm [3].

1.1 Artificial Neural Network Training

Learning is the phase in a neural network when new data is introduced into the network causing the network weights to be adjusted. In supervised learning, the network has some information (teacher) present during the learning to tell what the correct answer should be. The network then has a way to find whether or not its input was correct and knows how to apply its particular learning law to adjust itself. In contrast, unsupervised learning means that the network has no such knowledge of the correct answer (teacher) and thus cannot know exactly what the correct output should be [8][9].

1.2 Backpropagation Neural Network (BPNN)

Each neural network that has been trained by using backpropagation algorithm is already called backpropagation neural network (BPNN). The BPNN is currently the most widely used neural architecture. The primary reason for this is that: it is easy to implement, and the networks using backpropagation algorithm learn complicated multi-dimensional mappings more easily than when using other algorithms. We can say that the most commonly used training method in multi layered neural network is backpropagation algorithm. The backpropagation algorithm has been tested successfully for different kinds of tasks. The BPNN is a hierarchical feed-forward network system

consisting of three or more fully interconnected layers of processing units (including the input layer). Neurons are not connected to other neurons in the same layer (layered network). The backpropagation learning method can be applied to any multilayer network that uses differentiable activation function and supervised training. The backpropagation learning rule is generalized from Widrow & Hoff rule for multi layer networks [10]. Like the delta rule, it is an optimization procedure based on gradient descent that adjusts weights to reduce the system error or cost function. The name backpropagation arises from the method in which corrections are made to the weights [8].

The neuron is used as the fundamental building block for BPNN. A set of inputs is applied, either from the outside or from a previous layer. Each of these is multiplied by its associated weight, and the products are summed. The summation of products is termed NET_j and must be calculated for each neuron in the network [8].

$$NET_j = \sum_{i=1}^{N} X_i W_{ji} \qquad (1)$$

Where j is the jth neuron in the layer, Xi is the input vector, Wji is the associated weight, and N is the input vector length. After NET_j is calculated, an activation function (sigmoid function) F is applied to modify it, thereby producing the signal OUT.

$$OUT = F(NET_j) \qquad (2)$$

$$OUT = 1 / (1 + e^{-NET_j}) \qquad (3)$$

The sigmoid function (or squashing function because it compresses the range of NET_j so that OUT lies between zero and one) is desirable in that it has simple derivative, a fact we use in implementing the backpropagation algorithm; where:

$$F'(NET_j) = OUT\ (1 - OUT) \qquad (4)$$

1.3 Fast Backpropagation Algorithm (FBP)

The popularity of multi-layered neural networks motivated several researchers to focus on heuristic technique for

accelerating the error backpropagation algorithm by adapting the learning rate (η) during training or by using various other heuristics to improve the convergence of the algorithm. As a result, several new algorithms have been proposed for training neural networks with various architectures that converge faster than BP. The development of fast learning algorithms for feed-forward neural networks presented here is based on the minimization of an objective function after the initial adaption cycles, this minimization can be achieved by reducing lemda (λ) from unity to zero during the training of the network [11,12].

The BPNN trained by fast backpropagation algorithm (FBP) follows the same steps of BPNN trained by ordinary BP. The FBP algorithm differs from the ordinary BP algorithm in the development of the alternative training criterion, this criterion indicates that λ must change from 1 to 0 in the course of the training (i.e. as the total error decreases, λ should approach zero). This suggests that the value of λ should be determined in each adaption cycle from the total error at that point, according to some suitable rule, i.e., $\lambda = \lambda(E)$ where E is the error of the network. The above discussion indicates that $\lambda \approx 1$ when $E >> 1$. In this case, for any positive integer n, $1/E^n$ approaches zero and, therefore, $\exp(-1/E^n) \approx 1$. On the other hand, when $E << 1$, $1/E^n$ becomes very large and, therefore, $\exp(-1/E^n) \approx 0$. As a result, a suitable rule for the reduction of λ from one to zero is the following

$$\lambda = \lambda(E) = \exp(-\mu / E^n) \qquad (5)$$

where μ is a positive real number and n is a positive integer. The smaller the integer n, the faster the reduction of λ when $E >> 1$. It has been experimentally verified that if λ is much smaller than unity during the initial adaption cycles the algorithm may be trapped in a local minimum. This is an indication that n should be greater than 1. Thus, λ is determined during the training of any network according to the following rule [11]:

$$\lambda = \lambda(E) = \exp(-\mu/E^2) \qquad (6)$$

In training the BPNN by using fast backpropagation algorithm, all the hidden layer neurons and all the output layer neurons use the hyperbolic tangent function instead of sigmoid function used by all the BPNN neurons when using BP training algorithm. So the equation (3) is modified for hyperbolic tangent function as follows:

and the derivative of this function is as follows:

$$F'(NET_j) = (1- (F(NET_j)^2)) \quad (8)$$

So that $F(NET_j)$ lies between –1 and +1. Figure (1) shows the hyperbolic tangent function and its derivative.

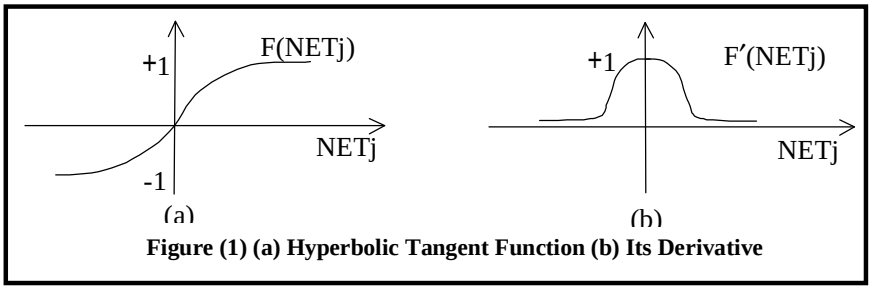

Figure (1) (a) Hyperbolic Tangent Function (b) Its Derivative

3. Typical Image Compression System

A typical encoding system consists of three processes, as shown in figure (2). The first stage transforms (maps) the input image data from the pixel domain to another domain that is useful for reducing the number of bits that must be used to encode the data (where the information can be encoded more efficiently). In the second stage, the quantizer rounds off the output of the mapper, and the coder assigns a codeword (such as a Huffman code) to the quantizer output. We consider that the X×Y image is divided into subimages of size P×P and the subimages are encoded separately. We use the non-adaptive transform encoding where the same coder is used for all subimages [13;14].

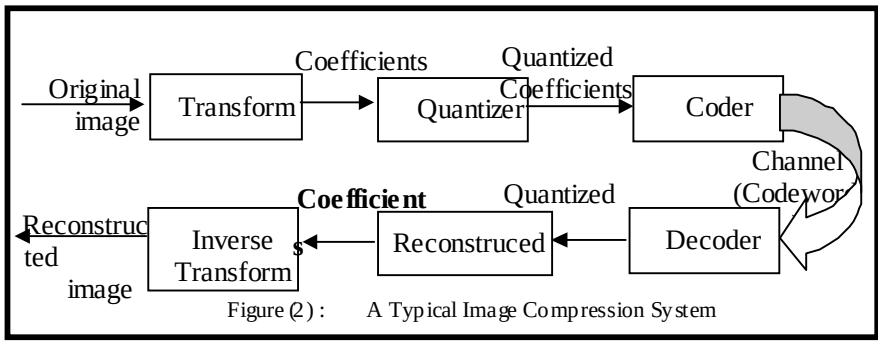

Figure (2) : A Typical Image Compression System

4. Backpropagation (BP&FBP) Neural Network for Image Compression

Image compression is a type of encoder problem. That is, a network is given the problem of performing an identity mapping over some set of inputs. The network is constructed to perform this mapping through a narrow channel of the network, forcing it to develop an efficient encoding in that channel. There are two interesting aspects to this: (a) the network is developing a compact representation of its "enviroment"; and (b) although the algorithms used were developed as supervised learning schemes, this problem can be regarded as unsupervised since the output is the same as the input –the system self- organizes to encode the enviroment [13]. A technique using backpropagation neural network is designed for the process of image compression. The architecture of the BPNN used for image compression is developed using the idea of the encoder problem. The BPNN used for encoding process is fed by the binary values at the input layer units and produces binary values at the output layer units. In contrast, the BPNN used for image compression is fed by the analog gray level of image as an input (values in the range between 0 and 255) and produces an appropriate compressed code at the outputs of the hidden layer units, and then in the reconstruction process, this network would produce an analog gray level image by the outputs of output layer units.

The image compression neural network consists of three layers of artificial neurons, arranged in planes to form layers as shown in figure (3). The input and output layers have N units each, and

an intermediate layer with K units (K<N) that is the channel between the other two.

5. Image Processing Performance Evaluation

In some image transmission systems some errors in the reconstructed image can be tolerated. In this case a fidelity criterion can be used as a measure of system quality. After completing the decoding process, the SNR (Signal to Noise Ratio) and PSNR (Peak Signal to Noise Ratio) and NMSE (Normalized Mean Squared Error) should be calculated between the reconstructed image and the original image to verify the quality of the decoded image with respect to the original one. The SNR of the decoded image is given by [4]:-

$$SNR_{dB} = 10\,Log_{10}\left[\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\hat{x}^{2}(i,j)}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[x(i,j)-\hat{x}(i,j)]^{2}}\right] \qquad (9)$$

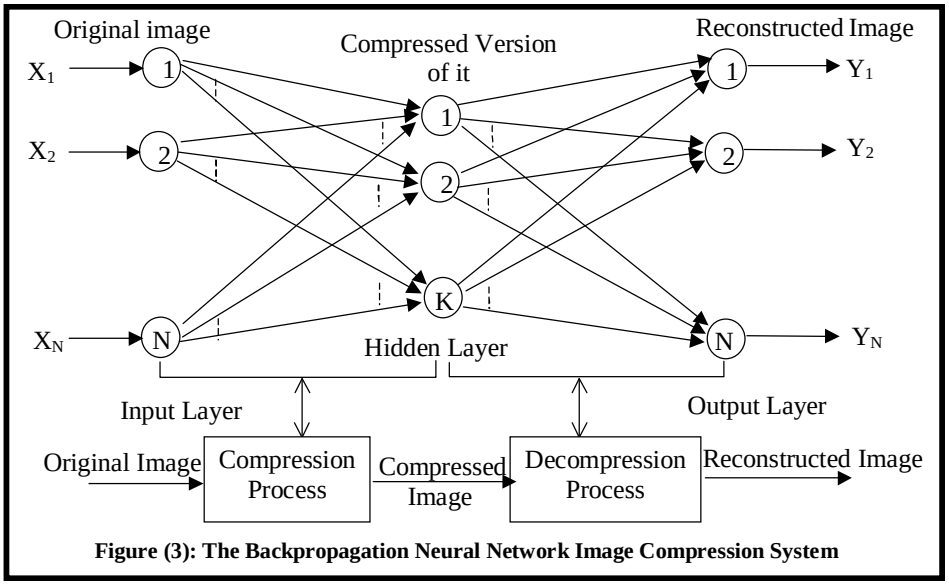

Figure (3): The Backpropagation Neural Network Image Compression System

For an image of size N×N, whereas the PSNR is given as:

$$PSNR_{dB} = 10\,Log_{10}\left[\frac{\left[\text{Peak value of } \hat{x}(i,j)\right]^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[x(i,j)-\hat{x}(i,j)]^2 / N^2}\right] \quad (10)$$

and the NMSE is given by [4]:-

$$NMSE = \left[\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[x(i,j)-\hat{x}(i,j)]^2}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[x^2(i,j)]^2}\right]\times 100 \quad (11)$$

The peak value of $\hat{X}(i,j)$ is the total dynamic range of the output image, and N is the source image dimension.

To check the compression performance, the values (C/R) compression ratio and bpp rate (bit per pixel rate) are calculated. The compression ratio is the amount of compression, while the bpp rate is the number of bits required to represent each pixel value of the compressed image.

$$\text{bpp rate} = \frac{\text{compressed image file size in bits}}{\text{image size in pixels}} \text{ bits/pixel (bpp)} \quad (12)$$

$$C/R = \frac{\text{source image file size}}{\text{compressed image file size}} \quad (13)$$

6. Implementation

6.1 Fast Backpropagation Neural Network Learning Algorithm

During the training of the BPNN using FBP algorithm, all the hidden layer neurons and output layer neurons use the hyperbolic tangent function instead of sigmoid function that is used by all neurons in BPNN training using BP algorithm. The hyperbolic tangent function has a range of [-1 to +1], thus, image normalization is done by transforming the coordinates [0-255] into the coordinates [-1 to +1]. Training using fast

backpropagation algorithm includes the following steps:

1- Initialize all the network weights W to small random values within the range [-λ , +ë]. Set the value of λ to 1. Determination of μ value, usually μ lies in the range [0.1-10]. Setting η to very small value, usually between [0.01-0.1]. The FBP algorithm requires setting the value of β greater than 1 (i.e. usually β is in the range [2-6]).

2- Select the next training pair from the training set $[X^P, t^P]$ (input, target),

and compute in a forward direction the output values for each unit j of each layer q, thus

$$O_j^q = f\left(\sum_i O_i^{q-1} W_{ji}^q\right) \qquad (14)$$

where O_j^q is the output of jth unit in layer q. Note that the inputs to layer one are indexed with superscript 0, and hence, $O_j^0 = X_j$.

3- Calculate the error between the actual output of the network (O) and the desired output (t) (the target vector from the training pair), and then, use the values O_j^Q computed by the final layer units and the corresponding target values t_j^p to compute the delta quantities (δ) for all j using pattern P.

$$\delta_j^Q = \lambda (O_j^Q - t_j^p) + (1-\lambda)\tanh(\beta (O_j^Q - t_j^P)) \qquad \textbf{(15)}$$

where t_j is the target value for unit j,

O_j is the output value for unit j, and Q represent the final layer.

NET_j is the weighted sum of inputs to unit j (sigmoid function).

$f'(NET_j)$ is the derivative of the sigmoid function $f(NET_j)$, and

4- Compute the deltas (δ) for each of the proceeding layers by

backpropagating the errors using

$$\delta_j^{q-1} = f'(NET_j^{q-1}) \left[\sum_{i=1}^{M} \delta_i^q W_{ji}^q \right] \qquad (16)$$

for all j in each of the layers q = Q, Q-1, …….,2. The M value would represent the number of neurons in layer q.

5- Update all weights W_{ji} using

$$W_{ji}^{new} = W_{ji}^{old} + \Delta W_{ji}^q \qquad (17)$$

for each layer q, where $\Delta W_{ji}^q = \eta \delta_i^q O_j^{q-1}$ (18)

where η is the learning rate and it is in the range [0.01..0.1].

6- Update λ so that $\lambda = \exp(-\mu / E^2)$ (19)

7- Return to step 2 and repeat for each pattern P in the training set until the total error has reached an acceptable level.

The total error in the performance of the network with a particular set of weights can be computed by comparing the actual output (O), and the desired output (t), for every training pattern. The total error (Etot) is defined by:

$$E_{tot} = \sum_c Ec \qquad (20)$$

where c is an index overall of the input-output pairs in the training set, and Ec (the local error) is defined as follows:

$$Ec = \frac{1}{2} \sum_{j=1} (O_j^Q - t_j^p)^2 \qquad (21)$$

where j is an index over output units (within a training pair).

At the end, we can say that the patterns in the training set are presented to the network repeatedly. Each training iteration consists of presenting each input/output pattern pair once. When all the patterns in the training set have been presented, the training iteration is completed, and the next training iteration is begun. A typical backpropagation example might entail hundreds or thousands of training iterations.

6.2 BPNN for Reducing the Effect of Channel Errors

Neural network has a strong immunity against noise. This phenomenon can be examined by simulating image transmission on analog channel with white noise added to the transmitted image. In analog transmission, noise is added to the image intensity value as suggested by Kangas [15]. However, in order to simulate the transmission process, we take the compressed file of an image that had been compressed using BPNN image compression system, and then, a random noise is added to the 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 (P) of changing the code can be varied by altering the value of the variance during the generation of the Gaussian distributed numbers.

7. Results

7.1 The Effectiveness of Number of Input layer Neurons on BPNN Compression Ratio

The image block dimension (P) plays a role in determining number of input layer neurons, where the number of input layer neurons (Ni) equals P×P. When the dimension of image block is increased then the number of input layer neurons is increased also, this would result in increasing the compression ratio of BPNN image compression system. Table (1) shows various BPNN architectures (for various image block dimension) versus their corresponding compression ratio, number of hidden layer neurons is fixed (Nh=2).

Table (1): The Effectiveness of Input Layer Neurons on Compression Ratio and Bit Rate (bpp)

Block Dimension	Ni	Compression Ratio	Bit Rate (bpp)
2	4	2 : 1	4
4	16	8 : 1	1
8	64	32 : 1	0.25
16	256	128 : 1	0.0625

7.2 The Effectiveness of Hidden Layer Neurons on the Compression Ratio

When the BPNN image compression system is used for image compression problem, the number of hidden layer neurons represents the compression version of image blocks. Compression ratio is inversely proportional to the hidden layer neurons. Let Ni represent the number of neurons in the input layer which is equal to the number of neurons of output layer, Nh is the number of hidden layer neurons. We select $Nh = 2^k$; $1 \leq k \leq Log_2 Ni$, the compression ratio is calculated using the formula: $CR = (1- (Nh / Ni)) \times 100$;

When $Ni = 64$; $CR = (1- (Nh / 64)) \times 100$;

Figure (4) shows the various network architectures (by changing the hidden layer neurons) versus their corresponding compression ratio.

Figure (4) : The Effectiveness of Hidden Layer Neurons on Compression Ratio (C/R)

7.3 BPNN Ability to Compress Untrained Images

The BPNN image compression system that has been trained by backpropagation neural network algorithm has the ability to compress untrained image (i.e. to compress any image that is not used in the training process) but with lower compression performance than when using trained image, because given a trained neural network, it is not possible to guarantee a particular level of performance on unseen

input/output pairs (i.e. generalization performance). It is in fact common for the generalization performance of a neural network to become sub-optimal if training is allowed to continue

trained image (256×256 pixels, 256 gray levels). Figures (5-c), and (5-d) show the original and decoded untrained image. Figures (5-e), and (5-f) show another example of original and decoded untrained image. Table (2) shows the SNR, PSNR, and RMSE of these decoded images with respect to their figures.

Table (2) : SNR, PSNR, and RMSE for three images

Trained Image or not Trained	SNR (dB)	PSNR (dB)	RMSE	C/R	Figure Name
Trained Image	28.13	32.35	6.17	8 : 1	(5-b)
Not Trained	22.0	30.0	8.28	8 : 1	(5-d)
Not Trained	21.1	28.3	9.75	8 : 1	(5-f)

7.4 BPNN for Removing Channel Errors

When the 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 channel (i.e. have the ability to enhance the noisy compressed image). This ability comes from the backpropagation training algorithm that is used to train the BPNN. Figures (6-a), and (6-b) respectively represent the source of the BMP image named Aows with (256×256 pixels, 256 gray levels) and the decoded image when the BPNN image compression system is used for encoding this image without adding noise. The decoded images shown in figures (6-c), and (6-d) are transmitted respectively through a very noisy analog channel with probabilities of error p=0.6, and p=0.9 respectively. Table (3) shows the SNR, and PSNR of these decoded images with respect to their probabilities of errors during transmission through noisy analog channel.

Table (3): SNR and PSNR of decoded images when transmission through analog channel

Probability (P) Of an Error	SNR (dB)	PSNR (dB)	C/R	Figure Name
0	20.0	28.0	32 : 1	(6-b)
0.6	19.71	27.20	32 : 1	(6-c)
0.9	19.58	27.17	32 : 1	(6-d)

7.5 Fast Backpropagation Algorithm Compared with Backpropagation Algorithm

In this section, the fast backpropagation algorithm is compared with the backpropagation on the basis of the number of adaption cycles required by each algorithm to achieve the same small value of the total error. The two algorithms are used individually to train the same BPNN architecture (input layer neurons = 64, hidden layer neurons = 8, output layer neurons = 64), and with the same initial set of synaptic weights that were provided by a random number generator producing numbers between –0.5 and +0.5. The value of momentum variable equals zero, and value of threshold convergence error equals 0.001. The backpropagation training algorithm can be used with different values of β and η, thus large values of η can lead to rapid convergence, and low value of η can lead to slow convergence. Large value of η with β=1 can lead to instability learning. For training using fast backpropagation algorithm, the value of β is used greater than 1 for achieving rapid convergence, at the same time we use small value of η [0.01 – 0.1] to avoid instability in the learning. Table (4) shows the number of iterations required in both algorithms for different values of η and β with the same value of μ =0.1.

This table also shows the objective fidelity criterion for each case

Table (4): The Values of Learning Rate and Beta Versus Number of Iterations Required for Both BP and FBP algorithms

Algorithm	Iterations	SNR (dB)	PSNR (dB)	C/R
EBP (η=0.07, β=1)	8963	42.50	51.95	32 : 1
FBP (η=0.07, β=2.5, μ=0.1)	130	41.40	50.84	32 : 1
EBP (η=0.05, β=1)	13291	43.60	53.05	32 : 1
FBP (η=0.05, β=3.0, μ=0.1)	271	40.89	50.31	32 : 1
EBP (η=0.03, β=1)	19181	43.06	52.51	32 : 1
FBP (η=0.03, β=6.0, μ=0.1)	327	41.95	51.38	32 : 1

From the table (4), we can say that, as the learning rate value decreases and the maximum allowable value of β increases, the difference in convergence becomes even more significant.

(a) Source of Boy (Trained Image)

(b) Decoded Boy Image

(c) Source of Car (Untrained Image)

(d) Decoded Car

(e) Source of Lena (Untrained

(f) Decoded Lena Image

Figure (5) : BPNN Ability to Compress Untrained Images

8. Conclusions

From the results, one can clearly see that the performance (objective fidelity criteria) of the simulated BPNN image compression system can be increased, this may be accomplished by modifying the network itself (i.e. by changing the number of input layer neurons and the number of hidden layer neurons). The BP (FBP) algorithm used for BPNN training is also called "self-supervised" because the output which corresponds to the certain input is not provided by an external teacher. The BPNN is essentially an N to N encoder with analog inputs and outputs. The BP algorithm is a parallel algorithm, in this work, a simulation program for BP algorithm was developed on a single processor computer rather than on a parallel computer (with multi processors). This leads to the problem of long execution times to train an image or image sequence. The FBP algorithm can be used to speed up the convergence. The FBP algorithm has the same steps of BP algorithm but it differs in many points.

Also, 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 (i.e. the images that were not seen during the training process), but not in the same performance of the trained images. This can be done especially when using small number of image block dimension (P) (i.e. small number of input layer units).

(a) Source of Aows (BMP) Image

(b) Decoded Image (P = 0) (Noiseless Channel)

(c) Decoded Image (P=0.6) (Very Noisy Channel)

(d) Decoded Image (P=0.9) (Very Noisy Channel)

Figure (6): The Source and Decoded Aows Images after Transmission Through a Noisy Analog Channel with Different Probabilities of Errors (P)(This is when the Source Image was Compressed using BPNNICS)

REFERENCES

[1] Lippmann, R. P. An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, 4 (2) ,(April 1987),pp. 4-22.

[2] Obermeier, K. K. and Barron, J. J. Time to Get Fired Up, BYTE, (August 1989) ,pp.217-224.

[3] Jeannette "Jet" Lawrence Untangling Neural Nets, Dr. Dobb's Journal, (April 1990) , pp.38-44.

[4] Gonzales, R. C. and Wintz, P. Digital Image Processing ,Addison Wesley Publishing Company, (1987).

[5] Morris , G. and Huifang, S. Image Sequence Coding Using Vector Quantization, IEEE Transactions on Communications, Vol.Com-34,No.7,(July1986),pp.703-710.

[6] Carrato, S. and Marsi, S. Adaptive Structure Based on Neural Networks for Subband-Filtered Image Compression, Neural Network World, Vol.1, (1993) ,pp.25-40.

[7] Brewster, R. L. and Dodgson, T. E. Effect of Channel Errors on Source Coded Image Data and the Provision of Adequate Protection of Transmission Bits, IEE Proceedings, Vol.135, Pt.F, No.6, (December 1988), pp.528-538.

[8] Wasserman, P. D. Neural Computing*:* Theory and Practice, VAN NOSTRAND REINHOLD, NewYork, (1989).

[9] Chitra, S. P. Use Neural Networks for Problem Solving, Chemical Engineering Progress, (1993) , pp.44-52.

[10] Widrow, B. and Hoff, M.E. Adaptive Switching Circuits, in IRE Western Electronic Show and Convention, Part 4, (1960), pp.96-104. (Cited by Neural Computing*:* Theory and Practice).

[11] Karayiannis, N. B. and Venetsanopoulos, A. N. Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications, Kluwer Academic Publishers, London, (1993).

[12] Patterson, D. W. Artificial Neural Networks: Theory and Applications, Prentice Hall, Signapore, (1995).

[13] Gregory, K. W. *The JPEG Still Picture Compression Standard*, Communications of the ACM 34 (4) , (April 1991) ,pp. 31-44.

[14] Cottrell, G. W., Munro, P., and Zipser, D. *Image Compression by Backpropagation: An Example of Extensional Programming*, Models of Cognition, Shorkey (Ed.) Norwood : Ablex, (1989) , pp. 208-240.

[15] Kangas, J. *Self-Organizing Maps in Error Tolerant Transmission of Vector Quantized Images*, Technical Report A21, Helsinki Univ. of Tech., Lab. of Comp. and Inf. Science, SF-02150 Espo. Firland, (1993).