Self-Organizing Map for Image Compression: A Study of Optimal Performance

إختزال الصور بالشبكات العصبية ذاتية التشكيل : دراسة للوصول للأداء الأمثل

H. H. Soliman, M. M. Awadalla, F. E. Z. Abou Chadi
Faculty of Engineering, Mansoura University

ملخص: لقد أصبح إختزال الصور ذا أهمية متزايدة نتيجة للتطبيقات العديدة مثل مؤتمرات الفيديو و التعليم عن بعد و التصوير الطبي و غيرها. وفى هذا البحث تمت دراسة إستخدام شبكة كوهنين العصبية ذاتية التشكيل مع تطبيق خوارزم معدل للتعليم للتغلب بنسبة كبيرة على مشاكل تصميم و إيجاد الشفرات المطلوبة. و قدم البحث دراسة تفصيلية لاختيار معاملات الشبكة وصولا إلى الأداء الأمثل لعملية التعليم فى حالة تطبيقات إختزال الصور. و قد استخدمت ١٤ صورة ملونة فى عملية التعليم بالإضافة لمجموعة أخرى للاختبار. و قد تمت مقارنة جودة الصور المسترجعة بعد الاختزال مع الصور الأصلية باستخدام معامل "القيمة القصوى لنسبة الإشارة إلى الشوشرة".

Abstract

Image and video compression is becoming an increasingly important area of investigation, with numerous applications to video conferencing, interactive education, home entertainment, and potential application to earth observation, medical imaging, digital libraries and many other areas. In this paper the Kohenen's self-organizing feature map (KSOFM) neural network and a modified frequency-sensitive-competitive learning algorithm have been utilized with a great deal of success to overcome the problem of codebook design in vector quantization. A detailed investigation of the network parameters has been conducted to achieve the optimal performance in image compression.

A set of fourteen color images obtained from the Internet were used as training and test samples. The results have shown that the quality of decompressed image depends on the choice of the network parameters. The quality of the decompressed images, was compared to the originals visually and using the peak-signal-to-noise-ratio (PSNR) as a measure of quality.

I. Introduction

In recent years the demand for digital image transmission and storage has been increased dramatically. This is due to the advances in multimedia applications and digital imaging technology. Obtaining high-resolution digitized images from databases over a computer network is becoming more popular than ever. This has lead to a growing need for compressing images, both for archiving and transmission. Compressing a digital image can facilitate its transmission, storage, and processing.

The overall goal of image compression is to encode images or image sequences into as few bits as possible so that a decoding mechanism reconstructs the original image with an acceptable visual and/or information quality. Many techniques have been suggested to implement digital image compression. One of the most popular techniques in image compression is vector quantization [1,2]. Vector quantization provides high compression ratios and simple decoding processes. However, studies on practical implementation of VQ have revealed some difficulties in codebook design. One of these difficulties is the calculation complexity required for codebook design.

The Kohonen self-organizing feature map (KSOFM) neural network has the ability to form clusters from training samples [2]. The success of KSOFM to make clustering or compression depends on a number of parameters. These parameters are the initial weight values, the learning rate or the updating gain, the neighborhood function gain, and the number of neighborhood nodes surrounding the winner node. Choosing the network parameters affect the overall performance of the network. There are no theoretical rules or guidelines for choosing these parameters. Previous researches in image compression using KSOFM concentrated on how to dimension the map [3,4], how to combine the algorithm with other algorithms [5], the convergence analysis of the network [6], and comparing the KSOFM performance with other algorithms. However, the effect of choosing the network parameters has not been studied in depth in the literatures [7-15].

Accepted May 10, 1999

The objective of this work is to investigate different mathematical forms for the Kohonen self-organizing map parameters in order to reach the optimum performance of the network, i.e., to obtain the highest compression ratio with the highest degree of image fidelity.

This paper is organized in five sections as follows: Section II briefly reviews the background of image compression techniques. Section III presents a background of Self-Organizing Feature Map network. Section IV presents the results obtained from the application of Kohonen self-organizing feature map algorithm, the parameter selection, and evaluation of the performance. Conclusion and discussion are presented in Section V.

II. Image Compression: a background

Generally, there are two basic types of compression: lossless and lossy compression. Lossless compression, which is also called noiseless coding, data compacting entropy coding, or invertible coding, refers to algorithms that allow the original pixel intensities to be perfectly recovered from the compressed representation. On the other hand, lossy compression algorithms do not provide exact recovery after compression [9,10].

Image compression refers to the process of reducing the amount of data redundancy in the image. Three basic data redundancies can be identified and exploited. These are coding redundancy, interpixel redundancy, and psychovisual redundancy. Data compression is achieved when one or more of these redundancies are reduced or eliminated.

A- Coding Redundancy

Coding redundancy appears, if the gray or color levels of image are coded in a way that uses more symbols than absolutely necessary to represent each level. In general, coding redundancy is present when the codes assigned to a set of events (such as gray level values) have not been selected to take full advantage of the probabilities of the events. To prevent coding redundancy, the code length must be variable and the more probable gray or color levels will be coded by the shortest code vector and next probable levels by next short code, this processing achieves data compression [10].

B- Interpixel Redundancy

This kind of redundancy is due to the correlation between pixels. This correlation results from the structural geometric relationships between the objects in the image. This will make great degree of correlation between pixels of the image so the value of any given pixel can reasonably be predicted from the value of its neighbors [10]. In order to reduce the interpixel redundancies in an image the 2-D pixel array normally used for human viewing and interpretation must be transformed into a more efficient (but usually *non visual*) format, for example, the differences between adjacent pixels can be used to represent an image.

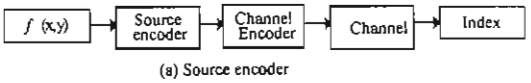
C- Psychovisual Redundancy

Such phenomena result from the fact that the eye does not respond with equal sensitivity to all visual information. Certain information simply has less relative importance than other information in normal visual processing. This information is said to be psychovisually redundant. It can be eliminated without significant impairing the quality of image perception [10].

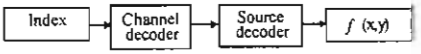
II.1 Image Compression Models

The general method that represents image compression system is shown in Fig. (1). It consists of two distinct structural blocks; an encoder and decoder or in other words compressor and decompressor [9].

An input image $f(x,y)$ is fed into the encoder which creates a set of symbols from the input data. After transmission over a channel, encoded representation is fed to the decoder, where a reconstructed output image $\hat{f}(x,y)$ is generated. In general $\hat{f}(x,y)$ may or may not be an exact replica of $f(x,y)$. If it is, the system is error free (information preserving), if not, some level of distortion is present in the reconstructed image.



(a) Source encoder



(b) Source decoder

Fig 1. Image compression model.

Both the encoder and decoder shown in Fig. (1) consist of two relatively independent functions or sub-blocks. The encoder is made up of a source encoder which removes input redundancies, and channel encoder, which increases the noise immunity of the source encoder's output. As would be expected, the decoder includes a channel decoder followed by a decoder. If the channel between the encoder and decoder is noise free, the channel encoder and decoder are omitted, and the general encoder and decoder become the source encoder and decoder, respectively.

II.2 Vector Quantization

A vector quantizer is a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over or storage in a digital channel. The goal of such a system is data compression to reduce the bit rate so as to minimize communication channel capacity or digital storage memory requirements while maintaining the necessary fidelity of the data [9]. This operation is nonlinear and noninvertible, it is *Lossy*.

Vector quantization (VQ) has emerged in recent year as a powerful technique that can provide large reductions in bit rate while preserving the essential signal characteristics [11-13]. A typical VQ system is depicted in Fig (2), where an input vector X consisting of blocks of pixels is quantized by being encoded into a binary index which then serves as an index for the output reproduction vector or code words. The compressed image is represented by these indices, and the compressed representation requires fewer bits than the original one. Practically, the optimal VQ should reduce the inter-vector correlation as much as possible and preserve the intra-vector correlation as much as possible. However, a general drawback for VQ is the computational load needed during the encoding stage, where an exhaustive search is required to calculate the distortion between an input vector and all the other codebook vectors to get the entire codebook. This is the reason why VQ had a limited use in the past in the field of image compression.

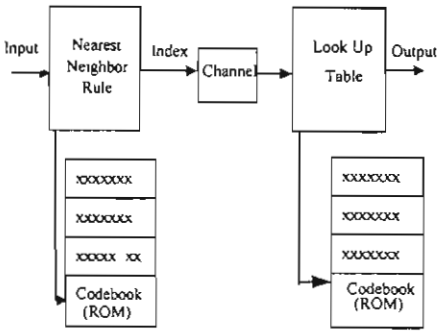


Fig. 2. A block diagram of a simple quantizer.

In order to make VQ to be an effective tool for data compression and obtain real time processing neural networks have been used to perform these massive computations.

III. Self-Organizing Feature Map Neural Network

The neural network model utilized in this work is the Kohenen's self-organizing Feature maps (KSOFM) algorithm. The KSOFM implements a characteristic of nonlinear projection from the high-dimensional space of sensory or other input signals onto a low dimensional array of neurons [7,8].

The technique is very useful, and in some ways related to the human learning process. However in many applications, it would be more beneficial if we could ask the network to form its own classifications of the training data. To do this we have to make two basic assumptions about the network; the first is that a class membership is broadly defined in input patterns that share common feature, the other is that the network will be able to identify common features across the range of input patterns.

III.1 Network Topology and Operation of KSOFM

The topology of a typical KSOFM network shown in Fig (3). It has n input nodes and mxm output nodes. Each output node j has a connection from each input node i, where w_{ij} being the connection weight between them. The procedures of Kohonen algorithm may be summarized as follows [7]:

1- Assign small random values to weights w_{ij}

2- Choose a sample vector X and apply it to the input of the network.

3- Find the winning output node j^* whose d_j is minimum using the following criteria:

$$d_{j\min} = \min\left[\sum_{i=0}^{n-1}(X_i(t) - w_{ij}(t))^2\right] \quad (1)$$

where w_{ij} is the weight connecting input nodes i to the output node j.

4- Updates the weight of winner node and its neighborhood nodes using the update rule given by:

$$w_{ij}(t+1) = \{w_{ij}(t) + \eta(t).(X_i(t) - w_{ij}(t))\} \text{ for } j \in \Lambda j(t) \quad (2)$$

Where $\eta(t)$ is the gain value $\Lambda j(t)$ is the neighborhood function.

5- Repeat steps 2 to 4 until no noticeable changes in the feature map are observed. The size of the neighborhood N_c is a function of time t and shrinks monotonically. The parameter gain is the step size of the adaptation weights and also shrinks monotonically with time.

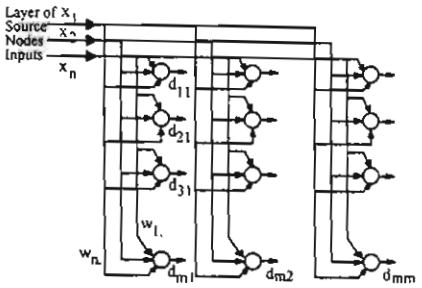


Fig. 3. Network topology of KSOFM.

In fact there are many properties that affect the performance of the algorithm. The first property is the boundary effect that may deform the maps. This effect is caused by the fact that arrays of processing units had borders. The outermost units had no neighbors on one side with which they could interact. Since the distribution of training vector is usually bordered too, some kind of "boundary effect" can then be observed. The most typical is "contraction" of the distribution of weight vectors, in fact the algorithm competes the distance between the weight vector and the input vector i.e., the difference between the orientation of the weight vector and the input vectors. So that the learning processing is trying to rotate the weight vectors orientation toward the orientation of the input vectors [7]. As the outermost units have no neighbors on one side these outermost vector will rotate more often inwards than outward [10,11] or by other means the output nodes at the boundary learn more the center nodes do.

The second effect is a general problem in the competitive learning architecture. If input patterns have complex structure, the net becomes unstable. It may then easily happen in self-organizing feature map that the weight vectors lying in zero density areas are affected by input vectors from all the surrounding parts of the non-zero distribution [10,11].

The third property is that a large groups of units give the same response may be found. This phenomenon occurs when the threshold of the node is too low. This means that if we have a number of units give the same value, then we have useless nodes. This is because only one node is enough to represent each group [10,11].

The fourth phenomenon is that some neural units may be underutilized. This means that nodes never learn. This is because the initial values of these units are very far from the training sample, or some nodes always win and do not let any chance to other nodes to win so that they update their weight vector [10,11].

III.2 The Modified KSOFM Algorithm (MKSOFM)

A modification to the basic Kohonen algorithm has been suggested by Oscal et al (1994) [10] to overcome under-utilization phenomenon. The modification proposed includes a new term known as the frequency-sensitive cost function to the learning rule. The frequency-sensitive cost function includes the use of a counter for each win node to be increased by one; then in the next iteration each output of the node is multiplied by its counter value. Then the minimum outcome (the frequency-sensitive cost function) is selected to be the winning node. This means that if we have two nodes having the same distance from the input pattern then we give the priority to the node which has the smallest outcome. By this way the network will have less chance to have two or more nodes having the same value and this leads to a lower probability to have a dead node or repeated node. The modified algorithm can be summarized in the following steps:

1- Assign small random values to weights w_{ij}.

2- For an input vector X_i find the frequency-sensitive distortion $Fd_j = d_j(X_p, w_i)$ for all the neurons given by

$$Fd_j = (1 + \frac{Fj}{F_{thd}}) \sum_{j=1}^{n-1} (x_i - w_{ij}(t))^2 \quad (3)$$

where F_j is the counter value of the node which initially set to zero, and F_{thd} is the frequency threshold which is a constant value chosen to be the average training frequency of each codevector. So, if we have a node number N and a K vector for training and the iteration number is T then the frequency threshold F_{thd} can be computed as:

$$F_{thd} = \frac{T * K}{N} \quad (4)$$

the term $\frac{Fj}{F_{thd}}$ represents the frequency sensitivity cost function.

3- Select the node with the smallest frequency sensitive distortion and label it as the winner node $w_i^*(t)$ and increase its winning frequency counter Fj by one.

4- Update the weight vector of that winner node and its neighborhood.

$$W_{ij}^*(t+1) = W_{ij}^*(t) + \mu[X(t) - W_{ij}^*(t)] \quad \text{for } j \text{ in } Nj^*(t) \quad 0 \le i \le n\text{-}1 \quad (5)$$

where μ is the updating gain.

5- Repeat by going to step 2 through 4 for all training vectors.

III.3 Choosing the Network Parameters

The learning process involved in the computation of a feature map is stochastic in nature, which means that the accuracy of the map depends on the number of iterations of the KSOFM algorithm. Moreover, the success of map formation is critically dependent on how the main parameters of the algorithm, namely the initial weight value, the gain, the neighborhood size, and the neighborhood function gain Λ, are chosen. Unfortunately, there are no theoretical bases for choosing such parameters [1,7-15].

In the present work several various forms of these network parameters are utilized to test the performance of the chosen network [16]. These various forms are depicted in Table 1.

Table 1 Different forms of Kohonen network parameters [16].

The neighborhood function gain	Updating gain	Initial value of the weights	Neighborhood size
-Decreasing with the distance -Constant with the distance -Neighborhood only adjacent to the winner.	$gain = 1 - \frac{0.9t}{T}$ $gain = \frac{0.4}{1 + 0.0005t}$ $gain = \frac{1}{1 + \log(1+t)}$ $gain$ = 0.1, 0.3, 0.5, 0.7, 0.9 Where t is iteration step T total number of iteration	-All initial weights are zeros. -All the initial weight equal to 128. -Random weights values between 0 and 255 -Random weights values between 125 and 130	- All the nodes - 90 nodes - 72 nodes - 36 nodes - 18 nodes - 8 nodes - 6 nodes - 4 nodes - zero

III.4 Performance Evaluation

In order to evaluate the performance of the network using different forms of each parameter, a metric is needed to quantify the quality of the reconstructed image. The most commonly used metric is the peak signal-to-noise ratio (PSNR) defined by [17,18]

$$\text{PSNR}=10\log_{10}\frac{(2^k-1)^2}{e_{rms}^2} \tag{6}$$

Where k is the maximum number of bits to represent each pixel and the e_{rms} is root-mean-square error defined as

$$e_{rms}=\sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[g(x,y)-f(x,y)\right]^2} \tag{7}$$

where MxN is the image size, $g(x,y)$is the original image pixel at position (x,y), and $f(x,y)$ is the decompression pixel at position (x,y).

IV. Experimental Results

In this section, we present the experimental results obtained by using the KSOFM. Here the MKSOFM algorithm is used for training and coding.

IV.1 Test Data

Fourteen color test images of size 256x256 of extension BMP have been collected from the Internet (Figure 8). Two factors have been taken into consideration in selecting these images. First, these images are representing different categories; for example faces, buildings, animal, ...etc. Second, these images represent a wide range of spatial frequency.

Table .2 illustrates the spatial frequencies of the used images. Spatial frequency for a given image is defined as follows: consider an M x N image, where M = number of rows and N number of columns [17].

$$\text{The Row Freq.} = \sqrt{\frac{1}{MN}\sum_{j=0}^{M-1}\sum_{K=1}^{N-1}\left(F(j,k)-F(j,k-1)\right)^2}$$

$$\text{The column freq.} = \sqrt{\frac{1}{MN}\sum_{j=1}^{M-1}\sum_{K=0}^{N-1}\left(F(j,k)-F(j,k-1)\right)^2}$$

$$\text{The spatial frequency} = \sqrt{(Row\ \text{freq})^2+(\text{column freq})^2} \tag{8}$$

IV.2 Test Procedures

A program written in C-language has been developed for the MKSOM algorithm. The program simulates a single layer Kohonen network. Each node is connected to all input vectors through individual connection weight values. The test images of 16M. colors and of size N X M were divided into non-overlapping blocks of size nXn pixels corresponding to a vector of length n^2. Each block represents an input vector to the network. The network makes clustering to these vectors. If there are many vectors representing the same cluster, only one node will represent all these vectors. Dealing with a vector belonging to one specific set will be through the index of its representative node, so that when the network terminates the training phase, it will produce a codebook. Each codevector in this codebook represents a cluster or a set of block vector in the image.

Table 2: The spatial frequencies of the used test images

Image name	Spatial frequency
Lena	54.29
Filfil4	16.89
Filfil5	22.06125
Gile5	20.4232
Forest13	24.067
Inpvec1	27.68126
Inpvec10	28.622872
Inpvec16	19.2957
Inpvec17	21.785
Inpvec20	32.9169
Sorhani	49.3837
Sunset	27.35622
Teacher1	43.796
Teacher2	30.889522

In the coding phase each block vector in the image is compared to all codevectors in the codebook. The closest codevector will be the representative to the input codevector. The index of the representative codevector will be stored or transmitted instead of the codevector itself. This means that the image is represented by a set of indices to the codebook instead of the pixel values. As the size of input vector increases the compression ratio also increases. The size of input vector or the image's sub-

block dimension varies from a block of 2x2 to a block of 16x16, but also as the block size increase more nodes will be needed and more iteration in training phase will be required. In the present work a minimum block size of (2x2) has been used as the parameters under investigation are independent on the block size or the compression ratio.

Initially the selection of the network parameters were chosen as follows [16]:

1- The updating gain is η =0.4/(1+0.0005t)

2- Initial weight values are random values lying between 125 and 130 (in the middle of the expected input values range which are from 0 to 255).

3- The neighborhood nodes are all the nodes in the network.

4- The neighborhood function gain is a decreasing function with distance from the winner node.

All various forms of each parameter were tested, while fixing the others, as indicated in Table 1. The test was performed by compressing and decompressing each image then the peak signal to noise ratio (PSNR) between the original and the decompressed image was calculated for each parameter form. The form which gave the maximum PSNR was selected to be the optimum form of this parameter and was used for choosing the next parameter and so on. These optimal parameters were used to train the network to generate a codebook for image compression. This codebook was tested by other new test images (not used in training) and their PSNR were calculated.

A- Testing The Effect of The Network Weights Initial Values

To determine the optimal weights for the network performance, the influence of four different forms of the initial weights shown in Table 1, was investigated. The other three network parameters which, are the updating gain, neighborhood size, and neighborhood updating gain were chosen as stated previously.

Using the above parameters, the fourteen test images were compressed and decompressed then the PSNR of each image was calculated. The results are shown in Fig. 4.

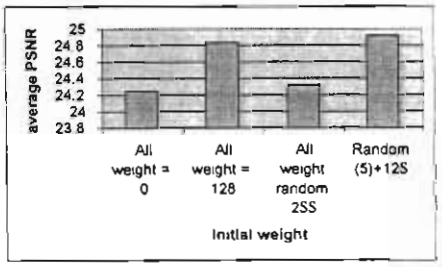


Fig. 4. Effect of initial weights.

From this figure it can be seen that the best PSNR can be obtained when the initial weight values are chosen randomly with values lying between 125 and 130.

B- Testing The Effect of The Updating Gain

All forms of the updating gain listed in Table 1, the neighborhood size, and the neighborhood function gain were chosen as stated before. However, the initial weight values were chosen to be random values lying between 125 and 130. The results of the PSNR obtained are shown in Fig. 5.

From this figure, we can conclude that the best PSNR can be obtained when the updating function gain is chosen in the form 1-0.9t/T .

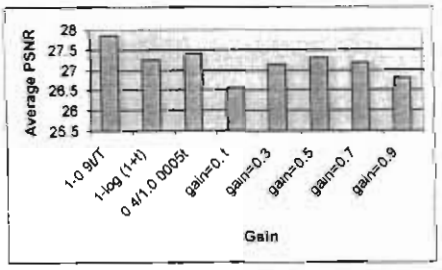


Fig. 5. Effect of updating gain.

C- Testing The Effect of The Neighborhood Size

The neighborhood size means the initial number of nodes that are surrounding the winner node. The neighborhood size was chosen as stated in Table 1. The other three network parameters were chosen as follows:

- The initial weight are random values lying between 125 and 130
- The updating function gain is in the form $1-0.9t/T$, where t is the iteration step number.
- The neighborhood updating function gain decreases with distance from the winner node.

Fig. 6 shows the resulted PSNR for the decompressed images. These results indicate that the best performance can be obtained when the neighborhood size selected is 6 nodes surrounding the winner node.

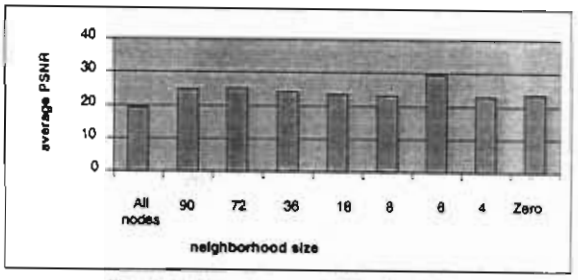


Fig. 6. Effect of the neighborhood size

D- Testing The Effect of The Neighborhood Function Gain

The neighborhood function gain means the updating gain of the neighborhood nodes. There are three neighborhood function gains as indicated in Table 1. The other three network parameters, which are the initial weight values, the updating gain, and the neighborhood size were chosen as follows:

- The initial weight values are random values lying between 125 and 130.
- The updating gain is in the form: $1-0.9t/T$, where t is the iteration step number.
- the neighborhood size to be 6 nodes.

The results are as shown in Fig. 7. It is clear that, the best PSNR can be obtained using the decreased function gain with distance.

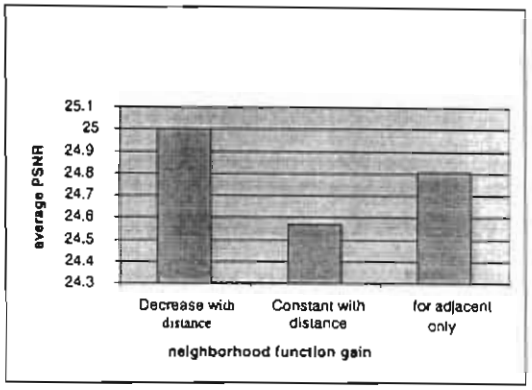


Fig. 7. The effect of neighborhood function gain.

V. Discussion and Conclusion

In this work, the application of neural networks to the problem of image compression has been investigated. A Kohonen self-organizing feature map (KSOFM) and a modification of frequency-sensitive competitive learning algorithm have been applied in a number of ways in order to obtain the optimum performance for image compression [16]. A detailed study of how to choose the Kohonen network parameters has been performed using a sample of 14 training images and 10 test images.

It can be concluded that the best quality of the decompressed images can be obtained when the initial weight values are chosen as small random values lying in the middle of the expected input range. This decreases the probability of having initial weight very far from the input values.

Updating neuron's weight vector in response to a training presentation means slightly rotate the weight vector toward the direction of the data vector. The updating gain is the factor that controls the movement of the weight value towards the training data. Best results are usually obtained by slowly decreasing the gain as the training progresses. A large updating gain will allow the training to progress faster. If the updating gain is too large, though, convergence may never occur. The weight vectors may oscillate widely. A good way to determine if the updating gain is acceptable is to watch the maximum error vector. The maximum error vector should quickly and steadily decrease. If it decreases slowly, the updating gain is too small. If it occasionally increases, the learning rate is dangerously high, leading to instability.

The results have shown that the best quality of the decompressed images are obtained using an updating gain having the form $1-0.9t/T$, where t is the iteration step and T is the total number of iterations. This is in agreement with previous researches [8]. The initial value of the gain is unity then it decreases with time and at the same time the term $0.9t/T$ prevents the gain value to be less than 0.1. Preventing the gain to be less than 0.1 will avoid the distortion that may occur after self-organizing due to very small updating gain. The updating function gain is linear this means the decreasing step size depends on number of network iteration, which means that more iteration steps leading to more tuning and better performance of the network.

As for the neighborhood size, the results obtained showed that the best image quality can be obtained using initial neighborhood size equals to 6 nodes. This result may seem to disagree with Kohonen initial suggestion that the neighborhood initial size should be all the nodes in the network used [8]. But for images, the effect of one color will never extended to all nodes in the image. The images generally contain different colors or clusters so that if neighborhood selected to be initially all the network nodes; this means that the image is supposed to include only one color or cluster, which is not accepted. Selecting the neighborhood size to be 6 nodes will reduce the required iteration steps to reach convergence.

The results also have shown that the neighborhood function gain around the winner node gives better performance when it has constant amplitude. This means all nodes inside the topological neighborhood fire at the same rate and interaction among those nodes is independent of their lateral distance from the winner. However, from a neurobiological viewpoint there is evidence for lateral interaction among neurons in the sense that a neuron that wins tends to excite the neurons in its immediate neighborhood more than those away from it. This means the neighborhood function gain decreases with distance from the winner. The results show that the best image quality can be obtained when a decreasing neighborhood function-gain with distance from the winner was used.

REFERENCES

1. Lu Cheng-Chang, and Yong Ho Shin "A neural network based image compression system", IEEE Trans. Communications, Vol.38, No.1, pp.25-29, February 1992.
2. Haykin S., Neural networks A Comprehensive Foundation. Addison-Weseley 1992.
3. Awcock G. J., Applied Image Processing. Addison-Weseley 1996.
4. Linde Y., Buso A., and Gray R. M. "An algorithm for vector quantizer design", IEEE Trans. Communications, Vol. 28, No. 1, pp. 84-95, January 1980.
5. Gray R. M. "Vector Quantization" IEEE ASSP magazine, pp. 4-29 April 1984.
6. Dony R. D. and Haykin S. "Neural network approaches to image compression", IEEE Proc., Vol.83, No. 2, pp. 288-303, February 1995.
7. Kohonen Teuvo "Self-organized formation of topologically correct feature maps", Neuro-computing, 1990.
8. Kohonen Teuvo, Erkki Oja, Olli Simula, Ari Visa, and Jari Kangas "Engineering applications of the Self-Organizing map", IEEE Proc., Vol.84, No.10, pp.1358-1384, October 1996.
9. Nasrabadi N. M. and King R. A. "Image coding using vector quantization: A Review", IEEE Trans. Communications, Vol.36, No.8, pp. 957-971, August 1988.
10. Oscal T. C. Chen, Bing J. Sheu ,and Fang Wai-Chi "Image compression using self-organization networks" IEEE Transaction on Circuits and Systems for Video Technology, Vol.4, No.5, pp. 480-489, October 1994.
11. Krishnamurthy Ashok K., Stanley C. Ahalt, Douglas E. Melton, and Prakoon Chen "Neural networks for vector quantization of speech and images", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 8, pp. 1449-1457, October 1990.
12. Cosman Pamela C., Gray R. M., and Vetterli Martin "Vector quantization of image sub-bands: A survey", IEEE Trans. Image Processing, Vol. 5, No. 2, pp. 202-227,February 1996.
13. Oehler Karen L., and Gray R. M. "Combining image compression and classification using vector quantization" IEEE Trans. Pattern Analysis and Machine Intelligence Vol. PAMI-17, No. 5, pp. 461-473, May 1995.
14. Fowler J. E. Kenneth, Adkins C., Bibyk Steven B., and Stanley C. Ahalt "Real time video compression using differential vector quantization", IEEE Trans. Circuits and Systems for Video Technology, Vol. 5, No. 1, pp. 14-24, February 1995.
15. Ienne Paolo, Patrick Thiran, and Nikolaos Vassiley "Modified self-organizing feature map algorithm for efficient digital hardware implementation", IEEE Trans. Neural Networks, Vol.8, No.2, March 1997.
16. Awadalla M. M., "Image Compression Using Neural Networks", M. Sc. thesis, Faculty of Engineering, Mansoura University, 1998.
17. Eskiciogla A. M. and Fisher Paul S. "Image quality measures and their performance", IEEE Trans. Communications, Vol. 43, No. 12, pp. 2959-2965, December 1995.
18. Cosman Pamela C., Gray R. M., and Olshen R. A. "Evaluation quality of compressed medical image" IEEE Proc., Vol. 82, No. 6, pp. 919-931, June 1994.

Original images	Decompressed images

Figure 8. A sample of the original test and decompressed images.