1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Fractal Image Compression and Decompression by Neural Network Approaches

S. J. Lee [*], *K. T. Sun* [*], *and P. Y. Wu* [+]

[*] Institute of Information Education

[+] Department of Mathematics Education

National Tainan Teachers College, Tainan, Taiwan, R.O.C.

{ M85503, ktsun, pywu}@ipx.ntntc.edu.tw

Abstract

In image compression technologies, fractal image compression and decompression have advantages of high compression-ratio and low loss-ratio. But, it requires a great deal of computation, which limits its applications. And, still now, there is no parallel processing technique that had been designed and implemented. In this paper, we applied neural networks to implement the numerous computations of fractal image compression and decompression in parallel. The simulation results show that the quality of generated pictures by neural networks is similar to traditional methods, which verifies the high value of our research – the neural network technologies are useful and efficient for fractal image compression and decompression.

Keywords: fractal image compression and decompression, neural networks, parallel processing.

1. Introduction

Recently, the graphical representation in computer is widely applied to many applications for its meaningful representation to human beings. However, it requires large storage and long transmission time.

The technique of image compression and decompression is useful and important for reducing the storage space and transmission time. In general, these compression technologies can be divided into two fields– lossy compression and lossless compression whether decompressed image is the same as the original one or not. If the proper loss-ratio is allowable, the lossy compression methods can get higher compression-ratio [1].

There are usually three technologies used in lossy compression–vector quantization (VQ), discrete cosine transformation (DCT) and fractal image compression. The method of VQ is to partition an image into numerous sub-images and to find some representatives as a codebook among them [2, 3]. The method of DCT is to convert the gray levels of an image into other coordinates (e.g., frequency), then quantizes and stores them [1, 4]. By the self-similarity characteristics in an image, an image will converge to an acceptable status after fractal image decompression [5, 6].

Unlike VQ, The fractal image compression does not require a codebook in decompression procedure [2]. The fractal image compression is also attractive by its high compression-ratio and low loss-ratio properties [4]. There are already some results in this technique: the Hutchinson metric is proposed to prove the condition of converging [7-9], and Mandelbort had generated images by fractal theory [9]. By developing a collage theorem and the iterated function system (IFS), Barnsley produces a high compression-ratio $\left(10^4:1 \sim 10^6:1\right)$ fractal code, and then it encourages many related researches to proceed [5, 7, 10-13]. But the fractal code can not be generated automatically by using IFS [4, 13-17]. Jacquin proposed a partitioned iterated function system (PIFS) to improve the IFS such that the fractal code is determined automatically [14, 15]. However, a great deal of computation is also required.

The neural network is a new and useful

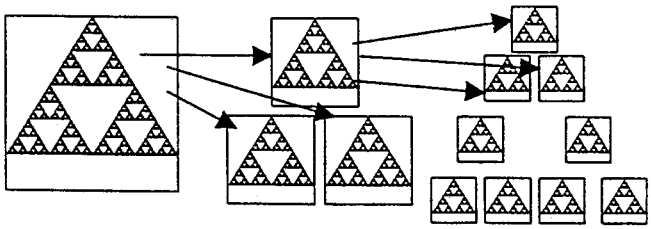
technology, and has been successfully used in many scopes [18-31]. J. Stark first proposed a research to apply the neural network on IFS [19, 20, 32]. His method is based on the Hopfield neural network to solve the linear progressive problem and got the Hutchinson metric quickly [20, 28]. However, his neural network approach only works in IFS decompression procedure.

In this paper, we apply neural network technologies on PIFS such that the fractal code is generated automatically. In our method, a neuron is used to represent a pixel of image, and the weights and the thresholds are used as the fractal code. In this way, proper weights and thresholds can be obtained in compression (training) procedure, and the original image can be constructed in decompression (retrieving) procedure. In Section 2, We will introduce PIFS theory and the idea of compression and decompression by using neural network technologies. In Section 3, the image compression by using neural networks on two different models will be introduced. Then, the decompression method is explained in Section 4. Section 5 shows some simulation results. Finally, a brief conclusion is discussed in Section 6.

2. Reviews of the partitioned iterated function system and neural network

2.1. Basic concepts of the fractal image compression.

The basic concept of fractal image compression is to use the characteristics of the self-similarity in an image. In figure 2-1 (a), the triangular can be divided into three sub-images, as shown in figure 2-1 (b). All of these sub-images are the same as the original image except that the size becomes a quarter, and they can be partitioned into smaller parts as shown in figure 2-1 (c). The smaller parts are also similar to those sub-images, respectively. These relationships are existed continuously between sub-images while partition operations proceed repeatedly. Then, we only determine these transformation functions how to map the original image to the sub-images. An image can be compressed by finding these transformation functions, and then the procedure of decompression can be performed by these transformation functions to construct the original image.



(a) (b) (c)

Figure 2-1. The characteristics of the self-similarity used in fractal image compression. (a) is the original image, (b) contains three similar sub-images after one iteration, and (c) contains nine similar sub-images after two iterations.

In figure 2-2, by applying decompression procedure, a different initial image can be used to construct a triangular image (i.e., original image), as shown in Fig. 2-1 (a), after fifteen iterations.

(a) (b) (c) (d)

Figure 2-2. The decompression procedure of fractal image. (a) is the initial image, (b) after one iteration, (c) after three iterations, and (d) after fifteen iteration.

There are three mapping transformations for figure 2-1, shown in the following *Eq.* (2-1), and this procedure is called as iterated function system (IFS).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \tau_1\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \tau_2\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}, \quad (2\text{-}1)$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \tau_3\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix},$$

where (x, y) is the coordinate of original image, and (x', y') is the coordinate of transferred image. Then, only three

transformation functions, $\{\tau_1, \tau_2, \tau_3\}$ (also called as fractal code), are stored instead of the image data.

2.2. Partitioned iterated function system (PIFS)

However, if we want to find the fractal code of IFS, which is almost impossible for figure 2-3 (a). But we can find some similarities between blocks of sub-images. As we can see, there are two pairs of blocks, which are similar to each other, as shown in figure 2-3 (b). One pair of them is the part of hat and the part of shoulder, and the other pair is the smaller part and the bigger part of face.

(a) (b)

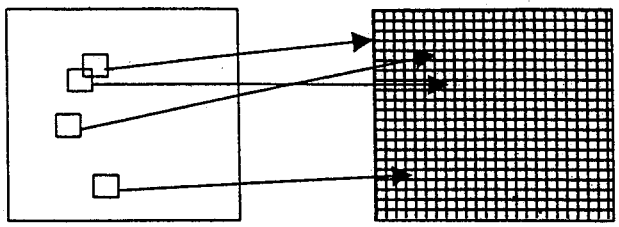
Figure 2-3. There are some similarities between sub-images in the image Lena. (a) is the orignal image, and (b) shows two pairs of blocks with similar shape.

When we consider the gray levels of an image, an additional dimension is added. The transformation function will become to the following *Eq.* (2-2):

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \tau_i \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}, \quad (2\text{-}2)$$

where z and z' are the gray levels, a_i、b_i、c_i and d_i are coordinates of this transformation, (e_i, f_i) is the offset of the transformation, and s_i and o_i represent contrast and brightness, respectively.

Figure 2-4 shows the concept of PIFS. Two identical images are partitioned and compared. Each non-overlapped sub-image in (b) will find a τ_i to transform a larger and similar sub-image from (a) to it.



(a) (b)

Figure 2-4. The concept of PIFS. (a) overlapped and larger images, and (b) non-overlapped and smaller images.

If we choose the size of sub-images in figure 2-4 (a) are 4 times (double length of height and width) of sub-images in figure 2-4 (b), then *Eq.* (2-2) will become *Eq.* (2-3).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \tau_i \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}. \quad (2\text{-}3)$$

These transformation functions are the fractal codes that can be used to represent the compressed image and be used to decompress for the image.

3. Applying neural networks to the fractal image compression

We propose two different neural network models to implement fractal image compression and decompression, and the architectures of these two models are similar except the transformation functions, and illustrated in figure 3-1. Each pixel of the image is processed by a neuron and the gray level of a pixel is represented by the state of a neuron. An image is duplicated into two images, and each is divided into many sub-images, called domains and the ranges, each pixel in domains corresponds to an input neuron, and each pixel in ranges corresponds to an output neuron. For each output neuron, four input neurons are connected to it. Therefore, each output neuron j connects to four input neurons i、$i+1$、$i+2$ and $i+3$. The output value z'_j of neuron j is determined by the values z_i、z_{i+1}、z_{i+2}、z_{i+3}, and the corresponding

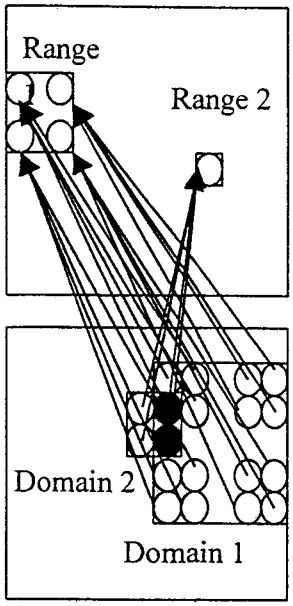


Figure 3-1. The architecture of the proposed neural network for implementing the fractal image compression.

weights W_{ji}、W_{ji+1}、W_{ji+2}、W_{ji+3} and the threshold θ_j.

Two different activation functions, the linear model and nonlinear model, of neurons are defined as the following *Eq.* (3-1) and *Eq.* (3-2), respectively.

$$z'_j = O_j\left(\sum_{k=i}^{i+3} W_{jk} \times Z_k - \theta_j\right), \qquad (3\text{-}1)$$

$$z'_j = O_j'\left(\frac{1}{B}\sum_{k=i}^{i+3} W_{jk} \times Z_k - \theta_j\right), \qquad (3\text{-}2)$$

where B is the maximum value of gray levels (e.g., B is assigned to 255 in this paper).

3.1. The linear model

Comparing *Eq.* (2-3) and *Eq.* (3-1), the value Z_k in *Eq.* (3-1) can be viewed as the gray level z of a pixel in *Eq.* (2-3). The weight w_{jk} and the threshold θ_j in *Eq.* (3-1) can be also viewed as a quarter of the contrast S_i and the negative value of the brightness O_i in Eq. (2-3), respectively. Then, the linear neural network approach (*Eq.* (3-1)) implements the computation of **PIFS** (*Eq.* (2-3)), and the activation function O_j is defined as following equation:

$$O_j(x) = \begin{cases} x, & \text{when } 0 \le x \le 255. \\ 0, & \text{otherwise.} \end{cases} \qquad (3\text{-}3)$$

Figure 3-2 shows the graphic representation of Eq. (3-3).

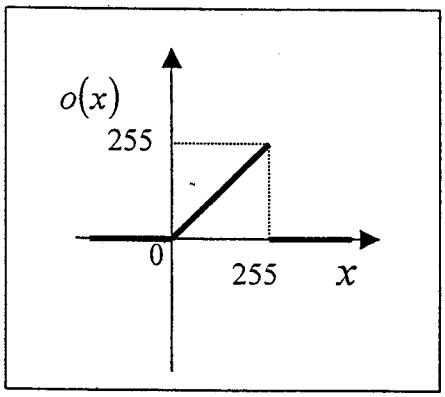


Figure 3-2. The activation function of neurons in the linear model.

According to the output values of neurons and the original gray levels of pixels, we can compute the difference δ_j between them for each neuron j by the following equation:

$$\delta_j = z_j^{true} - z'_j, \qquad (3\text{-}4)$$

where z'_j is the output value of activation function and z_j^{true} is the original gray level of pixel j. Then, the updated weight ΔW_{jk} between the output neuron j and the four corresponding input neurons k, $k = i \sim i+3$, can be derived as *Eq.* (3-5).

$$\Delta W_{jk} = \eta \times \delta_j / Z_k, \qquad k = i \sim i+3, \qquad (3\text{-}5)$$

where η is a learning rate parameter which can speed up the converging rate, and find a better solution. The updated value of threshold, $\Delta\theta_j$, is then defined as

$$\Delta\theta_j = -\eta \times \delta_j. \qquad (3\text{-}6)$$

The learning procedure is processed repeatedly until the output values of network are acceptable.

3.2. The nonlinear model

In the nonlinear model, the activation function O_j' is defined as *Eq.* (3-7), which is a composition function.

$$O_j'(x) = DeNor(Sigmoid(x)), \quad (3\text{-}7)$$

where

$$Sigmoid(x) = \frac{1}{(1+e^{-x})}, \quad (3\text{-}8)$$

$$DeNor(x) = K \times (x - \alpha), \quad (3\text{-}9)$$

The values of K and α in *Eq.* (3-9) are constants and are defined as:

$$K = \frac{B}{(Sigmoid(Upper) - Sigmoid(Lower))}. \quad (3\text{-}10)$$

$$\alpha = Sigmoid(Lower). \quad (3\text{-}11)$$

We define an input range, $2R$, to avoid the output of *Eq.* (3-8) to be trapped into saturation states. Then, the difference between the maximum value and the minimum value of x is $2R$ (i.e., $Upper - Lower = 2R$). Therefore, the output range of sigmoid function is equal to $[Sigmoid(Lower), Sigmoid(Upper)]$. Figure 3-3 shows these relationships.

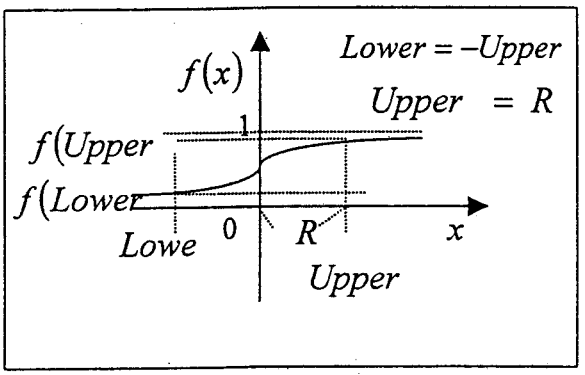


Figure 3-3. The activation function of neurons in the nonlinear model.

For each neuron j, we define the difference between the output of the proposed neural network and the original gray level of pixels as the following equation:

$$\delta_j = \frac{z'_j(B - z'_j)(z_j^{true} - z'_j)}{B^3} \quad (3\text{-}12)$$

In *Eq.* (3-12), all values of z'_j, B and z_j^{true} are in the range $[0, 255]$. Then, *Eq.* (3-12) can be divided by B^3 for keeping the output value in the range $[-1, 1]$. Similarly, The updated weight Δw_{jk} can be derived as *Eq.* (3-13).

$$\Delta w_{jk} = \frac{\eta \times \delta_j \times z_k}{B}, \quad k = i \sim i+3. \quad (3\text{-}13)$$

The steps of learning are operated repeatedly until the output values of neurons are acceptable.

4. Applying the neural network to the fractal image decompression

The architecture of our neural network for solving image decompression is similar to figure 3-1 except that the outputs of neurons in the output layer will feed back to the corresponding neurons in the input layer. The trained weights and threshold (i.e., fractal code of **PIFS**) had been determined during the fractal image compression.

The output state $z_j'^{(t)}$ for image decompression is defined as *Eq.* (4-1).

$$z_j'^{(t)} = \begin{cases} O_j\left(\sum_{k=i}^{i+3} w_{jk} \times z_k^{(t)} - \theta_j\right), \\ O_j'\left(\frac{1}{B}\sum_{k=i}^{i+3} w_{jk} \times z_k^{(t)} - \theta_j\right), \end{cases} \quad (4\text{-}1)$$

At the next time $t+1$, the state $z_j^{(t+1)}$ of neuron j in the input layer can be obtained from the output value $z_j'^{(t)}$ of neuron j in the output layer. This is defined as *Eq.* (4-2).

$$z_j^{(t+1)} = z_j'^{(t)} \quad (4\text{-}2)$$

Then, the states of neurons are changed repeatedly until the system reaches a stable state.

5. Performance Evaluations

Some images are used as examples to be compressed and decompressed by our neural network approaches. The value of *PSNR* is calculated and used to evaluate the system

performance. The value of $PSNR$ is defined as:

$$PSNR = 20\log_{10}\left(\frac{B}{rms}\right), \qquad (5\text{-}1)$$

where B is the maximum value of gray level (B is assigned to 255 in this paper), and rms is the root mean square of the distance between the original image and the decompressed image. The rms is defined as :

$$rms = \sqrt{\frac{\sum_{i=1}^{N}\left(z_i^{true} - z'_i\right)^2}{N}}, \qquad (5\text{-}2)$$

where z_i^{true} is the gray level of the pixel i in the original image, z'_i is the gray level of the pixel i in the decompressed image and N is the total number of pixels of this image. Then, the larger $PSNR$ is, the better quality of image displays.

5.1. The linear model

Different learning rates are selected to evaluate the quality of linear model. The experimental results are shown in figure 5-1.

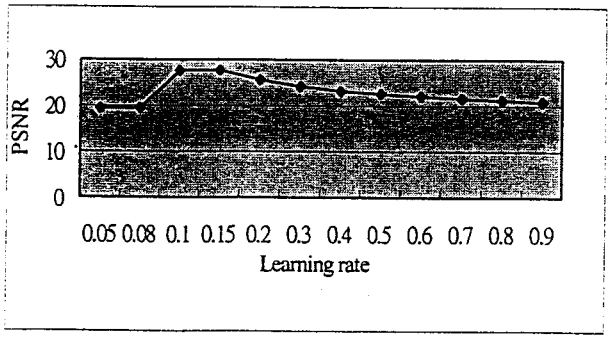


Figure 5-1. The relationship of learning rate and PSNR on the linear model.

According to the results of figure 5-1, we obtain the better quality and the smaller size of image when the learning rate is 0.1.

5.2. The nonlinear model

The values of input range and learning rate affect the quality of images during image decompression in nonlinear model. Four different input ranges (0.1, 0.2, 0.5, 0.9) and various learning rates are selected to simulate and results are shown in figure 5-2.

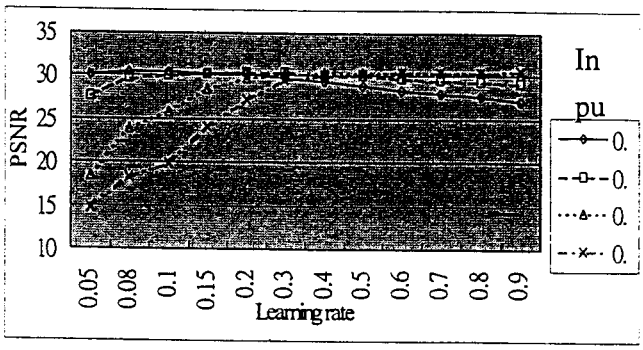


Figure 5-2. The relationship of learning rate and PSNR on the nonlinear model.

Figure 5-2 shows that the better quality of images are obtained by selecting the learning rates between 0.2~0.3, and $PSNR$s of images are less sensitive when input ranges are smaller (i.e., 0.1 or 0.2).

5.3. The comparisons of linear model and nonlinear model

There are many images used as examples to be compressed and decompressed by these two approaches. Experimental results shown that the nonlinear model of neural network gets better quality ($PSNR$) than the linear model, and the sizes of compressed images approaches the traditional PIFS method. In addition, our method can be executed in parallel.

6. Conclusions

In image compression and decompression, fractal theory can obtain a high compression-ratio and a low loss-ratio. But it is limited by its tremendous computations for determining the fractal code to perform the image decompression.

In this paper, we propose neural network approaches to implement the **PIFS** for image compression and decompression. Experiment results show that our neural network approaches can obtain high quality image, and the compression-ratio is as good as the traditional **PIFS** method. In addition, the compression and decompression of our approaches can be operated in parallel. Then, our method is more practicable than other approaches for its parallel processing ability.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital image processing* (Addison-Wesley, Reading, MA, 1993).

[2] G. M. Davis, A wavelet-based analysis of fractal image compression, *IEEE Trans. on Image Processing* 7 (3) (1998) 141-154.

[3] F. G. B. D. Natale, G. S. Desoli, D. D. Giusto, and G. Vernazza, Polynomial approximation and vector quantization: a region-based integration, *IEEE Trans. on Communication* 43 (2-4) (1998) 198-206.

[4] M. Nelson, *The data compression book* (2nd ed., M&T Books, New York, 1996).

[5] M. F. Barnsley and A. D. Sloan, A better way to compress images, *Byte* 13 (1) (1988) 215-223.

[6] Y. Fisher, *Fractal image compression* (Springer Verlag, New York, 1994).

[7] M. F. Barnsley, *Fractals everywhere* (Academic Press, New York, 1992).

[8] K. J. Falconer, *The geometry of fractal sets* (Cambridge University Press, Cambridge, UK, 1985).

[9] B. Mandelbort, *The fractal geometry of nature* (San Francisco, CA: freeman, 1982).

[10] M. F. Barnsley, A. Jacquin, L. Reuter, and A. D. Sloan, Harnessing chaos for image synthesis, *Computer Graphics* 22 (4) (1988) 131-141.

[11] S. Demko, L. Hodges and B. Nayloy, Construction of fractal objects with iterated function systems, *Computer Graphics* 19 (3) (1985) 271-278.

[12] Z. H. Fu, *Chaos and fractals with application on image compression*, Master Thesis of Institute of Electrical Engineering, National Taiwan University, Taiwan, 1992.

[13] Q. Y. Lin, *Fractal and its application to image compression,* Master Thesis of Institute of Electrical Engineering, National Taiwan University, Taiwan, 1993.

[14] A. E. Jacquin, Image coding based on a fractal theory of iterated constructive image transformations, *IEEE Trans. on Image Processing* 1 (1) (1992) 18-30.

[15] A. E. Jacquin, Fractal image coding a review, *Proceedings of the IEEE* 81 (10) (1993) 1451-1465.

[16] G. Zorpette, Fractal: not just another pretty picture, *IEEE Spectrum* 25 (10) (1988) 29-31.

[17] S. F. Chen, *Fractal-based image analysis,* Master Thesis of Institute of Information Science, National Tsing Hua University, Taiwan, 1991.

[18] S. J. Lee, P. Y. Wu and K. T. Sun, Fractal image compression using neural network, *Proc. of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98),* Anchorage, Alaska, (1998) 613-618.

[19] J. Stark, Iterated function systems as neural network, *Neural Network* 4 (5) (1991) 679-690.

[20] J. Stark, A neural network to compute the Hutchinson metric in fractal image processing, *IEEE Trans. on Neural Network* 2 (1) (1991) 156-158.

[21] L. Zhang, B. Zhang, and G. Chen, Generating and coding of fractal graphics by neural network and mathematical morphology methods, *IEEE Trans. on Neural Networks* 7 (2) (1996) 400-407.

[22] S. J. Lee, P. Y. Wu, and K. T. Sun, A study on Fractal image compression using neural network, *Proc. of the 1997 National Computer Symposium (NCS'97),* Tunghai University, Taiwan (1997) B-151-156.

[23] J. J. Hopfield and D. W. Tank, Neural computation of decisions in optimization problems, *Biological Cybemetics,* 52 (1985) 141-152.

[24] R. P. Lippmann, An introduction to computing with neural nets," *IEEE ASSP Magazine* 4 (1987) 4 -23.

[25] M. Mougeot, R. Azencott, and B. Angeniol, Image Compression with back propagation using different cost functions, *Neural Network* 4 (4) (1991) 467-476.

[26] K. T. Sun and H. C. Fu, A hybrid neural network model for solving optimization problems, *IEEE Trans. on Computers* 42 (2) (1993) 218-227.

[27] K. T. Sun and H. C. Fu, A neural network implementation for the traffic control problem on crossbar switch networks, *International Journal of Neural Systems* 3

(2) (1992) 209-218.

[28] D. W. Tank and J. J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit and a linear programming circuit, *IEEE Trans. on Circuits and Systems* 33 (5) (1986) 533-541.

[29] C. P. Lai, *A computer system for locating sequences of CT liver boundary using neural networks and fractal geometry,* Master Thesis of Institute of Electrical Engineering, National Cheng Kung University, Taiwan, 1994.

[30] Y. H. Huang, *Neural network for image compression and seismic signal processing,* Master Thesis of Institute of Information Science, National Chiao Tung University, Taiwan, 1992.

[31] R. Rojas, *Neural Networks: A Systematic Introduction* (Springer Verlag, NY, 1996).

[32] A. J. Crilly, R. A. Eamshaw, and H. Jones, *Fractals and chaos* (Springer Verlag, London, 1991).