ELSEVIER

Neurocomputing 41 (2001) 91–107

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Neural network approaches to fractal image compression and decompression

K.T. Sun[a,]*, S.J. Lee[a], P.Y. Wu[b]

[a]*Institute of Computer Science and Information Education, National Tainan Teachers College, Tainan 700, Taiwan*

[b]*Department of Mathematics and Sciences Education, National Tainan Teachers College, Tainan 700, Taiwan*

Accepted 7 November 2000

Abstract

In image compression technologies, fractal image compression/decompression has the advantages of a high compression ratio and a low loss ratio. However, it requires a great deal of computation, which limits its applications, and so far, no parallel processing technique has been designed and implemented. In this study, we use neural networks to perform a large number of computations in fractal image compression and decompression in parallel. The simulation results show that the quality of images generated by neural networks is similar to that produced using traditional methods, which verifies the high value of our research, which has shown that the neural network technology is useful and efficient when applied to fractal image compression and decompression. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Fractal image compression/decompression; Neural networks; Parallel processing

1. Introduction

Recently, graphical representation in computers has been widely applied in many applications because such representations are meaningful to human beings. However, this approach requires large storage and long transmission time.

The technique of image compression/decompression is useful and important for reducing the storage space and transmission time. In general, these compression technologies can be divided into two types — lossy compression and lossless compression — whether the decompressed image is the same as the original one or not. If the

* Corresponding author.
E-mail addresses: ktsun@ipx.ntntc.edu.tw (K.T. Sun), pywu@ipx.ntntc.edu.tw (P.Y. Wu).

proper loss ratio is allowable, the lossy compression methods can achieve higher compression ratios [11].

Three technologies are usually used in lossy compression: vector quantization (VQ), discrete cosine transformation (DCT) and fractal image compression. The VQ method partitions an image into numerous sub-images and finds some representatives as a codebook from them [6,23]. The DCT method converts the gray levels of an image into other coordinates (e.g., frequency), and then quantizes and stores them [11,24]. By the self-similarity characteristics in an image, an image will converge to an acceptable status after fractal image decompression [3,9].

Unlike VQ, fractal image compression does not require a codebook for the decompression procedure [6]. Fractal image compression is also attractive because of its high compression ratio and low loss ratio properties [24]. Some results have been obtained using this technique: the Hutchinson metric has been proposed to prove the condition of convergence [1,8,21], and Mandelbort has generated images based on fractal theory [21]. By developing a collage theorem and iterated function system (IFS), Barnsley produced a high compression ratio ($10^4:1$–$10^6:1$) fractal code, and this motivated many related researches [1-3,7,10,19]. However, fractal code cannot be generated automatically using IFS [4,14,15,19,24,33]. Jacquin proposed a partitioned iterated function system (PIFS) to improve IFS so that the fractal code can be determined automatically [14,15]. However, a great deal of computation is also required.

Neural network technology is new and useful, and has been successfully used in many scopes [5,12,13,16-18,20,22,25,27-32]. Stark was the first to propose application of the neural networks to IFS [5,27,28]. His method, based on the Hopfield neural network, solves the linear progressive problem and obtains the Hutchinson metric quickly [27,31]. However, his neural network approach only works with the IFS decompression procedure.

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

2. Review of research on the partitioned iterated function system

2.1. Basic concepts of fractal image compression

The basic idea of fractal image compression is to use the characteristics of self-similarity in an image. In Fig. 1(a), the triangle can be divided into three sub-images, as

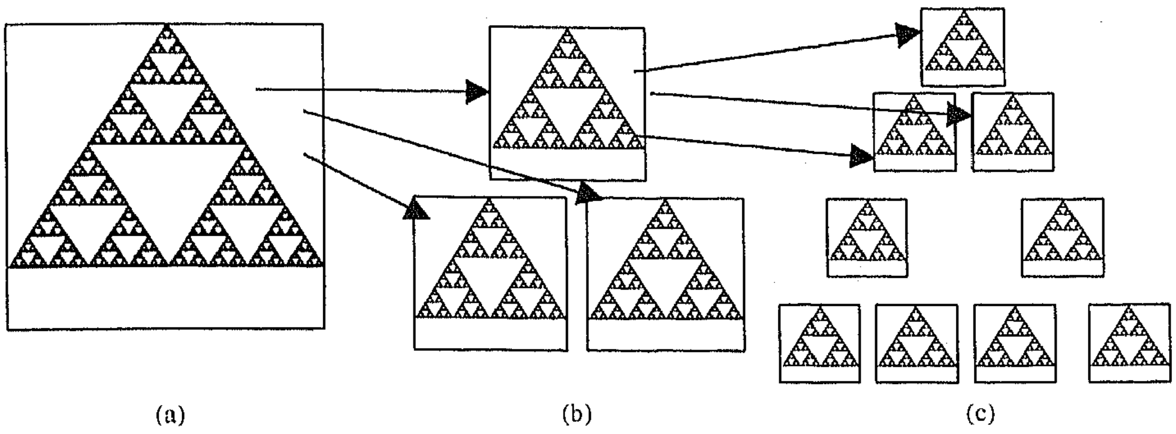


Fig. 1. Three transformation functions show self-similarity in an image. (a) The original image. (b) Partition into 3 similar sub-images after one iteration. (c) Partition into 9 similar sub-images after two iterations.

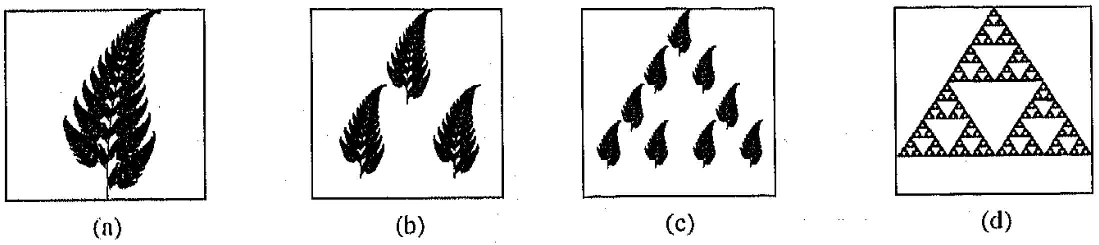


Fig. 2. The decompression procedure for a fractal image. (a) The initial image. (b) After one iteration. (c) After two iterations. (d) After fifteen iterations.

shown in Fig. 1(b). All of these sub-images are the same as the original image except that the size has been reduced 75%, and they can be partitioned into still smaller parts as shown in Fig. 1(c). The smaller parts are also similar to the sub-images. These relationships exist continuously between sub-images as partition operations are performed repeatedly. Then, we only determine the transformation functions which we need in order to map the original image to the sub-images. For example, in Fig. 1, three transformation functions are used to reduce an image into three sub-images with one quarter the size of the original image. And then one sub-image is put on the upper side, one on the lower right-hand side, and one on the lower left-hand side of the original image, respectively. Therefore, the original image (the triangle) can be decompressed using these transformation functions.

When the transformation functions have been obtained, any image can be used as the initial image for the decompression procedure, and then the original image can be generated after many decompression iterations. For example, we can use the "fern" image as the initial image, and the triangle (original image) can be generated after 15 iterations by applying the transformation functions (shown in Eq. (1) and Fig. 2).

Three mapping transformations for Fig. 1 are shown in Eq. (1), and this procedure is called the iterated function system (IFS)

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

Here, (x, y) is the coordinate of the original image, and (x', y') is the coordinate of the transformed image. As a result, only three transformation functions, $\{\tau_1, \tau_2, \tau_3\}$ (also called fractal code), are stored instead of the image data.

2.2. Partitioned iterated function system (PIFS)

For Fig. 3(a), it is almost impossible for finding the fractal code of IFS. However, we can find some similarities between blocks of sub-images. There are two pairs of blocks, which are similar to each other, as shown in Fig. 3(b). One pair is part of a hat and part of a shoulder, and the other pair is the smaller part and a large part of the face.

When we consider the gray level of an image, an additional dimension is added. The transformation function will become that in Eq. (2)

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

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.

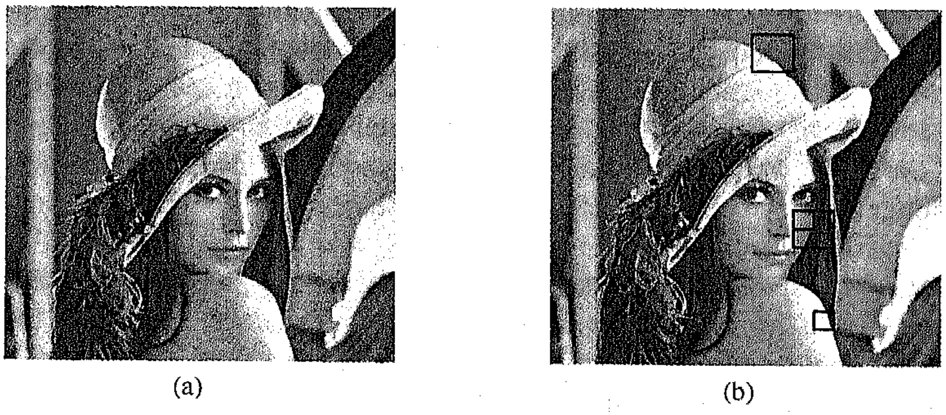

(a) (b)

Fig. 3. There are some similarities between the sub-images in the image Lena. (a) The original image of Lena. (b) Two pairs of blocks similar in shape.

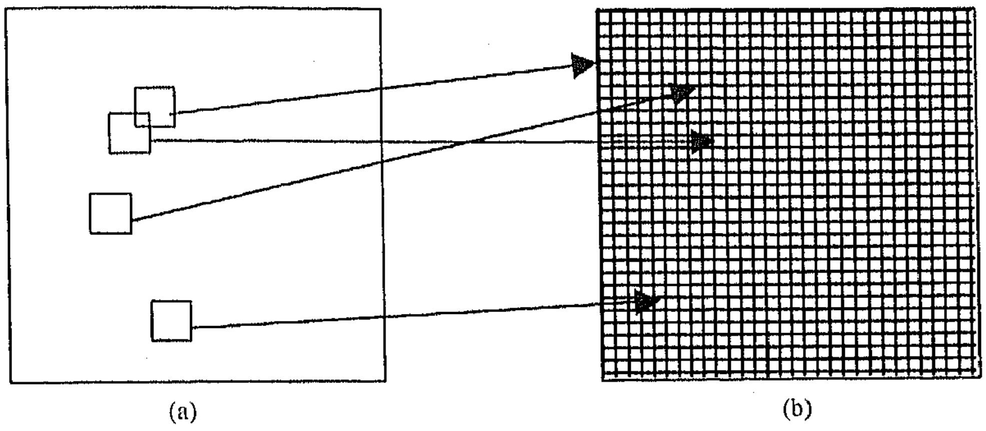


Fig. 4. The concept of PIFS. (a) Overlapping and larger sub-images. (b) Nonoverlapping and smaller sub-images.

Fig. 4 shows the concept of PIFS. Two identical images are partitioned and compared. Each non-overlapping sub-image in 4(b) will need a τ_i to transform a larger and similar sub-image from 4(a) to 4(b).

If we choose a size for the sub-images in Fig. 4(a) that is 4 times (two times the length of height and width, respectively) of the sub-images in Fig. 4 (b), then Eq. (2) will become Eq. (3):

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \tau_i\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & S_i \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}. \tag{3}$$

These transformation functions are the fractal codes that are used to represent the compressed image and will be used in decompressing process. Based on the concept of quadtree partitioning [9,26], the steps in the PIFS method [14,15] are as follows:

(1) Set a threshold value e_c for the error and a minimum size $r_{\min}$ for ranges. (This error is defined as the average of the absolute difference of gray levels of pixels between the range and the corresponding domain. In the domain, the gray levels of every four pixels are averaged and compared with the gray level of the corresponding single pixel in the range. The lower the value of e_c, the higher the similarity.)

(2) Divide the whole image into 4 non-overlapping sub-images (ranges) with one quarter the size of the original image.

(3) For each range i, a domain j (4 times the size of i) with the least error ($\leq e_c$) is found from all domains. Then, a transformation function τ_i is determined for the range i. By computing the differential equation for the transformation function, the contrast s_i and brightness o_i can be determined so as provide a minimum error for the transformation function τ_i.

(4) If the range i cannot provide a similar domain j (i.e., the error between i and j is greater than the threshold value e_c), then the range i is divided into 4 equal sized sub-images, and the size of each one is greater than or equal to $r_{\min}$. Go to step (3) to find the transformation function for each divided sub-image (range).
(5) If the size of range i is equal to $r_{\min}$, and if no similar domain can be found, then the range i is not divided continuously, and a domain j with the least error is selected. In this case, the transformation function τ_i is also determined even if the error between i and j is greater than the threshold value e_c.

Using the concept of quadtree partitioning, the PIFS can effectively find the transformation functions for image compression. However, this is a sequential approach to solve the differential equation for the transformation function. In this paper, we will propose a neural network approach that can generate a compressed image that is similar quality. This approach is very attractive to the parallel processing.

3. Applying neural networks to fractal image compression

We propose use of two different neural network models to implement fractal image compression and decompression. The architectures of these two models are similar except for the transformation functions, as illustrated in Fig. 5. Each pixel of an image is processed by a neuron, and the gray level of the pixel is represented by the state of the neuron. An image is duplicated, creating two images, each one is divided into many sub-images, called domains and ranges, each pixel in a domain corresponds to an input neuron, and each pixel in a range corresponds to an output neuron. To each output neuron, four input neurons are connected. Therefore, each output neuron j is connected to four input neurons i, $i+1$, $i+2$ and $i+3$. The output value z'_j of

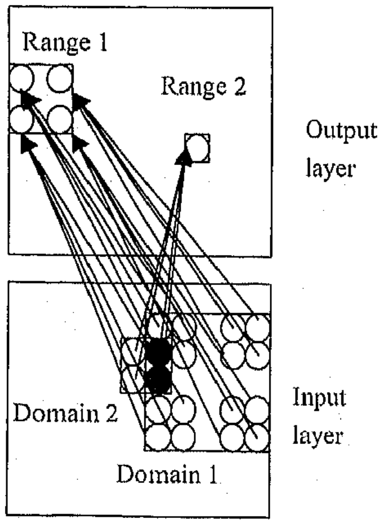


Fig. 5. The architecture of the proposed neural network for implementing fractal image compression.

neuron j is determined by the values Z_i, Z_{i+1}, Z_{i+2}, Z_{i+3}, the corresponding 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 in Eqs. (4) and (5), respectively

$$z'_j = O_j\left(\sum_{k=i}^{i+3} w_{jk} \times z_k - \theta_j\right) \quad \text{linear model,} \tag{4}$$

$$z'_j = O'_j\left(\frac{1}{B}\sum_{k=i}^{i+3} w_{jk} \times z_k - \theta_j\right) \quad \text{nonlinear model,} \tag{5}$$

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

The learning procedure in the neural network approach is based on quadtree partitioning [9,26], which is also used in PIFS. The detailed steps are as follows:

(1) Set a threshold value e_c for the error and a minimum size $r_{\min}$ for the ranges.
(2) Divide the image into many non-overlapping sub-images with 32×32 set as the initial size of the ranges.
(3) For each range i, find a domain j (4 times the size of i) where the error between i and j is less than or equal to the threshold value e_c. Then, determine a transformation function τ_i for the range i. Update the weights w_{ij}, $\forall$ pixels $\in$ range i and $\forall$ pixels $\in$ domain j, of the neural network using the delta learning rule to tune the contrast s_i and brightness o_i in the transformation function τ_i so that the error can be reduced.
(4) If the range i cannot provide a similar domain j (i.e., the error between i and j is greater than the threshold value e_c), then divide the range i into 4 equal sized sub-images (ranges), where the size of each one is greater than or equal to $r_{\min}$. Go to step (3) to find the transformation function for each divided range.
(5) If the size of range i is equal to $r_{\min}$, and if no similar domain can be found, then the range i is not divided continuously, and a domain j with the least error is selected. In this case, the transformation function τ_i is also determined, even if the error between i and j is greater than the threshold value e_c.

Using the delta learning rule of neural networks, a set of transformation functions can be obtained for each range, and they provide a high *PSNR* value for image compression.

3.1. The linear model

Comparing Eqs. (3) and (4), the value Z_k in Eq. (4) can be viewed as the gray level Z of a pixel in Eq. (3). The weight W_{jk} and the threshold θ_j in Eq. (4) can be also viewed as one quarter of the contrast S_i and the negative value of the brightness O_i in Eq. (3), respectively. Then, the linear neural network approach (Eq. (4)) can be used to

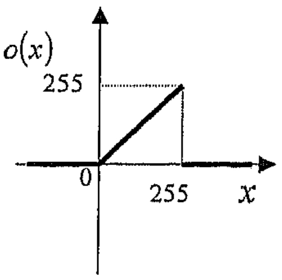


Fig. 6. The activation function of neuron in the linear model.

perform computation of PIFS (Eq. (3)), and the activation function O_j is defined in the following equation:

$$O_j(x) = \begin{cases} x & \text{when } 0 \le x \le 255, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Fig. 6 shows a graphic representation of Eq. (6).

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

$$\delta_j = z_j^{\text{true}} - z'_j, \tag{7}$$

where z'_j is the output value of the 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 by Eq. (8)

$$\Delta W_{jk} = \eta \times \delta_j / z_k, \quad k = i \sim i+3, \tag{8}$$

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

$$\Delta\theta_j = \eta \times \delta_j. \tag{9}$$

The learning procedure is repeated until the output values of the proposed neural network are acceptable.

3.2. The nonlinear model

In the nonlinear model, the activation function O'_j is defined as in Eq. (10), which is a composition function

$$O'_j(x) = DeNor(Sigmoid(x)), \tag{10}$$

where

$$Sigmoid(x) = \frac{1}{(1 + e^{-x})}, \tag{11}$$

$$DeNor(x) = K \times (x - \alpha). \tag{12}$$

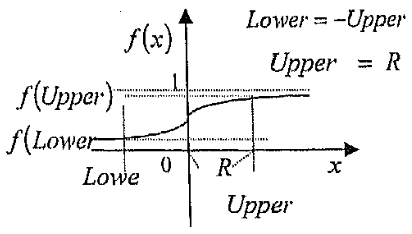


Fig. 7. The activation function of neuron in the nonlinear model.

The values of K and α in Eq. (12) are constants and are defined as

$$K = \frac{B}{(Sigmoid(Upper) - Sigmoid(Lower))}, \tag{13}$$

$$\alpha = Sigmoid(Lower). \tag{14}$$

We define an input range, $2R$, in order to prevent the output of Eq. (11) from being trapped into saturation states. Then, the difference between the maximum value and the minimum value of x is $2R$ (i.e., *Upper–Lower*). Therefore, the output range of the sigmoid function is equal to [*Sigmoid*(*Lower*), *Sigmoid*(*Upper*)]. Fig. 7 shows these relationships.

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

$$\delta_j = \frac{z'_j(B - z'_j)(z_j^{\text{true}} - z'_j)}{B'}. \tag{15}$$

In Eq. (15), all the values of z'_j, B and z_j^{true} are in the range [0, 255]. Then, Eq. (15) can be divided by B^3 to keep the output value in the range $[-1, 1]$. Similarly, the updated weight ΔW_{jk} can be derived by Eq. (16)

$$\Delta w_{jk} = \frac{\eta \times \delta_j \times z_k}{B}, \quad k = i \sim i + 3. \tag{16}$$

The steps for learning weights are repeated until the output values of the neurons are acceptable.

4. Applying the neural network to fractal image decompression

The architecture of our neural network for performing image decompression is shown in Fig. 8, which is similar to Fig. 5 except that the outputs of the neurons in the output layer will feed back to the corresponding neurons in the input layer. The trained weights and threshold (i.e., the fractal code of PIFS) were determined during fractal image compression.

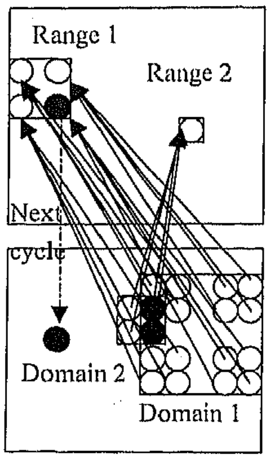


Fig. 8. The architecture of the neural network for fractal image decompression. The output of each neuron in the output layer feeds back to the corresponding neuron with the same position index in the input layer at the next iteration.

The output state $z_j'^{(t)}$ for image decompression is defined in Eq. (17).

$$z_j'^{(t)} = \begin{cases} O_j\left(\sum_{k=i}^{i+3} W_{jk} \times z_k^{(t)} - \theta_j\right) & \text{defined in linear model,} \\ O_j'\left(\frac{1}{B}\sum_{k=i}^{i+3} W_{jk} \times z_k^{(t)} - \theta_j\right) & \text{defined in nonlinear model.} \end{cases} \tag{17}$$

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 in Eq. (18)

$$z_j^{(t+1)} = z_j'^{(t)}. \tag{18}$$

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

5. Performance evaluations

Some images were compressed and decompressed using our neural network approach. The threshold value e_c for the error between two sub-images was set to 2. To find the similarity characteristics in the sub-images, the sizes of the sub-images are ranges from 64×64 to 8×8 in the domain and 32×32 to 4×4 in the range (i.e., $r_{\min} = 4\times 4$). The maximum complexity of the neural network was (64×64) (input layer) $\times (32\times 32)$ (output layer). The value of *PSNR* was calculated and used to evaluate the system performance. The value of *PSNR* was defined as

$$PSNR = 20\log_{10}\left(\frac{B}{rms}\right), \tag{19}$$

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

$$rms = \sqrt{\frac{\sum_{i=1}^{N} (z_i^{\text{true}} - z_i')^2}{N}}, \tag{20}$$

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

5.1. The linear model

The attributes of the Lena image are shown in Table 1. Different learning rates were selected to evaluate the quality of compressed image using a linear model. The experimental results are shown in Fig. 9 and Table 2.

According to the results shown in Fig. 9 and Table 2, we obtained the better quality and smaller compressed images when the learning rate of the neural network was 0.1.

5.2. The nonlinear model

The values of the input range and learning rate affect the quality of images during image decompression in the nonlinear model. Four different input ranges (0.1, 0.2, 0.5,

Table 1
The attributes of the Lena image

Attributes	**Lena**	
Image dimension	512×512	
Size of image data (Bytes)	262144	

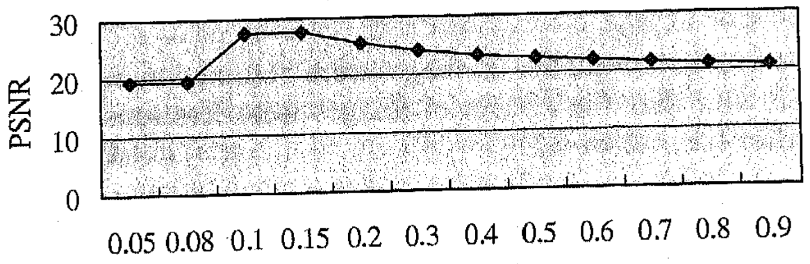
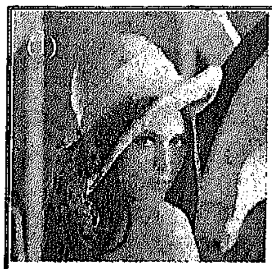
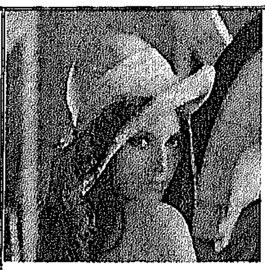
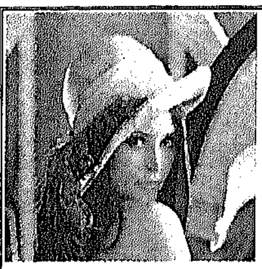
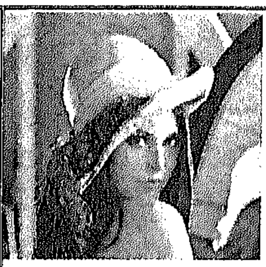
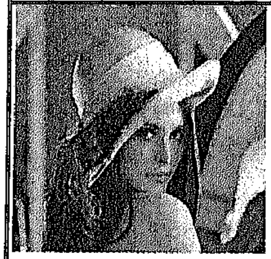
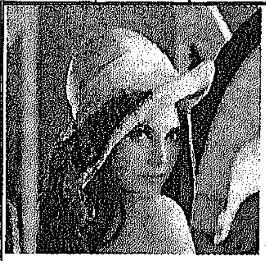
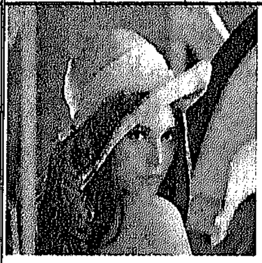
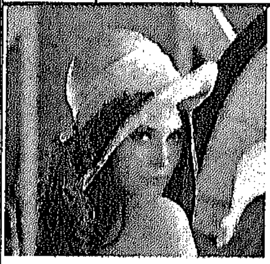
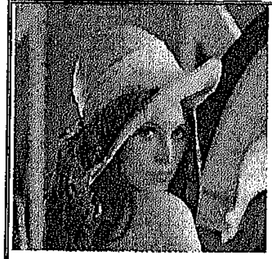
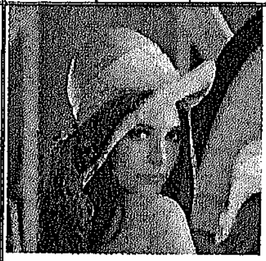
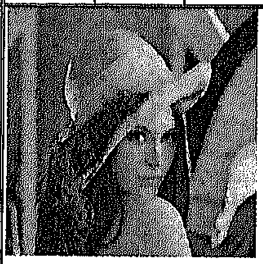
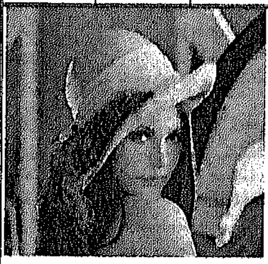


Fig. 9. The relationship between the learning rate and *PSNR* in the linear model.

Table 2
The decompressed images, sizes, and values of *PSNR* under different learning rates for the linear model. (1) The decompressed image. (2) The value of the learning rate. (3) The size of the image after encoding (compressing) (unit: byte). (4) The value of *PSNR*.

(1) 											
0.05 (2)	45659 (3)	19.40 (4)	0.08	45625	19.42	0.1	44370	27.59	0.15	45634	27.67
											
0.2	45642	25.59	0.3	45651	24.12	0.4	45651	23.17	0.5	45651	22.59
											
0.6	45651	22.09	0.7	45651	21.55	0.8	45651	21.13	0.9	45651	20.80

0.9) and various learning rates were selected for simulations and results are shown in Fig. 10.

Fig. 10 shows that better quality of images were obtained by selecting learning rates between 0.2–0.3, and that the *PSNR*s of images were less sensitive when the input ranges are smaller (i.e., 0.1 or 0.2).

5.3. Comparison of the linear model and nonlinear model

Basically, the linear model is similar to IFS (or PIFS) mapping (shown in Eq. (3)). For each transformation function τ_i, the linear model finds the contrast s_i and

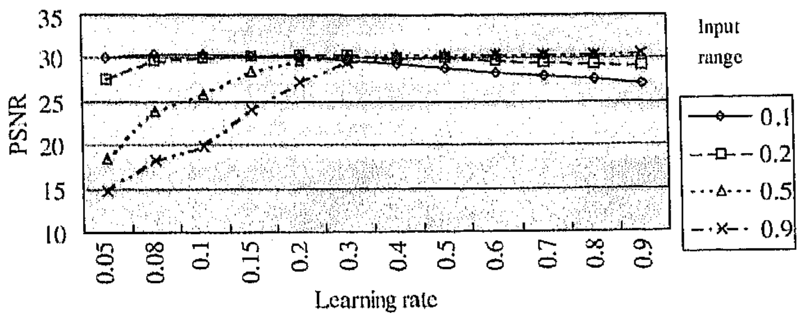


Fig. 10. The relationship of the learning rate and *PSNR* under different input ranges for the nonlinear model.

Table 3
The computation time (second) for image compression and decompression for the Lena image (Table 1) using the linear model, nonlinear model and traditional method (executed on a Pentium II-166 PC with 64 M RAM)

	Linear method	Nonlinear method	Traditional method
Compression	579	3542	36
Decompression	1.81	1.92	1.26

brightness o_i by updating the weights using the linear gradient descent method, which is less flexible and robust than the nonlinear gradient descent method. In addition, the linear model provides lower *PSNR* values relative to the nonlinear model. Figs. 9 and 10 show that the nonlinear model generated higher *PSNR* (i.e., better image quality) value due to its flexibility and robustness. However, the linear model is simpler than the nonlinear model, and the linear model required less computation time (as shown in Table 3). Table 3 shows that the time needed for image compression by our method was much greater than that needed by the traditional method, but that the time needed for image decompression by the different methods is similar. However, the traditional method finds the minimum error by applying the differential equation to the transformation function for the whole sub-image, but the neural network approach updates the weights to find a good transformation function. Therefore, the traditional method performs computations in a sequential mode, but the neural network approach does so in parallel. As a result, for a 32×32 sub-image in a given range, the computation time can be speeded up 32×32 times using the neural network approach if there are enough computing elements in the parallel processing system. In this way, the compression time can be greatly reduced using our method. Six different images were compressed and decompressed using these two proposed approaches and the traditional PIFS method [15]. The results are shown in Table 4. The sizes of the compressed images using the proposed nonlinear model are approaching to that using the traditional PIFS method. This verifies that our methods are useful for the image compression and decompression.

Table 4
Comparison of three fractal image compression methods

Original image		Linear model		Nonlinear model		Traditional method	

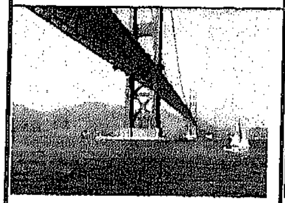
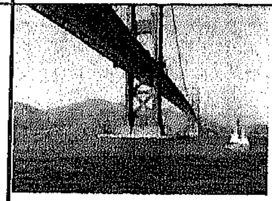
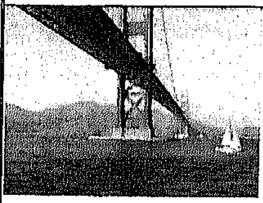
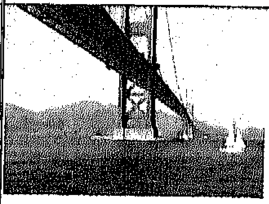
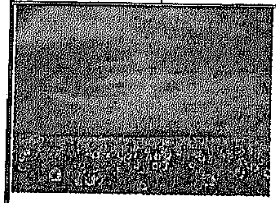
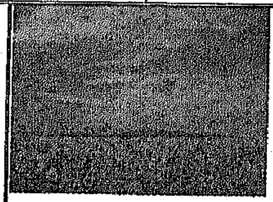
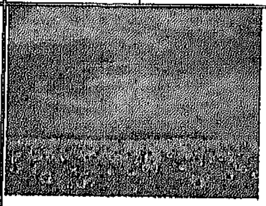
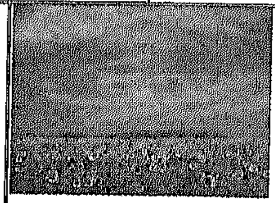
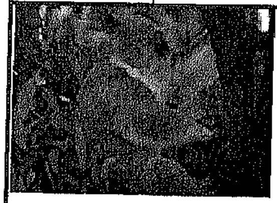
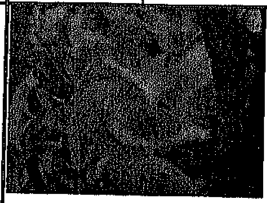
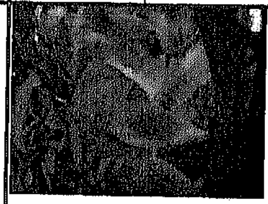
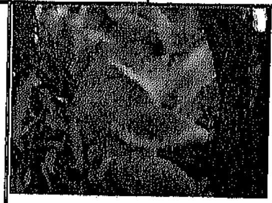
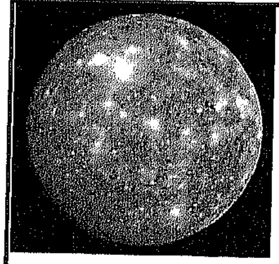
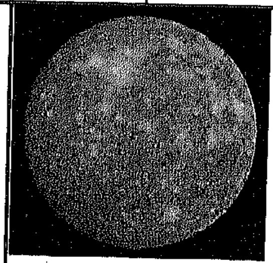
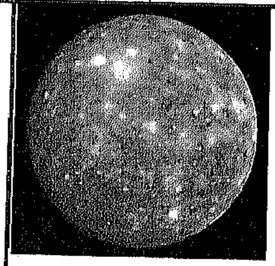
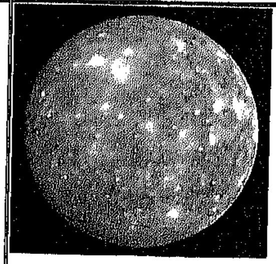
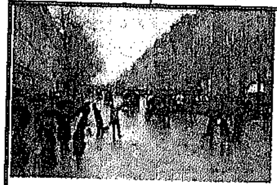
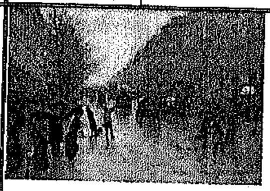
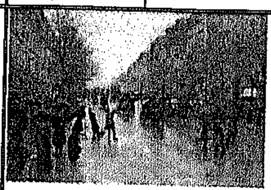
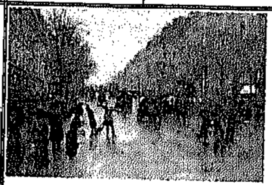
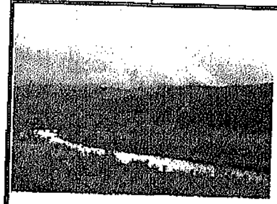
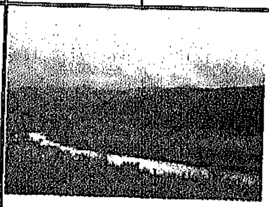
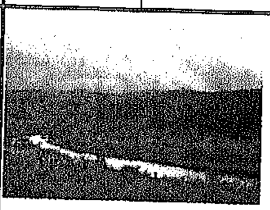
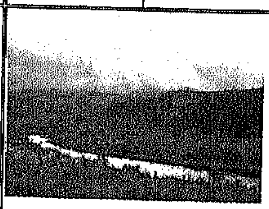

						

Image dimension	Bytes	Bytes	*PSNR*	Bytes	*PSNR*	Bytes	*PSNR*
640x480	307200	42326	26.87	39381	27.27	38022	30.61
							
640x480	307200	44574	23.38	35220	26.46	32034	28.77
							
640x480	307200	52799	23.58	49918	29.46	51580	33.25
							
524x524	274576	37310	20.56	36283	28.79	37594	31.25
							
636x432	274752	49836	24.33	49278	26.44	49322	29.67
							
640x480	307200	50080	27.30	42373	29.33	40453	30.89

6. Conclusion

In image compression and decompression, fractal theory can obtain a high compression ratio and a low loss ratio. However, it is limited by the tremendous number of computations required to determine the fractal code needed to perform image decompression. In this paper, we have proposed neural network approaches to apply PIFS to image compression and decompression. Experiment results show that our neural network approaches can obtain high-quality decompressed images, and that the compression ratio is as good as that obtained by the traditional PIFS method. In addition, the proposed neural nework approaches can be operated in parallel. As a result, image compression and decompression can be performed quickly on a parallel computing system. Our methods can be very useful for image compression and decompression using parallel processing techniques.

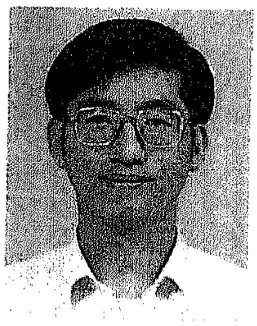
Acknowledgements

This research was supported by the National Science Council of Taiwan, ROC, under the grant NSC 88-2213-E-024-001.

References

[1] M.F. Barnsley, Fractals Everywhere, Academic Press, New York, 1992.
[2] M.F. Barnsley, A. Jacquin, L. Reuter, A.D. Sloan, Harnessing chaos for image synthesis, Comput. Graphics 22 (4) (1988) 131–141.
[3] M.F. Barnsley, A.D. Sloan, A better way to compress images, Byte 13 (1) (1988) 215–223.
[4] S.F. Chen, Fractal-based image analysis, Master Thesis of Institute of Information Science, National Tsing Hua University, Taiwan, 1991.
[5] A.J. Crilly, R.A. Eamshaw, H. Jones, Fractals and Chaos, Springer, London, 1991.
[6] G.M. Davis, A wavelet-based analysis of fractal image compression, IEEE Trans. Image Process. 7 (3) (1998) 141–154.
[7] S. Demko, L. Hodges, B. Nayloy, Construction of fractal objects with iterated function systems, Comput. Graphics 19 (3) (1985) 271–278.
[8] K.J. Falconer, The Geometry of Fractal Sets, Cambridge University Press, Cambridge, UK, 1985.
[9] Y. Fisher, Fractal Image Compression, Springer, New York, 1994.
[10] Z.H. Fu, Chaos and fractals with application on image compression, Master Thesis of Institute of Electrical Engineering, National Taiwan University, Taiwan, 1992.
[11] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1993.
[12] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biol. Cybemet. 52 (1985) 141–152.
[13] 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.
[14] A.E. Jacquin, Image coding based on a fractal theory of iterated constructive image transformations, IEEE Trans. Image Process. 1 (1) (1992) 18–30.
[15] A.E. Jacquin, Fractal image coding a review, Proc. IEEE 81 (10) (1993) 1451–1465.
[16] 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.

[17] S.J. Lee, P.Y. Wu, K.T. Sun, A study on Fractal image compression using neural network, Proceedings of the 1997 National Computer Symposium (NCS'97), Tunghai University, Taiwan, 1997, pp. B-151–156.
[18] S.J. Lee, P.Y. Wu, K.T. Sun, Fractal image compression using neural network, Proceedings of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98), Anchorage, Alaska, 1998, pp. 613–618.
[19] Q. Y. Lin, Fractal and its application to image compression, Master Thesis of Institute of Electrical Engineering, National Taiwan University, Taiwan, 1993.
[20] R.P. Lippmann, An introduction to computing with neural nets, IEEE Accoust. Speech Signal Process. Mag. 4 (1987) 4–23.
[21] B. Mandelbort, The Fractal Geometry of Nature, Freeman, San Francisco, CA, 1982.
[22] M. Mougeot, R. Azencott, B. Angeniol, Image Compression with back propagation using different cost functions, Neural Network 4 (4) (1991) 467–476.
[23] F.G.B.D. Natale, G.S. Desoli, D.D. Giusto, G. Vernazza, Polynomial approximation and vector quantization: a region-based integration, IEEE Trans. Commun. 43 (2–4) (1998) 198–206.
[24] M. Nelson, The Data Compression Book, 2nd Edition, M&T Books, New York, 1996.
[25] R. Rojas, Neural Networks: A Systematic Introduction, Springer, New York, NY, 1996.
[26] E. Shusterman, M. Feder, Image compression via improved quadtree decompression algorithm, IEEE Trans. Image Process. 3 (6) (1994) 207–215.
[27] J. Stark, A neural network to compute the Hutchinson metric in fractal image processing, IEEE Trans. Neural Network 2 (1) (1991) 156–158.
[28] J. Stark, Iterated function systems as neural network, Neural Network 4 (5) (1991) 679–690.
[29] K.T. Sun, H.C. Fu, A neural network implementation for the traffic control problem on crossbar switch networks, Int. J. Neural Systems 3 (2) (1992) 209–218.
[30] K.T. Sun, H.C. Fu, A hybrid neural network model for solving optimization problems, IEEE Trans. Comput. 42 (2) (1993) 218–227.
[31] D.W. Tank, J.J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit and a linear programming circuit, IEEE Trans. Circuits Systems 33 (5) (1986) 533–541.
[32] L. Zhang, B. Zhang, G. Chen, Generating and coding of fractal graphics by neural network and mathematical morphology methods, IEEE Trans. Neural Networks 7 (2) (1996) 400–407.
[33] G. Zorpette, Fractal: not just another pretty picture, IEEE Spectrum 25 (10) (1988) 29–31.



K.T. Sun received the B.S. degree in information science from Tunghai University in 1985 and the M.S. and Ph.D. degrees in computer science and information engineering from National Chiao-Tung University in 1987 and 1992, respectively. From 1992–1996, he was a Research Associate at the Chung Shan Institute of Science and Technology. Since 1996, he has been involved in the computer science and information education at National Tainan Teachers College, Taiwan, ROC, where he is currently an Associate Professor and the Director of the Department of Computer Science and Information Education. His current research interests are neural network, genetic algorithm, fuzzy set theory, computer-assisted instruction/learning design, and educational measurement.

Dr. Sun won the Drag Thesis Award (Ph.D.) granted by the Acer Co. in 1992.

S.J. Lee received the M.S. degree in computer science and information education from National Tainan Teachers College, Tainan, Taiwan, ROC, in 1998. Since 1998, he has been a primary school teacher in Taipei. His current research interests are neural network, fractal image compression, and education research.

P.Y. Wu received the B.S. degree in mathematics from National Kaohsiung Normal University in 1974, the M.S. degree in mathematics from National Tsing Hua University in 1976, the M.S. degree in electrical engineering from National Cheng Kung University in 1986 and the Ph.D. degree in computer science and information engineering from National Taiwan University in 1994. Since 1996, he has been involved in the mathematics education at National Tainan Teachers College, Taiwan, ROC, where he is currently an Associate Professor in the Department of Mathematics Education. His current research interests are fractal imaging, parallel processing, artificial intelligence (neural network, genetic algorithm, fuzzy set theory, etc.) and computer-assisted instruction/learning design.