中華民國八十六年全國計算機會議

ART 類神經網路之影像壓縮
Image Compression Using ART-Based Neural Networks

蔡賢亮　　孫頌賢　　李錫智
Hsien-Leing Tsai, Sung-Hsien Sun, and Shie-Jue Lee

國立中山大學電機工程學系
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung 80424, Taiwan, ROC

摘要

在類神經影像壓縮領域上，有很多演算法已被提出，然而，這些方法仍有些缺點。在本文章中，我們提出一個架構在 ART 模型上的類神經影像壓縮技術。它是一個混合式的類神經網路架構，包含 ART 及 OUTSTAR 類神經模型，而這個架構的訓練方法，我們採用 ART 和 CP 的混合訓練法則。

關鍵字：影像壓縮、類神經網路、ART。

Abstract

In the neural image compression field, many algorithms have been proposed [2, 3, 1]. However, these neural image compression techniques have some drawbacks. In this paper, we propose a neural approach to image compression, based on the adaptive resonance theory (ART) neural networks [5, 4]. Our architecture is a hybrid neural network consisting of one ART neural network and one outstar neural network, incorporated with a hybrid training algorithm of an ART training algorithm and a counter propagation (CP) training algorithm [6].

Keywords: Image compression, neural network, adaptive resonance theory (ART).

1 Introduction

Image compression is an important technique in many applications such as image storage, telecommunication and multi-media. The image compression techniques are to eliminate the redundancy of image so that the size of storage required to preserve the origin image can be reduced with acceptable degradation. For the existing image compression techniques, we can classify them into three main categories: the dimensionality reduction technique, the categorization technique and the hybrid image compression technique of the above two. Recently, researches on image compression using neural networks have been fruitful [2, 3, 1].

Currently, the image compression techniques using neural networks include: auto-associative multilayer perceptron (AMLP) [2], self-organizing feature map (SOFM) [3], and hybrid neural networks of AMLP and SOFM [1]. However, AMLP, SOFM and the hybrid neural networks of AMLP and SOFM still have some drawbacks: (1) When building a multilayer neural network for solving a problem, we are confronted with the determination of the number of hidden layers and the number of nodes in each hidden layer. Traditionally, we can use a trial-and-error method to search for a good network structure. However, it may be impractical in the real world. (2) For AMLP and the hybrid neural networks containing AMLP, we must use the back propagation algorithm to train the multi-layer perceptron. Our experience shows that it is very time consuming to train a neural network using back propagation algorithm. (3) There are too many parameters that would influence the performance of these neural image compressions such as the learning rate, the momentum rate, the stop criterion, the neighbor size of Kohonen training algorithm, and so on. To overcome the above drawbacks of the existing neural image compression techniques, a fast algorithm which is able to automatically construct a neural network is desired. The adaptive resonance theory (ART) is a good choice to meet these requirements.

In this paper, we propose a neural approach to improve the shortcomings of AMLP, SOFM and the hybrid neural networks of AMLP and SOFM. Our neural approach is based on the ART algorithm. The training algorithm is a two-step hybrid training algorithm. The details will be presented in the following sections.

2 Compression Background

In this section, we introduce basic image compression techniques and how to evaluate the performance of an image compression technique. The diagrammatical drawing of image compression technique is shown in Figure 1.

Generally, the image compression techniques are always performed by compressing small subimages instead of compressing a full image directly. Therefore, we first divide an image X equally into P small $H_{sub} * V_{sub}$ rectangular subimages as shown in Figure 1. H_{sub} and V_{sub} denote the numbers of pixels in each horizontal and vertical line, respectively. We can define the image X and the subimage X^p as

$$X = (X^1, ..., X^p, ..., X^P), \qquad 1 \leq p \leq P,$$
$$X^p = (X^p_{1,1}, ...X^p_{1,H_{sub}}, ...X^p_{V_{sub},H_{sub}}),$$

where the pixel $X^p_{i,j}$ has a gray scale from 0 to 255, for i=0,...,V_{sub} and j=0,...,H_{sub}. Then these subimages X^p are input to the encoding end one by one. The outputs at the encoding end represent the compressed image.

To reconstruct an image, we can transmit its compressed subimages to the decoding end one by one. The outputs of the decoding end are the decompressed image. We assume the reconstructed subimage of the original subimage X^p is denoted as Y^p. Then we have

$$Y^p = (Y^p_{1,1}, ..., Y^p_{1,H_{sub}}, ..., Y^p_{V_{sub},H_{sub}}), \qquad 1 \leq p \leq P.$$

In order to evaluate the performance of an image compression technique, we must calculate the compression ratio and the distortion between the original image and the decoded image. We adopt the mean square error (MSE) to sum the distortion between the original subimage X^p and the decoded subimage Y^p, for $p = 0, ..., P$:

$$MSE = \frac{1}{P * H_{sub} * V_{sub}} \sum_{p=1}^{P} \sum_{i=1}^{V_{sub}} \sum_{j=1}^{H_{sub}} |X^p_{i,j} - Y^p_{i,j}|^2. \quad (1)$$

In image compression, the signal-to-noise ratio (SNR) or peak signal-to-noise ratio (PSNR) is used to evaluate its performance. The signal-to-noise ratio (SNR) is defined as

$$SNR = 10 log_{10} \frac{E[|X^p_{i,j}|^2]}{MSE}, \quad (2)$$
$$E[|X^p_{i,j}|^2] = \frac{1}{P * H_{sub} * V_{sub}} \sum_{p=1}^{P} \sum_{i=1}^{V_{sub}} \sum_{j=1}^{H_{sub}} |X^p_{i,j}|^2.$$

The peak signal-to-noise ratio (PSNR) is defined as

$$PSNR = 10 log_{10} \frac{peak^2}{MSE}, \quad peak = 255. \quad (3)$$

The compression ratio (CR) is defined as

$$CR = \frac{T}{P * H_{sub} * V_{sub}}, \quad (4)$$

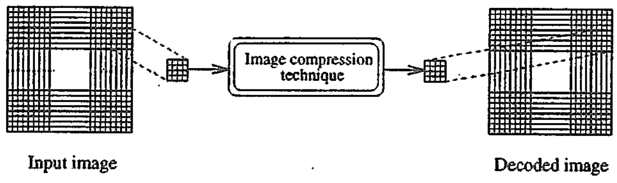

Figure 1: Image compression.

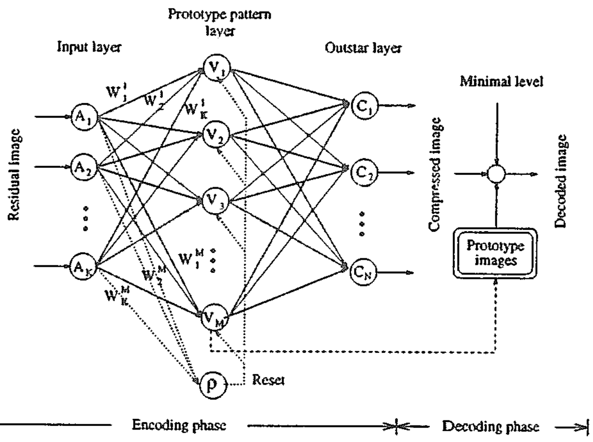

Figure 2: Our neural architecture.

where T is the number of bits required to encode a compressed image. It may include the bits of the compressed codes of all subimages and a codebook.

3 Our Neural Approach

The framework of our neural approach is based on ART1 [5] and fuzzy ARTMAP [4]. It is a hybrid neural network combined by one ART neural network and one outstar neural network in a serial way. Our training algorithm is also a hybrid training algorithm.

From the viewpoint of image compression, our neural approach is a categorization image compression technique. Our approach uses vector quantization to generate a codebook of prototype images. This behavior is similar to SOFM. However, our approach can automatically construct a neural network.

3.1 Our Neural Architecture

Our neural architecture is a three-layer hybrid neural network consisting of one input layer, one prototype pattern layer, and one outstar layer as shown in Figure 2. The input layer and the prototype pattern layer form an ART neural network. The outstar layer is an outstar neural network.

In the input layer, the input nodes receive input patterns and fan out them directly to all prototype pattern nodes in the prototype pattern layer.

The prototype pattern layer is responsible for storing the prototype patterns that are extracted from training patterns. Moreover, it is also responsible for monitoring the classification of input patterns. Because only one of these nodes will be activated by each input pattern, we can regard this layer as a MAXNET. This layer is the output end of the ART neural network. Each prototype pattern node preserves a prototype pattern and represents one category. This preserved prototype pattern is the mean of all training patterns that are classified to this category. Thus, we can use this prototype pattern to represent these similar input patterns.

Each prototype pattern node has a set of category weights W_j^i between it and the input layer. The category weights are used to monitor the classification of input patterns. They have a geometric interpretation as a category hypercube. If an input pattern locates inside the category hypercube of a prototype pattern node, then this input pattern may be classified to the category with a high probability. Furthermore, there is a vigilance controller ρ in the prototype pattern layer to monitor the training process. The magnitude of a vigilance controller represents the cover range of a pattern node. The larger it is, the larger cover range it has. In other words, the larger, the more general. If the above input pattern also passes the test of this vigilance controller then we say that this input pattern can be combined to the category of this prototype pattern node.

To calculate the size of a network, we define a *prototype subnet* as the set of input nodes, one prototype pattern node, and the links between the above two as shown in Figure 2.

The outstar layer is an encoding layer. This layer is responsible for encoding the index of the activated prototype pattern node to a corresponding binary code. This binary code is stored in the code weights between this activated prototype pattern node and the outstar layer. Moreover, each prototype pattern node has a different binary code. The binary code is the compressed image of an input image.

3.2 The Training/Encoding Algorithm

From the above description, it is easily seen that our neural architecture is similar to the counter propagation network (CPN) [6]. Naturally, our training algorithm is a two-step hybrid training algorithm. In the first step, an unsupervised ART training algorithm is used to train the first and second layers. In the second step, we train the outstar layer by using the counter propagation training algorithm.

In fact, the unsupervised ART training algorithm is a variation of the fuzzy ARTMAP and ART1. It is an incremental training algorithm and can automatically construct a neural network step by step. In the beginning, the architecture of the network to be trained only has input nodes in the input layer. After trained, the architecture of this network is similar to that shown in Figure 2.

To encode an image X, we divide the image X equally into P small rectangular subimages as shown in Figure 1. The image X can be denoted as

$$X = (X^1, X^2, ..., X^p, ..., X^P).$$

For each subimage X^p, we assume the numbers of pixels in each horizontal and vertical line are H_{sub} and V_{sub}, respectively. Then, the subimage X^p can be denoted as

$$X^p = (X^p_{1,1}, ..., X^p_{1,H_{sub}}, ..., X^p_{V_{sub},H_{sub}}), \qquad 1 \le p \le P,$$

where pixel $X^p_{i,j}$ has gray levels from 0 to 255, for i=0,...,V_{sub} and j=0,...,H_{sub}. Then, the subimage X^p is input to the network one by one.

Before the training/encoding phase, we perform an image preprocess for the subimage X^p to get the minimal gray level A^p_{min} and the residual image A^p. This image preprocess is defined as follows.

$$A^p_{min} = MIN(X^p_{i,j}), \qquad 1 \le i \le H_{sub}, \quad (5)$$
$$A^p_{(i-1)\times H_{sub}+j} = X^p_{i,j} - A^p_{min}, \qquad 1 \le j \le V_{sub}. \quad (6)$$

In this paper, we call this image preprocess minimum/residual process. The minimal gray level A^p_{min} will be used to reconstruct an image in the decoding end. We only use the residual image vector A^p to train the network. Then, the residual image vector flows through a complement coder which normalizes and expands the original vector into an expanding image vector. The residual image vector A^p and expanding image vector I are defined as

$$\begin{aligned} A^p &= (a^p_1, a^p_2, ..., a^p_d), \qquad d = H_{sub} \times V_{sub}, \\ I &= (A^p, (A^p)^C), \\ &= (a^p_1, a^p_2, ..., a^p_d, 1-a^p_1, 1-a^p_2, ..., 1-a^p_d), \\ &= (I_1, I_2, ..., I_d, I_{d+1}, I_{d+2}, ..., I_N), \qquad N = 2d. \end{aligned}$$

Then, the expanding input flows through the input layer and directly propagates to all the existing prototype pattern nodes.

When the expanding image vector presents to the prototype pattern layer, all the prototype pattern nodes become active to some degree. The activation function for the prototype pattern node V_j is defined as

$$T_j(I) = \frac{|I \wedge W^j|}{\alpha + |W^j|}, \qquad 1 \le i \le N,$$

where W^j is the category weights of the prototype pattern node V^j, and α is a small constant close to 0. The symbol "$\wedge$" is the fuzzy AND operator. This operator is defined as

$$A \wedge B = min(A, B), \qquad 0 \le A, B \le 255. \quad (7)$$

Then we pick out a winner from the prototype pattern layer with the maximum activation.

$$F_w = \max_{arg(j)}(T_j), \qquad 1 \leq j \leq M.$$

This action ensures that the category weight vector of the winning prototype pattern node is most similar to the expanding image vector.

Next, the matching function is used to decide if learning should occur for this winning hidden node. The matching function for the winning prototype pattern node F_w is defined as

$$M_{F_w}(I) = \frac{|I \wedge W^{F_w}|}{|I|}. \qquad (8)$$

In fuzzy logic, the matching function computes the degree that the expanding image vector I is a fuzzy subset of W^{F_w}. By the complement coding, we have $|I| = d$ (dimension of residual image vector). Therefore, we can simplify the matching function to

$$M_{F_w}(I) \;=\; \frac{|I \wedge W_{F_w}|}{d}. \qquad (9)$$

Next, we perform the vigilance test between this matching degree and the vigilance control ρ to decide if this expanding image vector is good enough to match the winning prototype pattern node. If the match degree is smaller than the vigilance parameter, the network is in a state of "mismatch reset". This state indicates that the hidden node is not good enough to encode this expanding input pattern. Therefore, the winning hidden node is suppressed. Then the next winning hidden node is selected and the vigilance test is repeated.

If the match degree is greater than the vigilance parameter, the network is in a state of resonance. Resonance state means that the winning prototype pattern node is good enough to encode this expanding image vector. Then the winning pattern node adjusts its category weights to learn this expanding image vector and the training of this pattern vector is completed. The adjustment of category weights is defined as

$$W_i^{F_w}(new) = \beta(I \wedge W_i^{F_w}(old)) + (1-\beta)W_i^{F_w}(old),$$
$$W_i^{F_w}(new) = W_i^{F_w}(old) \wedge I_i, \qquad \beta = 1. \; (10)$$

The adjustment of prototype pattern and the storing counter are defined as

$$\begin{aligned} PP_i^{F_w}(new) &= PP_i^{F_w}(old) + I_i, \quad 1 \leq i \leq N, \\ PC^{F_w}(new) &= PC^{F_w}(old) + 1. \end{aligned}$$

In case of no winning node passed the vigilance tests, fuzzy ARTMAP creates a new prototype pattern node for this expanding image vector. When this case occurs, it means that all the existing prototype pattern nodes in the network are not good enough to represent this training pattern. Therefore, it is necessary to create a new hidden node for it. The initialization of this new prototype pattern node is

$$\begin{aligned} W_i^{new} &= I_i, \qquad 1 \leq i \leq N, \\ PP_i^{new} &= I_i, \\ PC^{new} &= 1. \end{aligned} \qquad (11)$$

Then the system ends the training epoch of this input pattern and begins that of the next input pattern. The ART training algorithm will train the neural network iteratively until the training neural network is stable.

Finally, we use the counter propagation training algorithm to train the outstar neural network.

When the neural network has been trained by an image X, the compression of this image is also completed. For each subimage X^p, its compressed code consists of the minimal gray level X^p_{min} and the binary code O^p in outstar layer. The prototype patterns that store in the prototype pattern layer will also be preserved with those compressed codes. The prototype patterns are defined as

$$PP_i^j(final) = \frac{PP_i^j}{PC^j}, \quad 1 \leq i \leq N, \quad 1 \leq j \leq M, (12)$$

where M is the number of prototype pattern nodes. In the decoding phase, we must transmit the prototype patterns and those compressed codes to the decoding end.

3.3 The Recalling/Decoding Algorithm

To reconstruct a compressed image X, three kinds of information are critical: the minimal gray level X^p_{min}, the compressed index O^p of all subimages X^p, and the codebook of prototype patterns. In fact, our decoding algorithm just uses a table mapping function and a simple vector adder.

Here, we assume that we are reconstructing the subimage A^p. In the first step, we use the compressed index O^p to look up a prototype pattern in the codebook. The letter "Z" is used to denote this activated prototype pattern. However, "Z" is the decompressed residual image of A^p. Therefore, all elements of this activated prototype pattern plus the minimal gray level A^p_{min} form the decompressed image Y. We define the above behavior as follows.

$$Z = (Z_1, Z_2, ...Z_d), \qquad d = H_{sub} \times V_{sub},$$
$$Y = (Z_1 + A^p_{min}, Z_2 + A^p_{min}, ..., Z_d + A^p_{min}). \quad (13)$$

The vector Y is the decoded subimage of the subimage A^p.

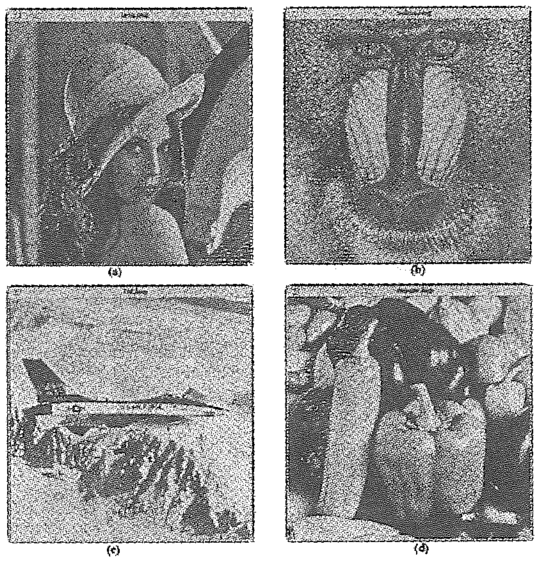

Figure 3: The benchmark images (a) "Lena"; (b) "Baboon"; (c) "Jet"; (d) "Pepper".

	SNR(dB)	CR(bits per pixel)
SOFM	29.02	1.75
AMLP	28.40	1.0
PHNN	29.22	1.03
SHNN	31.10	1.75

Table 1: Performance comparison among SOFM, AMLP, parallel hybrid neural network (PHNN), and serial hybrid neural network (SHNN).

Vigilance ρ	No. of subnets	SNR(dB)	CR(bits per pixel)
0.300	57	25.60	0.65
0.325	89	25.88	0.73
0.350	145	26.67	0.82
0.375	225	27.07	0.85
0.400	357	28.10	0.98
0.425	645	28.76	1.18
0.450	1278	30.94	1.56

Table 2: Image compression performance of our neural approach to the benchmark "Lena".

4 Experiments

In this section, the performances of the SOFM, AMLP, parallel hybrid neural network, serial hybrid neural network, and our neural approach are evaluated. We use four 256-graylevel, 512×512 sized, benchmark images to evaluate performances. For each subimage, the numbers of pixels in each horizontal and vertical line are 4 and 4, respectively. The four benchmark images are "Lena", "Baboon", "Jet", and "Pepper" as shown in Figure 3. We use peak signal-to-noise ratio (PSNR) and compression ratio (CR) to evaluate performances. PSNR is computed by Equation (1). CR is computed by Equation (3).

Performances on the benchmark image "Lena" for SOFM, AMLP, parallel hybrid neural network (PHNN), and serial hybrid neural network (SHNN) are shown in Table 4. The information can be referred to the paper [1]. In the referred paper, however, the computation of CR is not adequate. In the computation of CR, it only takes the number of bits of a compressed image into account. It fails to consider the size of a codebook or a set of decoding weights. Therefore, the values of CR in Table 4 are smaller than the correct ones. In our experiments, we adopt the correct CR to evaluate our performance.

In our neural approach, the vigilance controller ρ is the only parameter that influences the image compression performance. Therefore, our experiments focus on the relationship between the vigilance controller and image compression performance. The image performance of our neural approach on the "Lena", "Baboon", "Jet" and "Pepper" are shown in Tables 2, 3, 4 and 5, respectively. Figures 4, 5, 6, and 7 show the decoded images and the distortion images of the benchmark images "Lena", "Baboon", "Jet", and "Pepper", respectively. To show the distortion images more brightly, we add 128 to the gray level of each pixel in the corresponding figure.

From the results of our experiments, we conclude that the larger the vigilance controller is, the more the subnets are created. Furthermore, it also makes the PSNR better. However, it results in the number of bits per pixel to be larger. Therefore, we can adjust the vigilance controller to meet our requirements to image compression.

Our neural approach is a fast encoding algorithm. It only needs two or three training iterations to encode an image. However, AMLP and the hybrid neural networks containing of AMLP need much more training iterations to meet the stop criterion. Moreover, our approach does not need predecide its network structure by human beings. Therefore, our approach is good for image compression.

References

[1] M. A. Abidi, S. Yasuki and P. B. Crilly. Image compression using hybrid neural networks combining the auto-associative multi-layer per-

Vigilance ρ	No. of subnets	SNR(dB)	CR(bits per pixel)
0.250	113	19.28	0.74
0.275	208	19.88	0.85
0.300	377	20.42	0.99
0.325	644	20.77	1.18
0.350	1066	21.24	1.45
0.375	1631	21.96	1.73
0.400	2453	22.87	2.19

Table 3: Image compression performance of our neural approach to the benchmark "Baboon".

Vigilance ρ	No. of subnets	SNR(dB)	CR(bits per pixel)
0.300	79	24.52	0.83
0.325	122	25.47	0.83
0.350	182	26.23	0.83
0.375	296	26.42	0.83
0.400	459	27.72	1.03
0.425	750	28.97	1.24
0.450	1333	31.09	1.58

Table 4: Image compression performance of our neural approach to the benchmark “Jet”.

Vigilance ρ	No. of subnets	SNR(dB)	CR(bits per pixel)
0.300	76	25.12	0.72
0.325	108	25.68	0.74
0.350	162	25.80	0.82
0.375	247	26.85	0.87
0.400	366	27.99	0.99
0.425	634	29.22	1.18
0.450	1299	30.76	1.57

Table 5: Image compression performance of our neural approach to the benchmark “Pepper”.

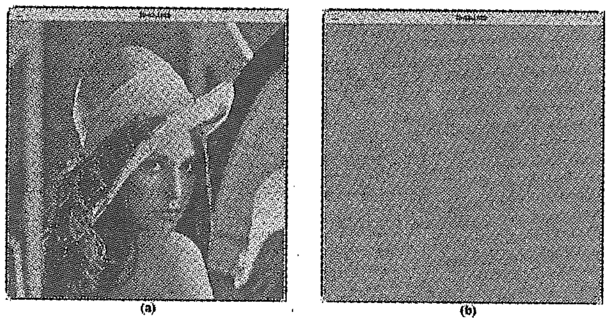

Figure 4: (a) The decoded image “Lena”, and (b) the distortion image.

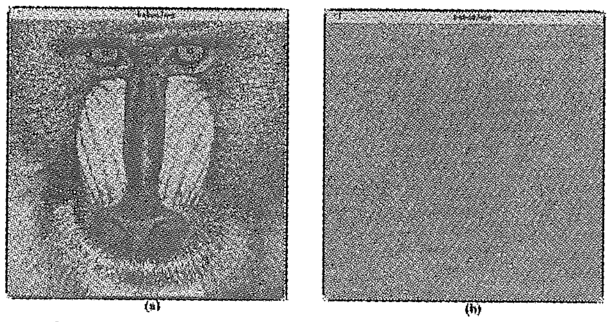

Figure 5: (a) The decoded image “Baboon”, and (b) the distortion image.

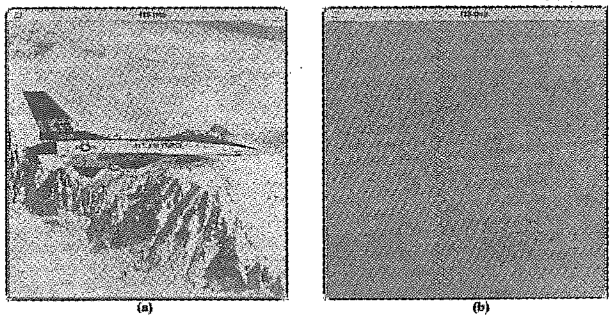

Figure 6: (a) The decoded image “Jet”, and (b) the distortion image.

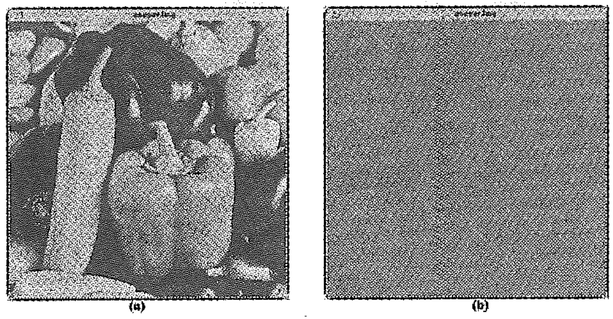

Figure 7: (a) The decoded image “Pepper”, and (b) the distortion image.

ceptron and the self-organizing feature map. *IEEE Transactions on Consumer Electronics*, 40(4):796–811, November, 1994.

[2] G. W. Cottrell, P. Munro and D. Zipser. Image compression by back propagation: an example of extensional programming. *ICS Report 8702, Institute for Cognitive Science, University of California, San Diego*, 1987.

[3] N. M. Nasrabadi and R. A. Feng. Vector quantization of images based upon Konhnen self-organizing feature map. *IEEE International Conference on Neural Networks*, 1:101–108, 1988.

[4] N. Markuzon J. H. Reynold G. A. Carpenter, S. Grossberg and D. B. Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, September 1992.

[5] G. A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *Computer*, pages:35–41, March, 1988.

[6] R. Hecht-Nielsen. Counterpropagation networks. *IEEE International Conference on Neural Networks*, 2:19–32, 1987.