ELSEVIER

Signal Processing: *Image Communication* 14 (1999) 737–760

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Image compression with neural networks – A survey

J. Jiang*

School of Computing, University of Glamorgan, Pontypridd CF37 1DL, UK

Received 24 April 1996

Abstract

Apart from the existing technology on image compression represented by series of JPEG, MPEG and H.26x standards, new technology such as neural networks and genetic algorithms are being developed to explore the future of image coding. Successful applications of neural networks to vector quantization have now become well established, and other aspects of neural network involvement in this area are stepping up to play significant roles in assisting with those traditional technologies. This paper presents an extensive survey on the development of neural networks for image compression which covers three categories: direct image compression by neural networks; neural network implementation of existing techniques, and neural network based technology which provide improvement over traditional algorithms.

Keywords: Neural network; Image compression and coding

1. Introduction

Image compression is a key technology in the development of various multimedia computer services and telecommunication applications such as teleconferencing, digital broadcast codec and video technology, etc. At present, the main core of image compression technology consists of three important processing stages: pixel transforms, quantization and entropy coding. In addition to these key techniques, new ideas are constantly appearing across different disciplines and new research fronts. As a model to simulate the learning function of human brains, neural networks have enjoyed widespread applications in telecommunication and computer science. Recent publications show a substantial increase in neural networks for image compression and coding. Together with the popular multimedia applications and related products, what role are the well-developed neural networks going to play in this special era where information processing and communication is in great demand? Although there is no sign of significant work on neural networks that can take over the existing technology, research on neural networks of image compression are still making steady advances. This could have a tremendous impact upon the development of new technologies and algorithms in this subject area.

* Tel.: +44 1443 482271; fax: +44 1443 482715; e-mail: jjiang@glam.ac.uk

0923-5965/99/$ - see front matter
PII: S0923-5965(98)00041-1

This paper comprises five sections. Section 2 discusses those neural networks which are directly developed for image compression. Section 3 contributes to neural network implementation of those conventional image compression algorithms. In Section 4, indirect neural network developments which are mainly designed to assist with those conventional algorithms and provide further improvement are discussed. Finally, Section 5 gives conclusions and summary for present research work and other possibilities of future research directions.

2. Direct neural network development for image compression

2.1. Back-propagation image compression

2.1.1. Basic back-propagation neural network

Back-propagation is one of the neural networks which are directly applied to image compression coding [9,17,47,48,57]. The neural network structure can be illustrated as in Fig. 1. Three layers, one input layer, one output layer and one hidden layer, are designed. The input layer and output layer are fully connected to the hidden layer. Compression is achieved by designing the value of K, the number of neurones at the hidden layer, less than that of neurones at both input and the output layers. The input image is split up into blocks or vectors of 8×8, 4×4 or 16×16 pixels. When the input vector is referred to as N-dimensional which is equal to the number of pixels included in each block, all the coupling weights connected to each neurone at the hidden layer can be represented by $\{w_{ji}, j = 1, 2, \ldots, K \text{ and } i = 1, 2, \ldots, N\}$, which can also be described by a matrix of order $K\times N$. From the hidden layer to the output layer, the connections can be represented by $\{w'_{ij}: 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant K\}$ which is another weight matrix of order $N\times K$. Image compression is achieved by training the network in such a way that the coupling weights, $\{w_{ji}\}$, scale the input vector of N-dimension into a narrow channel of K-dimension ($K < N$) at the hidden layer and produce the optimum output value which makes the quadratic error between input and output minimum. In accordance with the neural network structure shown in Fig. 1, the operation of a linear network can be described as follows:

$$h_j = \sum_{i=1}^{N} w_{ji}x_i, \quad 1 \leqslant j \leqslant K, \tag{2.1}$$

for encoding and

$$\bar{x}_i = \sum_{j=1}^{K} w'_{ij}h_j, \quad 1 \leqslant i \leqslant N, \tag{2.2}$$

for decoding where $x_i \in [0, 1]$ denotes the normalized pixel values for grey scale images with grey levels [0, 255] [5,9,14,34,35,47]. The reason for using normalized pixel values is due to the fact that neural networks can operate more efficiently when both their inputs and outputs are limited to a range of [0, 1] [5]. Good discussions on a number of normalization functions and their effect on neural network performances can be found in [5,34,35].

The above linear networks can also be transformed into non-linear ones if a transfer function such as sigmoid is added to the hidden layer and the output layer as shown in Fig. 1 to scale the summation down in the above equations. There is no proof, however, that the non-linear network can provide a better solution than its linear counterpart [14]. Experiments carried out on a number of image samples in [9] report that linear networks actually outperform the non-linear one in terms of both training speed and compression performance. Most of the back-propagation neural networks

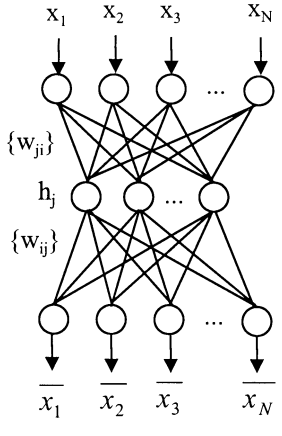

Fig. 1. Back propagation neural network.

developed for image compression are, in fact, designed as linear [9,14,47]. Theoretical discussion on the roles of the sigmoid transfer function can be found in [5,6].

With this basic back-propagation neural network, compression is conducted in two phases: training and encoding. In the first phase, a set of image samples is designed to train the network via the back-propagation learning rule which uses each input vector as the desired output. This is equivalent to compressing the input into the narrow channel represented by the hidden layer and then reconstructing the input from the hidden to the output layer.

The second phase simply involves the entropy coding of the state vector h_j at the hidden layer. In cases where adaptive training is conducted, the entropy coding of those coupling weights may also be required in order to catch up with some input characteristics that are not encountered at the training stage. The entropy coding is normally designed as the simple fixed length binary coding although many advanced variable length entropy coding algorithms are available [22,23,26,60]. One of the obvious reasons for this is perhaps that the research community is only concerned with the part played by neural networks rather than anything else. It is a straightforward thing to introduce better entropy coding methods after all. Therefore, the compression performance can be assessed either in terms of the compression ratio adopted by the computer science community or bit rate adopted by the telecommunication community. We use the bit rate throughout the paper to discuss all the neural network algorithms. For the back propagation narrow channel compression neural network, the bit rate can be defined as follows:

$$\text{bit rate} = \frac{nKT + NKt}{nN} \text{ bits/pixel}, \tag{2.3}$$

where input images are divided into n blocks of N pixels or n N-dimensional vectors; T and t stand for the number of bits used to encode each hidden neurone output and each coupling weight from the hidden layer to the output layer. When the coupling weights are maintained the same throughout the compression process after training is completed, the term NKt can be ignored and the bit rate becomes KT/N bits/pixel. Since the hidden neurone output is real valued, quantization is required for fixed length entropy coding which is normally designed as 32 level uniform quantization corresponding to 5 bit entropy coding [9,14].

This neural network development, in fact, is in the direction of K–L transform technology [17,21,50] which actually provides the optimum solution for all linear narrow channel type of image compression neural networks [17]. When Eqs. (2.1) and (2.2) are represented in matrix form, we have

$$[h] = [W]^{\mathrm{T}}[x], \tag{2.4}$$

$$[\bar{x}] = [W'][h] = [W'][W]^{\mathrm{T}}[x] \tag{2.5}$$

for encoding and decoding.

The K–L transform maps input images into a new vector space where all the coefficients in the new space is de-correlated. This means that the covariance matrix of the new vectors is a diagonal matrix whose elements along the diagonal are eigenvalues of the covariance matrix of the original input vectors. Let e_i and λ_i, $i = 1, 2, \ldots, n$, be eigenvectors and eigenvalues of c_x, the covariance matrix for input vector $\boldsymbol{x}$, and those corresponding eigenvalues are arranged in a descending order so that $\lambda_i \geqslant \lambda_{i+1}$, for $i = 1, 2, \ldots, n-1$. To extract the principal components, K eigenvectors corresponding to the K largest eigenvalues in c_x are normally used to construct the K–L transform matrix, $[A_K]$, in which all rows are formed by the eigenvectors of c_x. In addition, all eigenvectors in $[A_K]$ are ordered in such a way that the first row of $[A_K]$ is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the smallest eigenvalue. Hence, the forward K–L transform or encoding can be defined as

$$[y] = [A_K]([x] - [m_x]), \tag{2.6}$$

and the inverse K–L transform or decoding can be defined as:

$$[\bar{x}] = [A_K]^{\mathrm{T}}[y] + [m_x], \tag{2.7}$$

where $[m_x]$ is the mean value of $[x]$ and $[\bar{x}]$ represents the reconstructed vectors or image blocks. Thus the mean square error between $\boldsymbol{x}$ and $\bar{x}$ is

given by the following equation:

$$e_{ms} = E\{(x-\bar{x})^2\} = \frac{1}{M}\sum_{K=1}^{M}(x_K - \overline{x_K})^2$$

$$= \sum_{j=1}^{n}\lambda_j - \sum_{j=1}^{K}\lambda_j = \sum_{j=K+1}^{n}\lambda_j, \qquad (2.8)$$

where the statistical mean value $E\{\cdot\}$ is approximated by the average value over all the input vector samples which, in image coding, are all the non-overlapping blocks of 4×4 or 8×8 pixels.

Therefore, by selecting the K eigenvectors associated with the largest eigenvalues to run the K–L transform over input image pixels, the resulting errors between the reconstructed image and the original one can be minimized due to the fact that the values of λ's decrease monotonically.

From the comparison between the equation pair (2.4) and (2.5) and the equation pair (2.6) and (2.7), it can be concluded that the linear neural network reaches the optimum solution whenever the following condition is satisfied:

$$[W'][W]^{\mathrm{T}} = [A_K]^{\mathrm{T}}[A_K]. \qquad (2.9)$$

Under this circumstance, the neurone weights from input to hidden and from hidden to output can be described respectively as follows:

$$[W'] = [A_K][U]^{-1}, \qquad [W]^{\mathrm{T}} = [U][A_K]^{\mathrm{T}}, \quad (2.10)$$

where $[U]$ is an arbitrary $K\times K$ matrix and $[U][U]^{-1}$ gives an identity matrix of $K\times K$. Hence, it can be seen that the linear neural network can achieve the same compression performance as that of K–L transform without necessarily obtaining its weight matrices being equal to $[A_K]^{\mathrm{T}}$ and $[A_K]$.

Variations of the image compression scheme implemented on the basic network can be designed by using overlapped blocks and different error functions [47]. The use of overlapped blocks is justified in the sense that some extent of correlation always exists between the coefficients among the neighbouring blocks. The use of other error functions, in addition, may provide better interpretation of human visual perception for the quality of those reconstructed images.

2.1.2. Hierarchical back-propagation neural network

The basic back-propagation network is further extended to construct a hierarchical neural network by adding two more hidden layers into the existing network as proposed in [48]. The hierarchical neural network structure can be illustrated in Fig. 2 in which the three hidden layers are termed as the combiner layer, the compressor layer and the decombiner layer. The idea is to exploit correlation between pixels by inner hidden layer and to exploit correlation between blocks of pixels by outer hidden layers. From the input layer to the combiner layer and from the decombiner layer to the output layer, local connections are designed which have the same effect as M fully connected neural sub-networks. As seen in Fig. 2, all three hidden layers are fully connected. The basic idea is to divide an input image into M disjoint sub-scenes and each sub-scene is further partitioned into T pixel blocks of size $p\times p$.

For a standard image of 512×512, as proposed [48], it can be divided into 8 sub-scenes and each sub-scene has 512 pixel blocks of size 8×8. Accordingly, the proposed neural network structure is designed to have the following parameters:

Total number of neurones at the input layer $= Mp^2 = 8\times64 = 512$.
Total number of neurones at the combiner layer $= MN_h = 8\times8 = 64$.
Total number of neurones at the compressor layer $= Q = 8$.

M = 8
p^2 = 64
N_h = 8

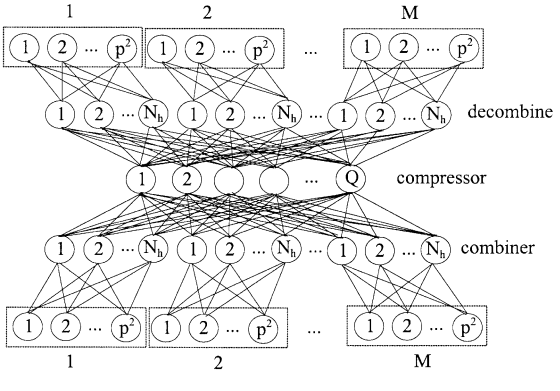

Fig. 2. Hierarchical neural network structure.

The total number of neurones for the decombiner layer and the output layer is the same as that of the combiner layer and the input layer, respectively.

A so called nested training algorithm (NTA) is proposed to reduce the overall neural network training time which can be explained in the following three steps.

Step 1. Outer loop neural network (OLNN) training. By taking the input layer, the combiner layer and the output layer out of the network shown in Fig. 2, we can obtain M 64–8–64 outer loop neural networks where 64–8–64 represents the number of neurones for its input layer, hidden layer and output layer, respectively. As the image is divided into 8 sub-scenes and each sub-scene has 512 pixel blocks each of which has the same size as that of the input layer, we have 8 training sets to train the 8 outer-loop neural networks independently. Each training set contains 512 training patterns (or pixel blocks). In this training process, the standard back-propagation learning rule is directly applied in which the desired output is equal to the input.

Step 2. Inner loop neural network (ILNN) training. By taking the three hidden layers in Fig. 2 into consideration, an inner loop neural network can be derived as shown in Fig. 3. As the related parameters are designed in such a way that $N_h = 8$; $Q = 8$; $M = 8$, the inner loop neural network is also a 64–8–64 network. Corresponding to the 8 sub-scenes each of which has 512 training patterns (or pixel blocks), we also have 8 groups of hidden layer outputs from the operations of step 1, in which each hidden layer output is an 8-dimensional vector and each group contains 512 such vectors. Therefore, for the inner loop network, the training set contains 512 training patterns each of them is a 64-dimensional vector when the outputs of all the 8 hidden layers inside the OLNN are directly used to train the ILNN. Again, the standard back-propagation learning rule is used in the training process. Throughout the two steps of training for both ILNN and OLNN, the linear transfer function (or activating function) is used.

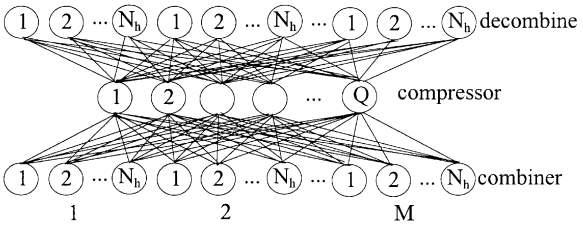

Fig. 3. Inner loop neural network.

Step 3. Reconstruction of the overall neural networks. From the previous two steps of training, we have four sets of coupling weights, two out of step 1 and two out of step 2. Hence, the overall neural network coupling weights can be assigned in such a way that the two sets of weights from step 1 are given to the outer layers in Fig. 2 involving the input layer connected to the combiner layer, and the decombiner layer connected to the output layer. Similarly, the two sets of coupling weights obtained from step 2 can be given to the inner layers in Fig. 2 involving the combiner layer connected to the compressor layer and the compressor layer connected to the decombiner layer.

After training is completed, the neural network is ready for image compression in which half of the network acts as an encoder and the other half as a decoder. The neurone weights are maintained the same throughout the image compression process.

2.1.3. Adaptive back-propagation neural network

Further to the basic narrow channel back-propagation image compression neural network, a number of adaptive schemes are proposed by Carrato [8,9] based on the principle that different neural networks are used to compress image blocks with different extent of complexity. The general structure for the adaptive schemes can be illustrated in Fig. 4 in which a group of neural networks with increasing number of hidden neurones, ($h_{\min}$, $h_{\max}$), is designed. The basic idea is to classify the input image blocks into a few sub-sets with different features according to their complexity measurement. A fine tuned neural network then compresses each sub-set.

Four schemes are proposed in [9] to train the neural networks which can be roughly classified as parallel training, serial training, activity-based training and activity and direction based training schemes.

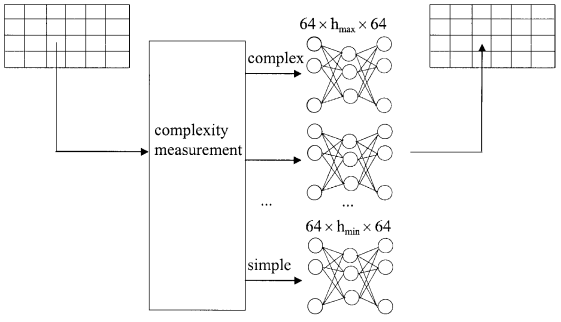

Fig. 4. Adaptive neural network structure.

The parallel training scheme applies the complete training set simultaneously to all neural networks and uses the S/N (signal-to-noise) ratio to roughly classify the image blocks into the same number of sub-sets as that of neural networks. After this initial coarse classification is completed, each neural network is then further trained by its corresponding refined sub-set of training blocks.

Serial training involves an adaptive searching process to build up the necessary number of neural networks to accommodate the different patterns embedded inside the training images. Starting with a neural network with pre-defined minimum number of hidden neurones, $h_{\min}$, the neural network is roughly trained by all the image blocks. The S/N ratio is used again to classify all the blocks into two classes depending on whether their S/N is greater than a pre-set threshold or not. For those blocks with higher S/N ratios, further training is started to the next neural network with the number of hidden neurones increased and the corresponding threshold readjusted for further classification. This process is repeated until the whole training set is classified into a maximum number of sub-sets corresponding to the same number of neural networks established.

In the next two training schemes, extra two parameters, activity $A(P_l)$ and four directions, are defined to classify the training set rather than using the neural networks. Hence the back propagation training of each neural network can be completed in one phase by its appropriate sub-set.

The so called activity of the lth block is defined as

$$A(P_l) = \sum_{\text{even } i,j} A_p(P_l(i,j)), \tag{2.11}$$

$$A_p(P_l(i,j)) = \sum_{r=-1}^{1} \sum_{s=-1}^{1} (P_l(i,j) - P_l(i+r, j+s))^2, \tag{2.12}$$

where $A_p(P_l(i,j))$ is the activity of each pixel which concerns its neighbouring 8 pixels as r and s vary from -1 to $+1$ in Eq. (2.12).

Prior to training, all image blocks are classified into four classes according to their activity values which are identified as very low, low, high and very high activities. Hence four neural networks are designed with increasing number of hidden neurones to compress the four different sub-sets of input images after the training phase is completed.

On top of the high activity parameter, a further feature extraction technique is applied by considering four main directions presented in the image details, i.e., horizontal, vertical and the two diagonal directions. These preferential direction features can be evaluated by calculating the values of mean squared differences among neighbouring pixels along the four directions [9].

For those image patterns classified as high activity, further four neural networks corresponding to the above directions are added to refine their structures and tune their learning processes to the preferential orientations of the input. Hence, the overall neural network system is designed to have six neural networks among which two correspond to low activity and medium activity sub-sets and other four networks correspond to the high activity and four direction classifications [9]. Among all the adaptive schemes, linear back-propagation neural network is mainly used as the core of all the different variations.

2.1.4. Performance assessments

Around the back-propagation neural network, we have described three representative schemes to achieve image data compression towards the direction of K–L transform. Their performances can normally be assessed by considering two measurements. One is the compression ratio or bit rate

which is used to measure the compression performance, and the other is mainly used to measure the quality of reconstructed images with regards to a specific compression ratio or bit rate. The definition of this measurement, however, is a little ambiguous at present. In practice, there exists two acceptable measurements for the quality of reconstructed images which are PSNR (peak-signal-to-noise ratio) and NMSE (normalised mean-square error). For a grey level image with n blocks of size N, or n vectors with dimension N, their definitions can be given, respectively, as follows:

$$\mathrm{PSNR} = 10 \log \frac{255^2}{\frac{1}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N}(\overline{P_{ij}} - P_{ij})^2} \text{ (dB)}, \tag{2.13}$$

$$\mathrm{NMSE} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{N}(\overline{P_{ij}} - P_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{N}P_{ij}^2}, \tag{2.14}$$

where $\overline{P_{ij}}$ is the intensity value of pixels in the reconstructed images; and P_{ij} the intensity value of pixels in the original images which are split up into n input vectors: $x_i = \{P_{i1}, P_{i2}, \ldots, P_{iN}\}$.

In order to bridge the differences between the two measurements, let us introduce an SNR (signal-to-noise ratio) measurement to interpret the NMSE values as an alternative indication of the quality of those reconstructed images in comparison with the PSNR figures. The SNR is defined below:

$$\mathrm{SNR} = -10 \log \mathrm{NMSE} \text{ (dB)}. \tag{2.15}$$

Considering the different settings for the experiments reported in various sources [9,14,47,48], it is difficult to make a comparison among all the algorithms presented in this section. To make the best use of all the experimental results available, we take *Lena* as the standard image sample and summarise the related experiments for all the algorithms as illustrated in Tables 1–3 which are grouped into basic back propagation, hierarchical back propagation and adaptive back propagation.

Experiments were also carried out to test the neural network schemes against the conventional image compression techniques such as DCT with SQ (scale quantization) and VQ (vector quantization), sub-band coding with VQ schemes [9]. The results reported reveal that the neural networks provide very competitive compression performance and even outperform the conventional techniques in some occasions [9].

Table 1
Basic back propagation on *Lena* (256 × 256)

Dimension N	Training schemes	PSNR (dB)	Bit rate (bits/pixel)
64	Non-linear	25.06	0.75
64	Linear	26.17	0.75

Table 2
Hierarchical back propagation on *Lena* (512 × 512)

Dimension N	Training schemes	SNR (dB)	Bit rate (bits/pixel)
4	Linear	14.395	1
64	Linear	19.755	1
144	Linear	25.67	1

Table 3
Adaptive back propagation on *Lena* (256 × 256)

Dimension N	Training schemes	PSNR (dB)	Bit rate (bits/pixel)
64	Activity based linear	27.73	0.68
64	Activity and direction based linear	27.93	0.66

Back propagation based neural networks have provided good alternatives for image compression in the framework of the K–L transform. Although most of the networks developed so far use linear training schemes and experiments support this option, it is not clear why non-linear training leads to inferior performance and how non-linear transfer function can be further exploited to achieve further improvement. Theoretical research is required in this area to provide in-depth information and to prepare for further exploration in line with non-linear signal processing using neural networks [12,19,53,61,63,74].

Generally speaking, the coding scheme used by neural networks tends to maintain the same

neurone weights, once being trained, throughout the whole process of image compression. This is because the bit rate will become KT/N instead of that given by Eq. (2.3) in which maximisation of their compression performance can be implemented. From experiments [4,9], in addition, reconstructed image quality does not show any significant difference for those images outside the training set due to the so-called generalization property [9] in neural networks. Circumstances where a drastic change of the input image statistics occurs might not be well covered by the training stage, however, adaptive training might prove worthwhile. Therefore, how to use the minimum number of bits to represent the adaptive training remains an interesting area to be further explored.

2.2. Hebbian learning-based image compression

While the back-propagation-based narrow-channel neural network aims at achieving a compression upper bounded by the K–L transform, a number of Hebbian learning rules have been developed to address the issue of how the principal components can be directly extracted from input image blocks to achieve image data compression [17,36,58,71]. The general neural network structure consists of one input layer and one output layer which is actually a half of the network shown in Fig. 1. Hebbian learning rule comes from Hebb's postulation that if two neurones were very active at the same time which is illustrated by the high values of both its output and one of its inputs, the strength of the connection between the two neurones will grow or increase. Hence, for the output values expressed as $[h] = [w]^{\mathrm{T}}[x]$, the learning rule can be described as

$$W_i(t+1) = \frac{W(t) + \alpha h_i(t)X(t)}{\|W_i(t) + \alpha h_i(t)X(t)\|}, \tag{2.16}$$

where $W_i(t+1) = \{w_{i1}, w_{i2}, \ldots, w_{iN}\}$ is the ith new coupling weight vector in the next cycle $(t+1)$; $1 \leqslant i \leqslant M$ and M is the number of output neurones. α the learning rate; $h_i(t)$ the ith output value; $X(t)$ the input vector, corresponding to each individual image block and $\|\cdot\|$ the Euclidean norm used to normalize the updated weights and make the learning stable.

From the above basic Hebbian learning, a so-called linearized Hebbian learning rule is developed by Oja [50,51] by expanding Eq. (2.16) into a series from which the updating of all coupling weights is constructed from below:

$$W_i(t+1) = W_i(t) + \alpha[h_i(t)X(t) - h_i^2(t)W_i(t)]. \tag{2.17}$$

To obtain the leading M principal components, Sanger [58] extends the above model to a learning rule which removes the previous principal components through Gram–Schmidt orthogonalization and made the coupling weights converge to the desired principal components. As each neurone at the output layer functions as a simple summation of all its inputs, the neural network is linear and forward connected. Experiments are reported [58] that at a bit rate of 0.36 bits/pixel, an SNR of 16.4 dB is achieved when the network is trained by images of 512×512 and the image compressed was outside the training set.

Other learning rules developed include the adaptive principal component extraction (APEX) [36] and robust algorithms for the extraction estimation of the principal components [62,72]. All the variations are designed to improve the neural network performance in converging to the principal components and in tracking down the high order statistics from the input data stream.

Similar to those back-propagation trained narrow channel neural networks, the compression performance of such neural networks are also dependent on the number of output neurones, i.e. the value of M. As explained in the above, the quality of the reconstructed images are determined by $e_{\mathrm{ms}} = E\{(x - \bar{x})^2\} = \sum_{j=M+1}^{N} \lambda_j$ for neural networks with N-dimensional input vectors and M output neurones. With the same characteristics as that of conventional image compression algorithms, high compression performance is always balanced by the quality of reconstructed images. Since conventional algorithms in computing the K–L transform involves lengthy operations in angle tuned trial and verification iterations for each vector and there is no fast algorithm in existence so far, neural network development provides advantages in terms of computing efficiency and speed in

dealing with these matters. In addition, the iterative nature of these learning processes is also helpful in capturing those slowly varying statistics embedded inside the input stream. Hence the whole system can be made adaptive in compressing various image contents such as those described in the last section.

2.3. Vector quantization neural networks

Since neural networks are capable of learning from input information and optimizing itself to obtain the appropriate environment for a wide range of tasks [38], a family of learning algorithms has been developed for vector quantization. For all the learning algorithms, the basic structure is similar which can be illustrated in Fig. 5. The input vector is constructed from a K-dimensional space. M neurones are designed in Fig. 5 to compute the vector quantization code-book in which each neurone relates to one code-word via its coupling weights. The coupling weight, $\{w_{ij}\}$, associated with the ith neurone is eventually trained to represent the code-word c_i in the code-book. As the neural network is being trained, all the coupling weights will be optimized to represent the best possible partition of all the input vectors. To train the network, a group of image samples known to both encoder and decoder is often designated as the training set, and the first M input vectors of the training data set are normally used to initialize all the neurones. With this general structure, various learning algorithms have been designed and developed such as Kohonen's self-organizing feature mapping [10,13,18,33,52,70], competitive learning [1,54,55,65], frequency sensitive competitive learning [1,10], fuzzy competitive learning [11,31,32], general learning [25,49], and distortion equalized fuzzy competitive learning [7] and PVQ (predictive VQ) neural networks [46].

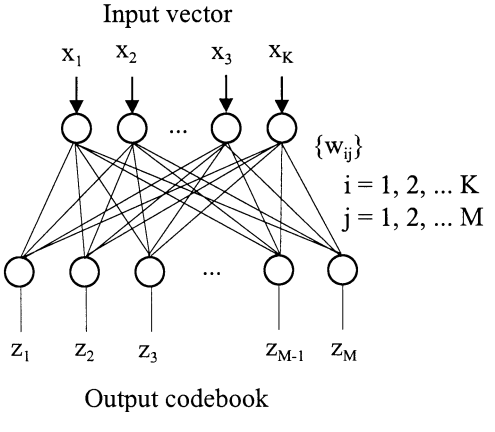

Fig. 5. Vector quantization neural network.

Let $\boldsymbol{W}_i(\boldsymbol{t})$ be the weight vector of the ith neurone at the tth iteration, the basic competitive learning algorithm can be summarized as follows:

$$z_i = \begin{cases} 1 & d(x, W_i(t)) = \min\limits_{1 \leqslant j \leqslant M} d(x, W_j(t)), \\ 0 & \text{otherwise}, \end{cases} \tag{2.18}$$

$$W_i(t+1) = W_i(t) + \alpha(x - W_i(t))z_i, \tag{2.19}$$

where $d(x, W_i(t))$ is the distance in the L_2 metric between the input vector x and the coupling weight vector $W_i(t) = \{w_{i1}, w_{i2}, \ldots, w_{iK}\}$; $K = p \times p$; α is the learning rate, and z_i is its output.

A so-called under-utilization problem [1,11] occurs in competitive learning which means some of the neurones are left out of the learning process and never win the competition. Various schemes are developed to tackle this problem. Kohonen self-organizing neural network [10,13,18] overcomes the problem by updating the winning neurone as well as those neurones in its neighbourhood.

Frequency sensitive competitive learning algorithm addresses the problem by keeping a record of how frequent each neurone is the winner to maintain that all neurones in the network are updated an approximately equal number of times. To implement this scheme, the distance is modified to include the total number of times that the neurone i is the winner. The modified distance measurement is defined as

$$d(x, W(t)_i) = d(x, W_i(t))u_i(t), \tag{2.20}$$

where $u_i(t)$ is the total number of winning times for neurone i up to the tth training cycle. Hence, the more the ith neurone wins the competition, the greater its distance from the next input vector. Thus, the chance of winning the competition diminishes. This way of tackling the under-utilization

problem does not provide interactive solutions in optimizing the code-book.

For all competitive learning based VQ (vector quantization) neural networks, fast algorithms can be developed to speed up the training process by optimizing the search for the winner and reducing the relevant computing costs [29].

By considering general optimizing methods [19], numerous variations of learning algorithms can be designed for the same neural network structure shown in Fig. 5. The general learning VQ [25,49] is one typical example. To model the optimization of centroids $w \in R^M$ for a fixed set of training vectors $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$, a total average mismatch between $\boldsymbol{x}$, the input vector, and $\boldsymbol{W}$, the code-book represented by neurone weights, is defined as

$$\Gamma(W) = \sum_{i=1}^{K} \sum_{j=1}^{M} \frac{g_{ir}\|x_i - W_r\|^2}{K} = \sum_{i=1}^{K} \sum_{j=1}^{M} L_x, \quad (2.21)$$

where g_{ir} is given by

$$g_{ir} = \begin{cases} 1 & \text{if} \quad r = i, \\ \dfrac{1}{\sum_{j=1}^{M}\|x - W_j\|^2} & \text{otherwise}, \end{cases}$$

and W_i is assumed to be the weight vector of the winning neurone.

To derive the optimum partition of $\boldsymbol{x}$, the problem is reduced down to minimizing the total average mismatch $\Gamma(W)$, where $W = \{W_1, W_2, \ldots, W_M\}$ is the code-book which varies with respect to the partition of $\boldsymbol{x}$. From the basic learning algorithm theory, the gradient of $\Gamma(W)$ can be obtained in an iterative, stepwise fashion by moving the learning algorithm in the direction of the gradient of L_x in Eq. (2.21). Since L_x can be rearranged as

$$\begin{aligned} L_x &= \sum_{r=1}^{M} g_{ir}\|x - W_r\|^2 \\ &= \|x - W_i\|^2 + \frac{\sum_{r=1,\, r\neq i}^{M}\|x - W_r\|^2}{\sum_{j=1}^{M}\|x - W_j\|^2} \\ &= \|x - W_i\|^2 + 1 - \frac{\|x - W_i\|^2}{\sum_{j=1}^{M}\|x - W_j\|^2}, \end{aligned} \quad (2.22)$$

L_x is differentiated with respect to W_i and W_j as follows:

$$\frac{\partial L_x}{\partial W_i} = -2(x - W_i)\frac{D^2 - D + \|x - W_i\|^2}{D^2}, \quad (2.23)$$

$$\frac{\partial L_x}{\partial W_j} = 2(x - W_j)\frac{\|x - W_i\|^2}{D^2}, \quad (2.24)$$

where $D = \sum_{r=1}^{M}\|x - W_r\|^2$.

Hence, the learning rule can be designed as follows:

$$W_i(t+1) = W_i(t) + \alpha(t)(x - W_i(t)) \times \frac{D^2 - D + \|x - W_i(t)\|^2}{D^2} \quad (2.25)$$

for the winning neurone i, and

$$W_j(t+1) = W_j(t) + \alpha(t)(x - W_j(t))\frac{\|x - W_i(t)\|^2}{D^2} \quad (2.26)$$

for the other $(M-1)$ neurones. This algorithm can also be classified as a variation of Kohonen's self-organizing neural network [33].

Around the competitive learning scheme, fuzzy membership functions are introduced to control the transition from soft to crisp decisions during the code-book design process [25,31]. The essential idea is that one input vector is assigned to a cluster only to a certain extent rather than either 'in' or 'out'. The fuzzy assignment is useful particularly at earlier training stages which guarantees that all input vectors are included in the formation of a new code-book represented by all the neurone coupling weights. Representative examples include direct fuzzy competitive learning [11], fuzzy algorithms for learning vector quantization [31,32] and distortion equalized fuzzy competitive learning algorithm [7], etc. The so-called distortion equalized fuzzy competitive learning algorithm modifies the distance measurement to update the neurone weights by taking fuzzy membership functions into consideration to optimize the learning process. Specifically, each neurone is allocated a distortion represented by $\boldsymbol{V}_j(\boldsymbol{t})$, $j = 1, 2, \ldots, M$, with their initial values $\boldsymbol{V}_j(\boldsymbol{0}) = 1$. The distance between the input vector $\boldsymbol{x}$ and all the neurone weights W_j is

then modified as follows:

$$D_{kj} = d(x_k, W_j(t)) = \frac{V_j(t)}{\sum_{l=1}^{M} V_l(t)} \|x - W_j(t)\|^2. \quad (2.27)$$

At each training cycle, a fuzzy membership function, $\mu_{kj}(t)$, for each neurone is constructed to update both distortion V_j and the neurone weights given below:

$$V_j(t+1) = V_j(t) + (\mu_{kj})^m D_{kj},$$

$$W_j(t+1) = W_j(t) + (\mu_{kj})^m \alpha(x_k - W_j(t),$$

where $m \geqslant 1$ is a fuzzy index which controls the fuzziness of the whole partition, and μ_{kj} is a fuzzy membership function which specifies the extent of the input vector being associated with a coupling weight. Throughout the learning process, the fuzzy idea is incorporated to control the partition of input vectors as well as avoid the under-utilization problem.

Experiments are carried out to vector quantize image samples directly without any transform by a number of typical CLVQ neural networks [7]. The results show that, in terms of PSNR, the fuzzy competitive learning neural network achieves 26.99 dB for *Lena* when the compression ratio is maintained at 0.44 bits/pixel for all the neural networks. In addition, with bit rate being 0.56 bits/pixel, the fuzzy algorithms presented in [32] achieve a PSNR figure around 32.54 dB for the compression of *Lena* which is a significant improvement compared with the conventional LBG algorithm [32].

In this section, we discussed three major developments in neural networks for direct image compression. The first two conform to the conventional route of pixel transforms in which principal components are extracted to reduce the redundancy embedded inside input images. The third type corresponds to the well developed quantization technique in which neural learning algorithms are called in to help producing the best possible codebook. While the basic structure for encoders are very similar which contain two layers of neurones, the working procedure is fundamentally different. Apart from the fact that the pixel transform based narrow channel networks always have less number of hidden neurones than those of input ones, they also require a complete neural network to reconstruct those compressed images. Yet for those VQ neural networks, reconstruction stands the same as their conventional counterparts, which only involves a look-up table operation. These developments, in fact, cover two sides of image compression in the light of conventional route: pixel transform, quantization and entropy coding. Therefore, one natural alternative is to make these two developments work together which lead to a neural network designed to extract the desired principal components and then followed by another hidden layer of neurones to do the vector quantization. In that sense, the choice of leading principal components could be made more flexible. One simple example is to keep all the principal components instead of discarding those smaller ones and then represent them by a code-book which is refined according to their associated eigenvalues.

3. Neural network development of existing technology

In this section, we show that the existing conventional image compression technology can be developed right into various learning algorithms to build up neural networks for image compression. This will be a significant development in the sense that various existing image compression algorithms can actually be implemented by simply one neural network architecture empowered with different learning algorithms. Hence, the powerful parallel computing and learning capability with neural networks can be fully exploited to build up a universal test bed where various compression algorithms can be evaluated and assessed. Three conventional techniques are covered in this section which include wavelet transforms, fractals and predictive coding.

3.1. Wavelet neural networks

Based on wavelet transforms, a number of neural networks are designed for image processing and representation [16,24,41,66–68,75]. This direction of research is mainly to develop existing image

coding techniques into various neural network structures and learning algorithms.

When a signal $s(t)$ is approximated by daughters of a mother wavelet $h(t)$ as follows, for instance, a neural network structure can be established as shown in Fig. 6 [66–68].

$$\overline{s(t)} = \sum_{k=1}^{K} w_k h\left(\frac{t-b_k}{a_k}\right), \tag{3.1}$$

where the w_k, b_k and a_k are weight coefficients, shifts and dilations for each daughter wavelet. To achieve the optimum approximation, the following least-mean-square energy function can be used to develop a learning algorithm for the neural network in Fig. 6.

$$E = \frac{1}{2}\sum_{t=1}^{T} (s(t) - \overline{s(t)})^2. \tag{3.2}$$

The value of E can be minimized by adaptively changing or learning the best possible values of the coefficients, w_k, b_k and a_k. One typical example is to use a conjugate gradient method [19,67] to achieve optimum approximation of the original signal $s(t)$. Forming the column vectors $[\boldsymbol{g}(\boldsymbol{w})]$ and $[\boldsymbol{w}]$ from the gradient analysis and coefficients w_k, the ith iteration for minimising E with respect to $[\boldsymbol{w}]$ proceeds according to the following two steps:

(i) if k is multiple of n then: $[s(w)^I] = -[g(w)^I]$ else:

$$[s(w)^i] = -[g(w)^i] + \frac{[g(w)^i]^{\mathrm{T}}[g(w)^i]}{[g(w)^{i-1}]^{\mathrm{T}}[g(w)^{i-1}]}[s(w)^{i-1}]; \tag{3.3}$$

(ii)

$$[w^{i+1}] = [w^i] + \alpha^i[s(w)^i]. \tag{3.4}$$

Step (i) computes a search direction $[\boldsymbol{s}]$ at iteration i. Step (ii) computes the new weight vector using a variable step-size α. By simply choosing the step-size α as the learning rate, the above two steps can be constructed as a learning algorithm for the wavelet neural network in Fig. 6.

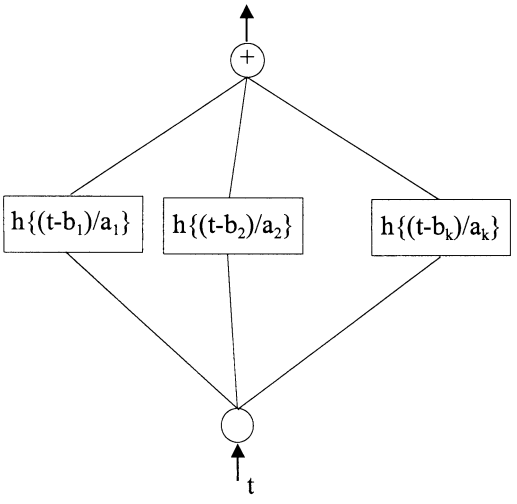

Fig. 6. Wavelet neural network.

With the similar theory (energy function is changed into an error function), [41] suggested a back-propagation and convolution based neural network to search for the optimal wavelet decomposition of input images for efficient image data compression. The neural network structure is designed to have one input layer and one output layer. Local connections of neurones are established to complete the training corresponding to four channels in wavelet decomposition [41,61], i.e. LL (low pass–low pass), LH (low pass–high pass), HL (high pass–low pass) and HH (high pass–high pass). From the well-known back-propagation learning rule, a convolution based training algorithm is used to optimize the neural network training in which quantization of neural network output is used as the desired output to allow back propagation learning. To modify the learning rule, minimization of both quantization error and entropy is taken into consideration by incorporating a so-called entropy reduction function, $Z(QT(i,j))$, hence the overall learning rule can be designed as follows:

$$K_c(u,v)[t+1] = K_c(u,v)[t] + \eta\sum_{i,j}\delta(i,j) \times S(i-u, j-v) + \alpha\Delta K_c(u,v)[t], \tag{3.5}$$

where $K_c(u, v)[t+1]$ is the neural network coupling weights for channel c at $(t+1)$th cycle; η, α are the learning rate and momentum terms, respectively, in modified back propagation to accelerate the learning process; $S(i-u, j-v)$ is the original input image pixel; $\Delta K_c(u, v)[t] = K_c(u, v)[t] - K_c(u, v)[t-1]$; $\delta(i, j) = \dfrac{\partial Ef}{\partial K_c(u, v)}$ the so-called weight-update function derived from the local gradient in the back propagation scheme.

Ef is an error function which is defined below:

$$\mathrm{Ef}(i, j) = Z(\mathrm{QT}(i, j))(T(i, j) - \mathrm{QT}(i, j))^2/2, \tag{3.6}$$

where $T(i, j)$ is the neural network output used to approximate wavelet coefficients;

$\mathrm{QT}(i, j)$ is the quantization of $T(i, j)$;

$$Z(\mathrm{QT}(i, j)) = \begin{cases} 0 & \text{for } \mathrm{QT}(i, j) = 0, \\ 1 & \text{for } |\mathrm{QT}(i, j)| = 1, \\ F(n, q) & \text{for } |\mathrm{QT}(i, j)| = n. \end{cases}$$

$Z(\mathrm{QT}(i, j))$ is the above mentioned entropy reduction function; with $F(n, q)$ being a ramp function, $Z(\mathrm{QT}(i, j))$ can be illustrated in Fig. 7.

While the neural network outputs converge to the optimal wavelet coefficients, its coupling weights also converge to an optimal wavelet low pass filter. To make this happen, extra theoretical work is required to refine the system design.

Firstly, the standard wavelet transform comprises two orthonormal filters, one low pass and the other high pass. To facilitate the neural network learning and its convergence to one optimal wavelet filter, the entire wavelet decomposition needs to be represented by one type of filter rather than two. This can be done by working on the relation between the two orthonormal filters [2,41]. Hence all the three channels, LH, HL and HH, can be represented by channel LL, and the whole wavelet decomposition can be implemented by training the neural network into one optimal low pass filter. This type of redesign produces equivalently a pre-processing of input pixel for LH, HL and HH channels. Basically, the pre-processing involves a flipping operation of the input matrix and an alternating operation of signs for the flipped matrix [39].

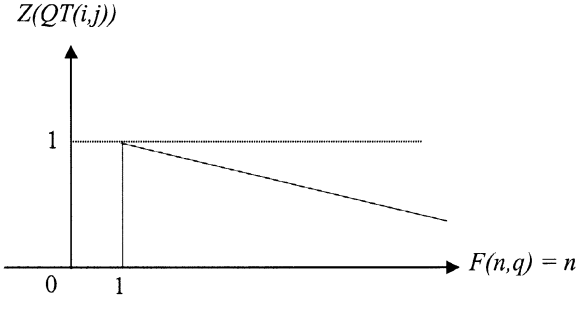

Fig. 7. Entropy reduction function.

Secondly, each epoch of the neural network training is only a suggestion or approximation of the desired optimal wavelet low-pass filter, which also gives us the smallest possible values of the error function Ef. To make it the best possible approximation, a post-processing is added to convert the neural network coupling weights into a required wavelet filter. Assuming that this desired wavelet filter is h'_u, the distance between h'_u and the neural network suggestions can be evaluated as

$$f(h'_u) = \sum_{u,v} (h'_u h'_v - K_c(u, v))^2. \tag{3.7}$$

To minimize the distance subject to constraint equations $C_p(h'_u) = 0$, the Langrangian multiplier method is used to obtain a set of h'_u, which is given below:

$$\mathrm{d}f(h'_u) + \sum_p \lambda_p \, \mathrm{d}C_p(h'_u) = 0, \tag{3.8}$$

where λ_p is the Lagrangian multiplier, $\mathrm{d}f(\cdot)$ the differentiation of function $f(\cdot)$ and $C_p(h'_u)$ the equations corresponding to the following constraints for wavelet low pass filters [36]:

$$\begin{cases} [\sum_u h_{2u}] - \sqrt{2}/2 = 0, \\ [\sum h_u h_{u+2n}] - \delta_{u,u+2n} = 0, \end{cases}$$

δ_{ij} is the Dirac delta function.

Experiments reported [39] on a number of image samples support the proposed neural network system by finding out that Daubechie's wavelet

produces a satisfactory compression with the smallest errors and Harr's wavelet produces the best results on sharp edges and low-noise smooth areas. The advantage of using such a neural network is that the searching process for the best possible wavelets can be incorporated directly into the wavelet decomposition to compress images [41,74]. Yet conventional wavelet based image compression technique has to run an independent evaluation stage to find out the most suitable wavelets for the multi-band decomposition [69]. In addition, the powerful learning capability of neural networks may provide adaptive solutions for problems involved in sub-band coding of those images which may require different wavelets to achieve the best possible compression performance [4,71].

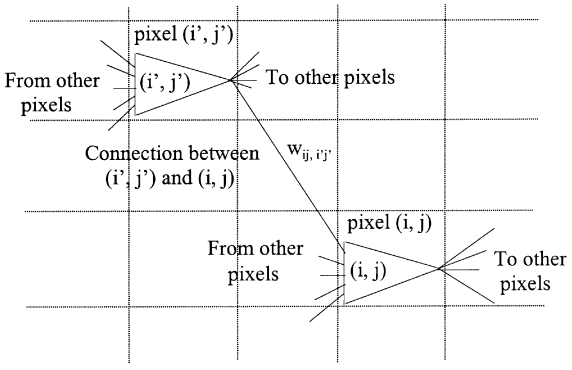

Fig. 8. IFS neural network.

3.2. Fractal neural networks

Fractally configured neural networks [45,64,71] based on IFS (iterated function systems [3]) codes represent another example along the direction of developing existing image compression technology into neural networks. Its conventional counterpart involves representing images by fractals and each fractal is then represented by a so-called IFS which consists of a group of affine transformations. To generate images from IFS, random iteration algorithm is the most typical technique associated with fractal based image decompression [3]. Hence, fractal based image compression features higher speed in decompression and lower speed in compression.

By establishing one neurone per pixel, two traditional algorithms of generating images using IFSs are formulated into neural networks in which all the neurones are organized as a topology with two dimensions [64]. The network structure can be illustrated in Fig. 8 in which $\boldsymbol{w}_{ij,\ i'j'}$ is the coupling weight between (ij)th neurone to $(i'j')$th one, and $\boldsymbol{s}_{ij}$ is the state output of the neurone at position (i, j). The training algorithm is directly obtained from the random iteration algorithm in which the coupling weights are used to interpret the self similarity between pixels [64]. Since not all the neurones in the network are fully connected, the topology can actually be described by a sparse matrix theoretically. In common with most neural networks, the majority of the work operated in the neural network is to compute and optimize the coupling weights, $\boldsymbol{w}_{ij,\ i'j'}$. Once these have been calculated, the required image can typically be generated in a small number of iterations. Hence, the neural network implementation of the IFS based image coding system could lead to massively parallel implementation on a dedicated hardware for generating IFS fractals. Although the essential algorithm stays the same as its conventional algorithm, solutions could be provided by neural networks for the computing intensive problems which are currently under intensive investigation in the conventional fractal based image compression research area.

The neural network proposed in the original reference [64] can only complete image generations from IFS codes. Research in fractal image compression, however, focuses on developing fast encoding algorithms before real-time application can be improved. This involves extensive search for affine transformations to produce fractals for a given image and generating output entropy codes. The work described [64] actually covers the decompression side of the technology. In addition, the neural network requires a large number of neurones arranged in two dimensions which is the same as the number of pixels in any image. Yet large number of the neurones are not active according to the system design. Further work, therefore, is required to be done along this line of research.

3.3. Predictive coding neural networks

Predictive coding has been proved to be a powerful technique in de-correlating input data for speech compression and image compression where a high degree of correlation is embedded among neighbouring data samples. Although general predictive coding is classified into various models such as AR and ARMA, etc., autoregressive model (AR) has been successfully applied to image compression. Hence, predictive coding in terms of applications in image compression can be further classified into linear and non-linear AR models. Conventional technology provides a mature environment and well developed theory for predictive coding which is represented by LPC (linear predictive coding) PCM (pulse code modulation), DPCM (delta PCM) or their modified variations. Non-linear predictive coding, however, is very limited due to the difficulties involved in optimizing the coefficients extraction to obtain the best possible predictive values. Under this circumstance, a neural network provides a very promising approach in optimizing non-linear predictive coding [17,42].

With a linear AR model, predictive coding can be described by the following equation:

$$X_n = \sum_{i=0}^{N} a_i X_{n-i} + v_n = p + v_n, \tag{3.9}$$

where p represents the predictive value for the pixel X_n which is to be encoded in the next step. Its neighbouring pixels, $X_{n-1}, X_{n-2}, \ldots, X_{n-N}$, are used by the linear model to produce the predictive value. v_n stands for the errors between the input pixel and its predictive value. v_n can also be modelled by a set of zero-mean independent and identically distributed random variables.

Based on the above linear AR model, a multilayer perceptron neural network can be constructed to achieve the design of its corresponding non-linear predictor as shown in Fig. 9. For the pixel X_n which is to be predicted, its N neighbouring pixels obtained from its predictive pattern are arranged into a one dimensional input vector $X = \{X_{n-1}, X_{n-2}, \ldots, X_{n-N}\}$ for the neural network. A hidden layer is designed to carry out back propagation learning for training the neural network. The output of each neurone, say the jth neurone, can be derived from the equation given below:

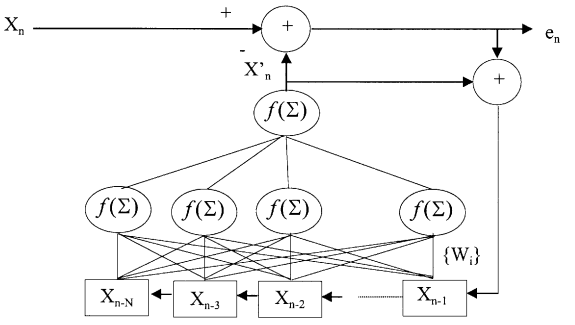

Fig. 9. Predictive neural network I.

$$h_j = f(\Sigma) = f\left(\sum_{i=1}^{N} w_{ji} x_{n-i}\right), \tag{3.10}$$

where $f(v) = 1/(1 + e^{-v})$ is a sigmoid transfer function.

To predict those drastically changing features inside images such as edges, contours, etc., high-order terms are added to improve the predictive performance. This corresponds to a non-linear AR model expressed as follows:

$$X_n = \sum_i a_i X_{n-i} + \sum_i \sum_j a_{ij} X_{n-i} X_{n-j} + \sum_i \sum_j \sum_k a_{ijk} X_{n-i} X_{n-j} X_{n-k} + \cdots + v_n. \tag{3.11}$$

Hence, another so-called functional link type neural network can be designed [42] to implement this type of a non-linear AR model with high-order terms. The structure of the network is illustrated in Fig. 10. It contains only two layers of neurones, one for input and the other for output. Coupling weights, $\{w_i\}$, between the input layer and the output layer are trained towards minimizing the residual energy which is defined as

$$\mathrm{RE} = \sum_n e_n = \sum_n (X_n - \overline{X_n})^2, \tag{3.12}$$

where $\overline{X_n}$ is the predictive value for the pixel X_n. Predictive performance with these neural networks

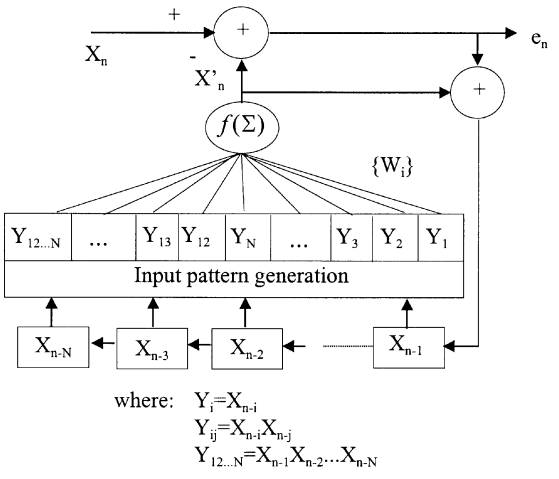

Fig. 10. Predictive neural network II.

is claimed to outperform the conventional optimum linear predictors by about 4.17 and 3.74 dB for two test images [42]. Further research, especially for non-linear networks, is encouraged by the reported results to optimize their learning rules for prediction of those images whose contents are subject to abrupt statistical changes.

4. Neural network based image compression

In this section, we present two image compression systems which are developed firmly based on neural networks. This illustrates how neural networks may play important roles in assisting with new technology development in the image compression area.

4.1. Neural network based adaptive image coding

The structure of a neural network-based adaptive image coding system [21] can be illustrated in Fig. 11.

The system is developed based on a composite source model in which multiple sub-sources are involved. The principle is similar to those adaptive narrow-channel neural networks described in Section 2.1.3. The basic idea is that the input image

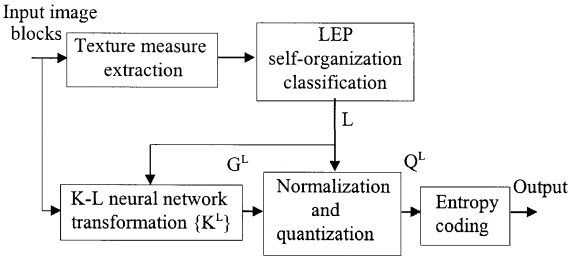

Fig. 11. Neural network adaptive image coding system.

can be classified into a certain number of different classes or sub-sources by features pre-defined for any input image data set. Features of an image are extracted through image texture analysis as shown in the first block in Fig. 11. After the sub-source or class to which the input image block belongs is identified, a K–L neural network transform and a number of quantization code books specifically refined for each individual sub-source are then activated by the class label, L, to further process and achieve the best possible data compression.

The LEP self-organization neural network in Fig. 11 consists of two layers, one input layer and one output layer. The input neurones are organized according to the feature sets. Accordingly, the input neurones can be represented by $\{(x_{mn}), m = 1, 2, \ldots, M_n, n = 1, 2, \ldots, N\}$, where M_n is the dimension of input data with feature set n. It can be seen from this representation that there are N features in the input data each of which consists of M_n elements. In order to classify the input image data into various sub-sources, the neurones at the output layer are represented by $\{(u_{jk}), j = 1, 2, \ldots, p; k = 1, 2, \ldots, q\}$, where p is the number of output sub-sources and q the number of feature sets or perspectives as termed by the original paper [21]. The output neurone representation is also illustrated in Fig. 12 where the neurones in each of the output sub-sources, for the convenience of understanding, are represented by different symbols such as square, triangle, circles and rectangles. It can be seen from Fig. 12 that each individual sub-source of the input data is typified by q features or perspectives at the output layer. Therefore, the overall learning classification system comprises q perspective neural networks which can be summarized in Fig. 13.

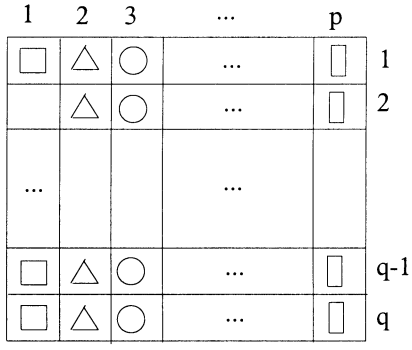

Fig. 12. Output neurons at LEP neural network.

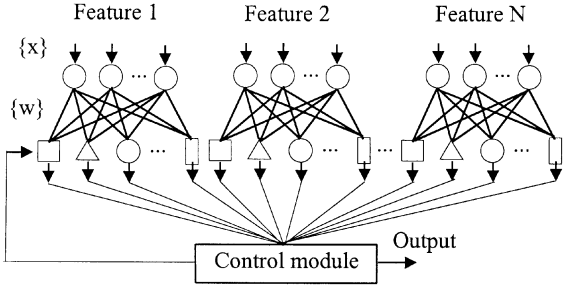

Fig. 13. LEP overall neural network structure.

To classify the input block pattern, each input feature is connected to a perspective array of output neurones which represent different sub-sources or classes. Competitive learning is then run inside each perspective neural network in Fig. 13 to obtain a small proportion of winners in terms of the potential values computed at the output neurones. The potential value is calculated from a so-called activated function. The winners obtained from each neural network, in fact, are referred to as activated. That is because the real winner would be picked up by the overall neural network rather than each individual one, and the winner is a sub-source represented by a column in Fig. 12. Since those activated sub-units or neurones picked up by each individual neural network will not always belong to the same output sub-source, the overall winner will not be selected until all of its sub-units are activated. In Fig. 13, for example, if all triangle neurones are activated in each perspective neural network, the overall winner will then be the sub-source represented by the column of triangles in Fig. 12. In case no match is found, a new sub-source has to be added to the system.

An extended delta learning rule is designed to update the connection weights $\{w_{(mn)(jk)},\ m = 1,2,\ldots,M_n;\ \ n = 1,2,\ldots,N;\ \ j = 1,2,\ldots,p,\ \ k = 1,2,\ldots,q\}$ only for those winning sub-sources. The weight, $w_{(mn)(jk)}$, defines the connection strength between the (mn)th input neurone and the (jk)th output neurone. The modification is

$$\Delta s_{(mn)(jk)} = d(s_{mn}, w_{(mn)(jk)})c_j\alpha(e_i), \quad k = 1,2,\ldots,q, \tag{4.1}$$

where $d(s_{(mn)}, w_{(mn)(jk)})$ is defined as similarity measures [21]; $c_j = \sum \alpha_k c_{jk}$ is the so-called total confidence factor of sub-source j with respect to the input pattern. To judge the reliability of learned information and to decide how far the learning process can modify the existing connection strengths, an experience record, e_j, for each sub-source is defined by the number of times the sub-source has won the competition. In order to be adaptive to the spatially and temporally variant environment, the learning process is also subject to a forgetting process where an experienced attenuation factor is considered.

The K–L transform neural network implemented in Fig. 11 is another example of developing the existing technology into neural learning algorithms [21]. As explained in Section 2.1, the idea of the K–L transform is to de-correlate the input data and produce a set of orthonormal eigenvectors in descending order with respect to variance amplitude. Hence image compression can be achieved by choosing a number of largest principal components out of the whole eigenvector set to reconstruct the image. The neural network structure is the same as that of LEP. For the ith neurone at the output layer, the vector of weight regarding connections to all the input neurones can be represented by $\{w_{ij}, j = 1,2,\ldots,N\}$, where N is the dimension of input vectors. After a certain number of complete learning cycles, all the weight vectors, $\{w_{ij}, j = 1,2,\ldots,N\}$, can be taken as the eigenvectors for the particular K–L transform. Therefore, each weight vector associated with the output neurone represents one

axis of the new co-ordinate system. In other words, the training of the neural network is designed to converge all the axes in the output neurones to the new co-ordinate system described by the orthonormal eigenvectors.

The function within the ith neurone at the output layer is defined as the squared projection of the input pattern on the ith co-ordinate in $N-i+1$ dimensional space:

$$u_i = \left(\sum_{j=1}^{N} w_{ij} s_j \right)^2, \tag{4.2}$$

where w_{ij} is the weight of connection between the jth input neurone and the ith output neurone, and s_j is the jth element of the input vector.

The learning rule is designed on the basis of rotating each co-ordinate axis of the original N-dimensional Cartesian co-ordinate system to the main direction of the processed vectors. Specifically, each time an input vector comes into the network, all the output neurones will be trained one by one, sequentially, for a maximum of $N-1$ times until the angle of rotation is so small that the axis represented by the neurone concerned is almost the same as the main direction of the input patterns. This algorithm is basically developed from the mathematical iterations for obtaining K–L transforms conventionally. As a matter of fact, the back-propagation learning discussed earlier can be adopted to achieve a good approximation of the K–L transform and an improvement over its processing speed.

With this adaptive image coding system, a compression performance of 0.18 bit/pixel is reported for 8 bit grey level images [21].

4.2. Lossless image compression

Lossless image compression covers another important side of image coding where images can be reconstructed exactly the same as originals without losing any information. There are a number of key application areas, notably the compression of X-ray images, in which distortion or error cannot be tolerated.

Theoretically, lossless image compression can be described as an inductive inference problem, in which a conditional probability distribution of future data is learned from the previously encoded data set: $C = \{x_{mn}: n < j \text{ and } 0 \leqslant m < N; n = j \text{ and } 0 \leqslant m < i\}$, where the image encoded is of the size $N \times M$, and x_{ij} is the next pixel to be encoded. To achieve the best possible compression performance, it is desired to make inferences on x_{ij} such that the probability assigned to the entire two dimensional data set,

$$P\{x_{ij}|i,j \in (N,M)\} = \prod_{i,\,j=0}^{N-1,\,M-1} P(x_{ij}|S_{ij}), \quad S_{ij} \subseteq C, \tag{4.3}$$

is maximized, where S_{ij} is the context used to condition the probability assigned.

Due to the fact that images are digitized from analogue signals, strong correlation exists among neighbouring pixels. In practice, therefore, various prediction techniques [22,27,35,43,44,71] are developed to assist with the above inferences. The idea is to de-correlate the pixel data in such a way that the residue of the prediction becomes an independent random data set. Hence, better inferences can be made to maximize the probabilities allocated to all error entries, and the entropy of the error data set can also be minimized correspondingly. As a matter of fact, comparative studies [43,71] reveal that the main redundancy reduction is achieved through the inferences of the maximum probability assigned to each pixel encoded. The context based algorithm [71], for instance, gives the best performance according to the results reported in the literature.

Normally, the criterion for prediction is to minimize the values of errors in which a mean square error function (MSE) is used to measure the prediction performance. This notion leads to a number of linear adaptive prediction schemes which produce a good compression of images on a lossless basis [37,43]. Since prediction also reduces spatial redundancy in image pixels, another criterion of minimizing the zero-order entropy of those errors proves working better than MSE based prediction [30] which often leads to non-linear prediction [44] or complicated linear prediction out of iterations [30]. This is put forward from the observation that optimal MSE based prediction does not

necessarily yield optimal entropy of the predicted errors. Experiments [30] show that around 10% difference exists between the MSE based prediction and the entropy based prediction in terms of the entropy of errors. Under such a context, one possibility of using neural networks for lossless image compression is to train the network such that minimum entropy of its hidden layer outputs can be obtained. The output values equivalent to the predicted errors is then further entropy coded by Huffman or arithmetic coding algorithms.

The work reported in [30] can be taken as an example close to the above scheme. In this work, lossless compression is achieved through both linear prediction and non-linear prediction in which linear prediction is used first to compress the smooth areas of the input image and the non-linear prediction implemented on the neural network is used for those non-smooth areas. Fig. 14 illustrates the linear predictive pattern (Fig. 14(a)) and the multi-layer perceptron neural network structure (Fig. 14(b)). The linear prediction is designed according to the criterion that minimum entropy of the predicted errors is achieved. This is carried out through a number of iterations to refine the coefficients towards their optimal values. Hence, each image will have its own optimal coefficients and these have to be transmitted as the overhead. After the first pass, each pixel will have a corresponding error produced from the optimal linear prediction represented as $g(i, j)$. The pixel is further classified as being inside the non-smooth area if the predicted errors of its neighbouring pixels 1, and 2 given in Fig. 14(a) satisfy the following equation:

$$\mathrm{TH}_1 < |g(i, j-1) + g(i-1, j)| < \mathrm{TH}_2, \tag{4.4}$$

where TH_1 and TH_2 stand for the thresholds.

Around the pixel to be encoded, four predicted errors, $\{g_{i-1,j},\ g_{i-1,j-1},\ g_{i,j-1},\ g_{i+1,j-1}\}$, whose positions are shown in Fig. 15 are taken as the input of the neural network and a supervised training is applied to adjust the weights and thresholds to minimize the MSE between the desired output, $g(i, j)$, and the real output. After the training is finished, the values of weights and thresholds are transmitted as overheads to the decoding end and the neural network is ready to apply a further non-linear prediction to those pixels inside non-smooth areas. The overheads incurred in both optimal linear and neural network non-linear prediction can be viewed as an extra cost for statistics modelling [71]. In fact, the entropy-based criterion is to optimize the predictive coefficients by taking probability assignment or statistical modelling into consideration. No work has been reported, however, regarding how the coefficients can be efficiently and adaptively optimised under such a context without significantly increasing the

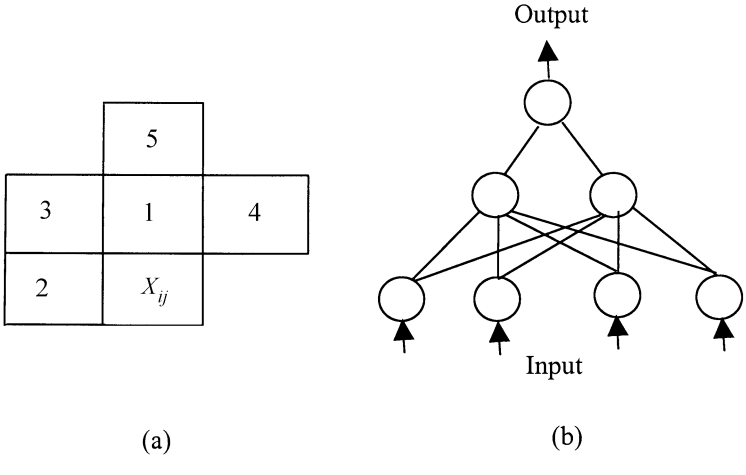

Fig. 14. (a) Predictive pattern for entropy optimal linear prediction; (b) multi-layer perceptron neural network structure.

modelling cost (intensive computing, overheads etc.). This remains an interesting area for further research.

Edge detection based prediction under development by the new JPEG-LS standard [40] produces a simple yet successful prediction scheme in terms of reducing the predicted error entropy. It takes the three neighbouring pixels, the north, the west and the north west location, into consideration to determine the predictive value depending on whether a vertical edge or a horizontal edge is detected or not. Detailed discussion of such a scheme, however, is outside the category of this paper since it is not directly relevant to any neural network application. Further details are referred to the JPEG-LS documents [40].

With MSE based linear prediction, the main problem is that the predictive value for any pixel has to be optimized by its preceding pixel values rather than the pixel itself. Fig. 15 illustrates an example of predictive pattern in which the pixel x_{ij} is to be predicted by its neighbouring four pixels as identified by $x_{i-1,j-1}$, $x_{i,j-1}$, $x_{i+1,\,j-1}$ and $x_{i-1,\,j}$. Thus the predictive value of x_{ij} can be expressed as

$$p_{ij} = a_{ij}^1 x_{i-1,\,j} + a_{ij}^2 x_{i-1,\,j-1} + a_{ij}^3 x_{i,\,j-1} + a_{ij}^4 x_{i+1,\,j-1}. \tag{4.5}$$

To optimize the value of P_{ij}, a group of encoded pixels as shown inside the dashed line area in Fig. 15 which can also be represented by $\cup$ can be selected in place of the pixel x_{ij} to determine the best possible values of a_{ij}^1, a_{ij}^2, a_{ij}^3 and a_{ij}^4. When the MSE criterion is used, the problem comes down to minimizing the following error function:

$$e = \sum_{X_{ij} \in \cup} (x_{ij} - p_{ij})^2. \tag{4.6}$$

The idea is based on the assumption that the coefficients, a_{ij}^1, a_{ij}^2, a_{ij}^3 and a_{ij}^4, will give the best possible predictive value P_{ij} if they can keep the total error in predicting all other pixels inside the window in Fig. 15 to a minimum. The assumption is made on the ground that the image pixels are highly correlated. But when the window goes through those drastically changed image content areas like edges, lines, etc., the assumption would not be correct which causes serious distortion between the predictive value and the current pixel value encoded. One way of resolving this problem is to use a vector quantization neural network to quantize the pixels in $\cup$ into a number of clusters [27]. The idea is to use the neural network to coarsely pre-classify all the pixels inside the window $\cup$ and to exclude those pixels which are unlikely to be close to the pixel to be predicted. In other words, only those pixels which are classified as being inside the same group by the vector quantization neural network are involved in the above MSE based optimal linear prediction. The overall structure of the system can be illustrated in Fig. 16 in which the vector quantization neural network is adopted to produce an adaptive code-book for quantizing or pre-classifying the part of image up to the pixel to be predicted. Since the vector quantization is only

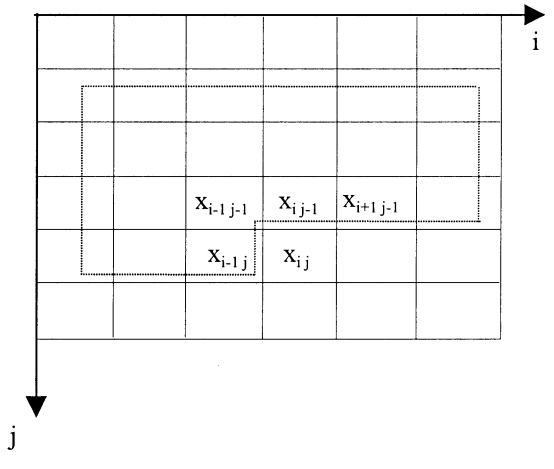

Fig. 15. Predictive pattern.

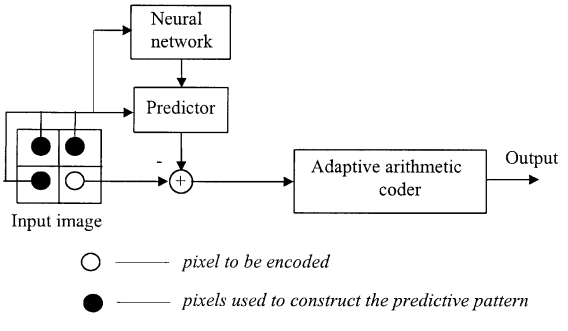

Fig. 16. VQ neural network assisted prediction.

used to pre-classify those previously encoded pixels, no loss of information is incurred during this pre-classification process. To this end, such a neural network application can be viewed as an indirect application for lossless image compression. By considering each pixel as an individual symbol in a pre-defined scanned order, other general lossless data compression neural networks can also be applied for image compression [28,59].

In fact, theoretical results are analysed [56] based on Kolmogorov's mapping neural network existence theorem [20] that a C grey level image of $n \times n$ can be completely described by a three-layer neural network with $2\lceil \log n \rceil$ inputs, $4\lceil \log n \rceil + 2$ hidden neurones and $\lceil \log C \rceil$ output neurones [56]. This leads to a storage of full connections represented by: $8\lceil \log^2 n \rceil + (1 + \lceil \log C \rceil)4 \log n + 2\lceil \log C \rceil$. Compared with the original image which requires $\lceil \log C \rceil n^2$ bits, a theoretical lossless compression ratio can be expected. Further research, therefore, can be initiated under the guidance of this analysis to achieve the theoretical target for various classes of input images.

5. Conclusions

In this paper, we have discussed various up-to-date image compression neural networks which are classified into three different categories according to the nature of their applications and design. These include direct development of neural learning algorithms for image compression, neural network implementation of traditional image compression algorithms, and indirect applications of neural networks to assist with those existing image compression techniques.

With direct learning algorithm development, vector quantization neural networks and narrow-channel neural networks stand out to be the most promising technique which have shown competitive performances and even improvements over those traditional algorithms. While conventional image compression technology is developed along the route of compacting information (transforms within different domains), then quantization and entropy coding, the principal components extraction based narrow-channel neural networks produce better approximation of extracting principal components from the input images, and VQ neural networks make a good replacement for quantization. Since vector quantization is included in many image compression algorithms such as those wavelets based variations, etc., many practical applications can be initiated in image processing and image coding related area.

Theoretically, all the existing state-of-the-art image compression algorithms can be possibly implemented by extended neural network structures, such as wavelets, fractals and predictive coding described in this paper. One of the advantages of doing so is that implementation of various techniques can be standardized on dedicated hardware and architectures. Extensive evaluation and assessment for a wide range of different techniques and algorithms can be conveniently carried out on generalized neural network structures. From this point of view, image compression development can be made essentially as the design of learning algorithms for neural networks. People often get wrong impressions that neural networks are computing intensive and time consuming. The fact is the contrary. For most neural networks discussed in the paper, the main bottleneck is the training or convergence of coupling weights. This stage, however, is only a preparation for the neural network. This will not affect the actual processing speed. In the case of vector quantization, for example, a set of pre-defined image samples is often used to train the neural network. After the training is finished, the converged code-book will be used to vector quantize all the input images throughout the whole process due to the generalization property in neural networks [4,9]. To this end, the neural network will at least perform as equally efficient as that of conventional vector quantizers for software implementation. As a matter of fact, recent work carried out at Loughborough clearly shows that when LBG is implemented on a neural network structure, the software simulation is actually faster than the conventional LBG [4]. With dedicated hardware implementation, the massive parallel computing nature of neural networks is quite obvious due to the parallel structure and arrangement of all the neurones within each layer. In addition, neural networks can also be implemented on general

purpose parallel processing architectures or arrays with programmable capability to change their structures and hence their functionality [18,73].

Indirect neural network applications are developed to assist with traditional techniques and provide a very good potential for further improvement on conventional image coding and compression algorithms. This is typified by significant research work on image pattern recognition, feature extraction and classifications by neural networks. When traditional compression technology is applied to those pre-processed patterns and features, it can be expected to achieve improvement by using neural networks since their applications in these area are well established. Hence, image processing based compression technology could be one of the major research directions in the next stage of image compression development.

At present, research in image compression neural networks is limited to the mode pioneered by conventional technology, namely, *information compacting (transforms) + quantization + entropy coding*. Neural networks are only developed to target individual problems inside this mode [9,15,36,47]. Typical examples are the narrow channel type for information compacting, and LVQ for quantization, etc. Although significant work has been done towards neural network development for image compression, and strong competition can be forced on conventional techniques, it is premature to say that neural network technology standing alone can provide better solutions for practical image coding problems in comparison with those traditional techniques. Co-ordinated efforts world-wide are required to assess the neural networks developed on practical applications in which the training set and image samples should be standardized. In this way, every algorithm proposed can go through the same assessment with the same test data set. Further research can also be targeted to design neural networks capable of both information compacting and quantizing. Hence the advantages of both techniques can be fully exploited. Therefore, future research work in image compression neural networks can be considered by designing more hidden layers to allow the neural networks go through more interactive training and sophisticated learning procedures. Accordingly, high performance compression algorithms may be developed and implemented in those neural networks. Within the infrastructure, dynamic connections of various neurones and non-linear transfer functions can also be considered and explored to improve their learning performances for those image patterns with drastically changed statistics.

References

[1] S.C. Ahalt, A.K. Krishnamurthy et al., Competitive learning algorithms for vector quantization, Neural Networks 3 (1990) 277–290.

[2] A. Averbuch, D. Lazar, M. Israeli, Image compression using wavelet transform and multiresolution decomposition, IEEE Trans. Image Process. 5 (1) (January 1996) 4–15.

[3] M.F. Barnsley, L.P. Hurd, Fractal Image Compression, AK Peters Ltd, 1993, ISBN: 1-56881-000-8.

[4] G. Basil, J. Jiang, Experiments on neural network implementation of LBG vector quantization, Research Report, Department of Computer Studies, Loughborough University, 1997.

[5] Benbenisti et al., New simple three-layer neural network for image compression, Opt. Eng. 36 (1997) 1814–1817.

[6] H. Bourlard, Y. Kamp, Autoassociation by multilayer perceptrons and singular values decomposition, Biol. Cybernet. 59 (1988) 291–294.

[7] D. Butler, J. Jiang, Distortion equalised fuzzy competitive learning for image data vector quantization, in: IEEE Proc. ICASSP'96, Vol. 6, 1996, pp. 3390–3394, ISBN: 0-7803-3193-1.

[8] G. Candotti, S. Carrato et al., Pyramidal multiresolution source coding for progressive sequences, IEEE Trans. Consumer Electronics 40 (4) (November 1994) 789–795.

[9] S. Carrato, Neural networks for image compression, Neural Networks: Adv. and Appl. 2 ed., Gelenbe Pub, North-Holland, Amsterdam, 1992, pp. 177–198.

[10] O.T.C. Chen et al., Image compression using self-organisation networks, IEEE Trans. Circuits Systems For Video Technol. 4 (5) (1994) 480–489.

[11] F.L. Chung, T. Lee, Fuzzy competitive learning, Neural Networks 7 (3) (1994) 539–551.

[12] P. Cicconi et al., New trends in image data compression, Comput. Med. Imaging Graphics 18 (2) (1994) 107–124, ISSN 0895-6111.

[13] S. Conforto et al., High-quality compression of echographic images by neural networks and vector quantization, Med. Biol. Eng. Comput. 33 (5) (1995) 695–698, ISSN 0140-0118.

[14] G.W. Cottrell, P. Munro, D. Zipser, Learning internal representations from grey-scale images: an example of extensional programming, in: Proc. 9th Annual Cognitive Science Society Conf., Seattle, WA, 1987, pp. 461–473.

[15] J. Dangman, Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression, IEEE Trans. ASSP 36 (1988) 1169–1179.
[16] T.K. Denk, V. Parhi, Cherkasky, Combining neural networks and the wavelet transform for image compression, in: IEEE Proc. ASSP, Vol. I, 1993, pp. 637–640.
[17] R.D. Dony, S. Haykin, Neural network approaches to image compression, Proc. IEEE 83 (2) (February 1995) 288–303.
[18] W.C. Fang, B.J. Sheu, T.C. Chen, J. Choi, A VLSI neural processor for image data compression using self-organisation networks, IEEE Trans. Neural Networks 3 (3) (1992) 506–519.
[19] R. Fletcher, Practical Methods of Optimisation, Wiley, ISBN: 0-471-91547-5, New York, 1987.
[20] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Reading, MA, 1990.
[21] N. Heinrich, J.K. Wu, Neural network adaptive image coding, IEEE Trans. Neural Networks 4 (4) (1993) 605–627.
[22] P.G. Howard, J.S. Vitter, New methods for lossless image compression using arithmetic coding, Inform. Process. Management 28 (6) (1992) 765–779.
[23] C.H. Hsieh, J.C. Tsai, Lossless compression of VQ index with search-order coding, IEEE Trans. Image Process. 5 (11) (November 1996) 1579–1582.
[24] K.M. Iftekharuddin et al., Feature-based neural wavelet optical character recognition system, Opt. Eng. 34 (11) (1995) 3193–3199.
[25] J. Jiang, Algorithm design of an image compression neural network, in: Proc. World Congress on Neural Networks, Washington, DC, 17–21 July 1995, pp. I792–I798, ISBN: 0-8058-2125.
[26] J. Jiang, A novel design of arithmetic coding for data compression, IEE Proc.-E: Computer and Digital Techniques 142 (6) (November 1995) 419–424, ISSN 1350-2387.
[27] J. Jiang, A neural network based lossless image compression, in: Proc. Visual'96: Internat. Conf. on Visual Information Systems, 5–6 February 1996, Melbourne, Australia, pp. 192–201, ISBN: 1-875-33852-7.
[28] J. Jiang, Design of neural networks for lossless data compression, Opt. Eng. 35 (7) (July 1996) 1837–1843.
[29] J. Jiang, Fast competitive learning algorithm for image compression neural networks, Electronic Lett. 32 (15) (July 1996) 1380–1381.
[30] W.W. Jiang et al., Lossless compression for medical imaging systems using linear/non-linear prediction and arithmetic coding, in: Proc. ISCAS'93: IEEE Internat. Symp. on Circuits and Systems, Chicago, 3–6 May 1993, ISBN 0-7803-1281-3.
[31] N.B. Karayiannis, P.I. Pai, Fuzzy vector quantization algorithms and their application in image compression, IEEE Trans. Image Process. 4 (9) (1995) 1193–1201.
[32] N.G. Karayiannis, P.I. Pai, Fuzzy algorithms for learning vector quantization, IEEE Trans. Neural Networks 7 (5) (1996) 1196–1211.
[33] T. Kohonen, Self-Organisation and Associative Memory, Springer, Berlin, 1984.
[34] D. Kornreich, Y. Benbenisti, H.B. Mitchell, P. Schaefer, Normalization schemes in a neural network image compression algorithm, Signal Processing: Image Communication 10 (1997) 269–278.
[35] D. Kornreich, Y. Benbenisti, H.B. Mitchell, P. Schaefer, A high performance single-structure image compression neural network, IEEE Trans. Aerospace Electronic Systems 33 (1997) 1060–1063.
[36] S.Y. Kung, K.I. Diamantaras, J.S. Taur, Adaptive principal component extraction (APEX) and applications, IEEE Trans. Signal Process. 42 (May 1994) 1202–1217.
[37] N. Kuroki et al., Lossless image compression by two-dimensional linear prediction with variable coefficients, IEICE Trans. Fund. E75-A (7) (1992) 882–889.
[38] R.P. Lippmann, An introduction to computing with neural nets, IEEE ASSP Mag. (April 1987) 4–21.
[39] S.B. Lo, H. Li et al., On optimization of orthonormal wavelet decomposition: implication of data accuracy, feature preservation, and compression effects, SPIE Proc., Vol. 2707, 1996, pp. 201–214.
[40] Lossless and near-lossless coding of continuous tone still images (JPEG-LS), ISO/IEC JTC 1/SC 29/WG 1 FCD 14495-public draft, 1997 (http://www.jpeg.org/public/jpeg-links.htm).
[41] S.G. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (July 1989) 674–693.
[42] C.N. Manikopoulos, Neural network approach to DPCM system design for image coding, IEE Proc.-I 139 (5) (October 1992) 501–507.
[43] N.D. Memon, K. Sayood, Lossless image compression: A comparative study, in: Proc. SPIE Still Image Compression, Vol. 2418, 1995, pp. 8–27.
[44] N. Memon et al., Differential lossless encoding of images using non-linear predictive techniques, in: Proc. ICIP-94, Internat. Conf. on Image Processing, Vol. III, 13–16 November, Austin, Texas, 1994, pp. 841–845, ISBN: 0-8186-6950-0.
[45] W.L. Merrill John, R.F. Port, Fractally configured neural networks, Neural Networks 4 (1) (1991) 53–60.
[46] N. Mohsenian, S.A. Rizvi, N.M. Nasrabadi, Predictive vector quantization using a neural network approach, Opt. Eng. 32 (7) (July 1993) 1503–1513.
[47] M. Mougeot, R. Azencott, B. Angeniol, Image compression with back propagation: improvement of the visual restoration using different cost functions, Neural Networks 4 (4) (1991) 467–476.
[48] A. Namphol, S. Chin, M. Arozullah, Image compression with a hierarchical neural network IEEE Trans. Aerospace Electronic Systems 32 (1) (January 1996) 326–337.
[49] R.P. Nikhil, C.J. Bezdek, E.C.K. Tsao, Generalized clustering networks and Kohonen's Self-organizing scheme, IEEE Trans. Neural Networks 4 (4) (1993) 549–557.
[50] E. Oja, A simplified neuron model as a principal component analyser, J. Math. Biol. 15 (1982) 267–273.

[51] E. Oja, Data compression, feature extraction, and autoassociation in feedforward neural networks, Artificial Neural Networks, Elsevier, Amsterdam, 1991.
[52] I. Pitas et al., Robust and adaptive training algorithms in self-organising neural networks, Electronic Imaging SPIE 5 (2) (1997).
[53] X.N. Ran, N. Farvardin, A perceptually motivated 3-component image model 2. applications to image compression, IEEE Trans. Image Process. 4 (4) (1995) 430–447.
[54] S.A. Rizvi, N.M. Nasrabadi, Residual vector quantization using a multilayer competitive neural network, IEEE J. Selected Areas Commun. 12 (9) (December 1994) 1452–1459.
[55] S.A. Rizvi, N.M. Nasrabadi, Finite-state residual vector quantization using a tree structured competitive neural network, IEEE Trans. Circuits Systems Video Technol. 7 (2) (1997) 377–390.
[56] S.G. Romaniuk, Theoretical results for applying neural networks – To lossless image compression, Network-Comput. Neural Systems 5 (4) (1994) 583–597, ISSN 0954-898X.
[57] L.E. Russo, E.C. Real, Image compression using an outer product neural network, in: Proc. ICASSP, Vol. 2, San Francisco, CA, 1992, pp. 377–380.
[58] T.D. Sanger, Optimal unsupervised learning in a single-layer linear feed forward neural network, Neural Networks 2 (1989) 459–473.
[59] J. Schmidhuber, S. Heil, Sequential neural text compression, IEEE Trans. Neural Networks 7 (1) (January 1996) 142–146.
[60] J. Shanbehzadeh, P.O. Ogunbona, Index-compressed vector quantization based on index mapping, IEE Proc.-Vision Image Signal Process. 144 (1) (February 1997) 31–38.
[61] J.M. Shapiro, Embedded image-coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 41 (12) (1993) 3445–3462.
[62] W. Skarbek, A. Cichocki, Robust image association by recurrent neural subnetworks, Neural Process. Lett. 3 (3) (1996) 131–138, ISSN 1370-4621.
[63] H.H. Song, S.W. Lee, LVQ combined with simulated annealing for optimal design of large-set reference models, Neural Networks 9 (2) (1996) 329–336.
[64] J. Stark, Iterated function systems as neural networks, Neural Networks 4 (5) (1992) 679–690.
[65] A. Stoffels et al., On object-oriented video coding using the CNN universal machine, IEEE Trans. Circuits Systems I-Fund. Theory Appl. 43 (11) (1996) 948–952.
[66] H. Szu, B. Telfer, J. Garcia, Wavelet transforms and neural networks for compression and recognition, Neural Networks 9 (4) (1996) 695–798.
[67] H. Szu, B. Telfer, S. Kadambe, Neural network adaptive wavelets for signal representation and classification, Opt. Eng. 31 (September 1992) 1907–1916.
[68] C.P. Veronin, Priddy et al., Optical image segmentation using neural based wavelet filtering techniques, Opt. Eng. 31 (February 1992) 287–294.
[69] J.D. Villasenor, B. Belzer, J. Liao, Wavelet filter evaluation for image compression, IEEE Trans. Image Process. 4 (8) (August 1995) 1053–1060.
[70] N.P. Walker et al., Image compression using neural networks, GEC J. Res. 11 (2) (1994) 66–75, ISSN 0264-9187.
[71] M.J. Weinberger, J.J. Rissanen, R.B. Arps, Applications of universal context modelling to lossless compression of grey-scale images, IEEE Trans. Image Process. 5 (4) (1996) 575–586.
[72] L. Xu, A. Yuille, Robust principal component analysis by self-organising rules based on statistical physics approach, Tech. Report, 92-93, Harvard Robotics Lab, February 1992.
[73] R.B. Yates et al., An array processor for general-purpose digital image compression, IEEE J. Solid-state Circuits 30 (3) (1995) 244–250.
[74] Q. Zhang, A. Benveniste, Approximation by non-linear wavelet networks, in: Proc. IEEE Internat. Conf. ASSP, Vol. 5, May 1991, pp. 3417–3420.
[75] L. Zhang et al., Generating and coding of fractal graphs by neural network and mathematical morphology methods, IEEE Trans. Neural Networks 7 (2) (1996) 400–407.