Neural Network Approaches to Image Compression

ROBERT D. DONY, STUDENT MEMBER, IEEE, AND SIMON HAYKIN, FELLOW, IEEE

This paper presents a tutorial overview of neural networks as signal processing tools for image compression. They are well suited to the problem of image compression due to their massively parallel and distributed architecture. Their characteristics are analogous to some of the features of our own visual system, which allow us to process visual information with much ease. For example, multilayer perceptions can be used as nonlinear predictors in differential pulse-code modulation (DPCM). Such predictors have been shown to increase the predictive gain relative to a linear predictor. Another active area of research is in the application of Hebbian learning to the extraction of principal components, which are the basis vectors for the optimal linear Karhunen–Loève transform (KLT). These learning algorithms are iterative, have some computational advantages over standard eigendecomposition techniques, and can be made to adapt to changes in the input signal. Yet another model, the self-organizing feature map (SOFM), has been used with a great deal of success in the design of codebooks for vector quantization (VQ). The resulting codebooks are less sensitive to initial conditions than the standard LBG algorithm, and the topological ordering of the entries can be exploited to further increasing coding efficiency and reduce computational complexity.

I. INTRODUCTION

"A picture is worth a thousand words." This axiom expresses the essential difference between our ability to perceive linguistic information and visual information. For the same message, a visual representation tends to be perceived as being more efficient than the spoken or written word. This is hardly surprising when viewed from an evolutionary perspective. Language is but an instantaneous development in the course of evolutionary history and is manifested in only a single species. In contrast, vision, in one form or another, has existed for hundreds of millions of years and is shared by a countless number of organisms. But why should there be such a difference between the two representations in such an advanced being as ourselves?

The processing of language is inherently serial. Words and their meanings are recorded or perceived one at a time

in a causal manner. Visual information, on the other hand, is processed by massively parallel interconnected networks of processing units. In the mammalian visual system, this parallelism is evident from the retina right through to the higher-order structures in the visual cortex and beyond. The efficiency of such parallel architectures over serial processing is reflected by the efficiency with which we process images over language.

For artificial information processing systems, there are also the two approaches. The most common of these is the serial computer. Its serial nature has made it well suited for the implementation of mathematical formulas and algorithms which tend to be expressed in a linguistic form. Recently, there has been an increased interest in the investigation of parallel information processing systems [1]–[5]. These so-called "artificial neural networks," or simply "neural networks" are networks of simple computational units operating in a parallel and highly interconnected fashion.

The analogy between artificial and natural parallel processing systems is not without some validity. In many instances, the characteristics of natural networks have been the inspiration for artificial networks. As well, the performance of some natural systems have been successfully modelled by artificial systems. There are many tasks which the brain performs very well, but which have eluded efficient implementation in a serial architecture. In would seem logical that to emulate such tasks at which the brain is very efficient in an artificial system, the architectural characteristics of the artificial system should reflect the corresponding natural system. For image processing applications, this suggests that the most efficient computational model would be a parallel, highly interconnected neural network.

Recently there has been a tremendous growth of interest in the field of neural networks. Yet the wide range of applications and architectures which fall under this category make a specific definition of the term "neural network" difficult. A neural network can be defined as a "massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use" [5]. Generally, neural networks refer

to a computational paradigm in which a large number of simple computational units, interconnected to form a network, perform complex computational tasks. There is an analogy between such a computational model and the functioning of complex neurobiological systems. Higher-order neurobiological systems, of which the human brain is the ultimate example, perform extremely complex tasks using a highly connected network of neurons in which each neuron performs a relatively simple information processing task.

This model contrasts sharply with the classical serial computational model found in most general-purpose computers in use today. In the serial model, a highly complex processor performs computations in a rigidly serial manner. The way in which the two models are programmed is also fundamentally different. In the classical model, explicit program steps or states are provided to the processor which must account for all possible input states. In contrast, neural networks are trained using examples of data which the networks will encounter. During training, the network forms an internal representation of the state space so that novel data presented later will be satisfactorily processed by the network.

In many applications, the neural network model may have a number of advantages over the serial model. Because of its parallel architecture, neural networks may break down some of the computational bottlenecks which limit the performance of serial machines. Since neural networks are trained using example data, they can be made to adapt to changes in the input data by allowing the training to continue during the processing of new data. Another advantage of training is that since data are presented individually, no overhead is required to store the entire training set. This is particularly important when processing very large data sets, of which images are an example. The high degree of connectivity can allow neural networks to self-organize, which is an advantage when the structure of the data is not known beforehand. Finally, since there is an analogy between neural networks and neurobiological systems, existing biological networks could be used as models for designing artificial neural networks; it is hoped that some of performance characteristics of the biological network will be inherited by the artificial network.

The sections of the paper are organized as follows. Section II briefly reviews three of the major approaches to image compression: predictive coding, transform coding, and vector quantization. Section III discusses image compression techniques which use neural networks as nonlinear predictors. Transform coding methods using neural networks are presented in Section IV. These methods include linear principal components analysis (PCA) using Hebbian learning, autoassociative coding, and adaptive coding. Section V discusses the application of the self-organizing feature map (SOFM) and its variations to vector quantization. Some of the issues relating to a comprehensive evaluation of the image compression techniques presented herein are discussed in Section VI. Finally, Section VII
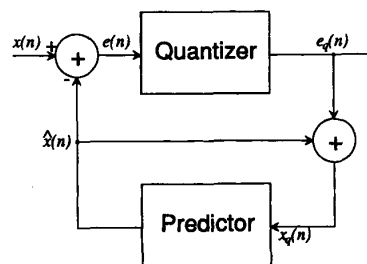

Fig. 1. Block diagram of a DPCM system.

summarizes the important aspects of the methods presented and concludes the paper.

II. IMAGE COMPRESSION

The study of image compression methods has been an active area of research since the inception of digital image processing. Since images can be regarded as two-dimensional signals with the independent variables being the coordinates of a two-dimensional space, many digital compression techniques for one-dimensional signals can be extended to images with relative ease. As a result, a number of approaches to the problem are well established [6]–[14]. Most current approaches fall into one of three major categories: predictive coding, transform coding, or vector quantization. Alternatively, a combination of these techniques may be applied in a hybrid approach.

A. Predictive Coding

Typically, images exhibit a high degree of correlation among neighboring samples. A high degree of correlation implies a high degree of redundancy in the raw data. Therefore, if the redundancy is removed by decorrelating the data, a more efficient and hence compressed coding of the signal is possible. This can be accomplished through the use of predictive coding or differential pulse-code modulation (DPCM).

Fig. 1 shows a block diagram for such a system. The predictor uses past samples $x(n-1), x(n-2), \ldots, x(n-p)$, or in the case of images, neighboring pixels, to calculate an estimate, $\hat{x}(n)$, of the current sample. It is the difference between the true value and the estimate, namely $e(n) = x(n) - \hat{x}(n)$, which is used for storage or transmission. As the accuracy of the predictor increases, the variance of the difference decreases resulting in a higher predictive gain and therefore a higher compression ratio.

The problem, of course, is how to design the predictor. One approach is to use a statistical model of the data to derive a function which relates the value of the neighboring pixels to that of the current one in an optimal manner. An autoregressive model (AR) is one such model which has been successfully applied to images. For a pth order causal AR process, the nth value $x(n)$ is related to the previous

p values in the following manner:

$$x(n) = \sum_{j=1}^{p} w_j x(n-j) + \epsilon_n \qquad (1)$$

where $\{w_j\}$ is a set of AR coefficients, and $\{\epsilon_n\}$ is a set of zero-mean independent and identically distributed (iid) random variables. In this case, the predicted value is a linear sum of the neighboring samples (pixels) as shown by

$$\hat{x}_n = \sum_{j=1}^{p} w_j x(n-j). \qquad (2)$$

Equation (2) is the basis of linear predictive coding (LPC). To minimize the mean squared error $E[(\hat{x}-x)^2]$, the following relationship must be satisfied

$$\mathbf{Rw} = \mathbf{d} \qquad (3)$$

where $[\mathbf{R}]_{ij} = E[x(i)x(j)]$ is ijth element of the autocovariance matrix and, $d_j = E[\hat{x}(n)x(j)]$ is the jth element of the cross covariance vector $\mathbf{d}$. Knowing $\mathbf{R}$ and $\mathbf{d}$, the unknown coefficient vector $\mathbf{w}$ can be computed, and the AR model (i.e., predictor) is thereby determined.

B. Transform Coding

Another approach to image compression is the use of transformations that operate on an image to produce a set of coefficients. A subset of these coefficients is chosen and quantized for transmission across a channel or for storage. The goal of this technique is to choose a transformation for which such a subset of coefficients is adequate to reconstruct an image with a minimum of discernible distortion.

A simple, yet powerful, class of transform coding techniques is linear block transform coding. An image is subdivided into nonoverlapping blocks of $n \times n$ pixels which can be considered as N-dimensional vectors $\mathbf{x}$ with $N = n \times n$. A linear transformation, which can be written as an $M \times N$-dimensional matrix $\mathbf{W}$ with $M \leq N$, is performed on each block with the M rows of W, $\mathbf{w}_i$ being the basis vectors of the transformation. The resulting M-dimensional coefficient vector $\mathbf{y}$ is calculated as

$$\mathbf{y} = \mathbf{Wx}. \qquad (4)$$

If the basis vectors $\mathbf{w}_i$ are orthonormal, that is

$$\mathbf{w}_i^T \mathbf{w}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \qquad (5)$$

then the inverse transformation is given by the transpose of the forward transformation matrix resulting in the reconstructed vector

$$\hat{\mathbf{x}} = \mathbf{W}^T \mathbf{y}. \qquad (6)$$

The optimal linear transformation with respect to minimizing the mean squared error (MSE) is the Karhunen–Loève transformation (KLT). The transformation matrix $\mathbf{W}$ consists of M rows of the eigenvectors corresponding to the M largest eigenvalues of the sample autocovariance matrix

$$\Sigma = E[\mathbf{xx}^T]. \qquad (7)$$

The KLT also produces uncorrelated coefficients and therefore results in the most efficient coding of the data since the redundancy due to the high degree of correlation between neighboring pixels is removed. The KLT is related to principal components analysis (PCA), since the basis vectors are also the M principal components of the data. Because the KLT is an orthonormal transformation, its inverse is simply its transpose.

A number of practical difficulties exist when trying to implement the above approach. The calculation of the estimate of the covariance of an image may be unwieldy and may require a large amount of memory. In addition, the solution of the eigenvectors and eigenvalues is computationally intensive. Finally, the calculation of the forward and inverse transforms is of order $O(MN)$ for each image block. Due to these difficulties, fixed-basis transforms such as the discrete cosine transform (DCT) [15], which can be computed in order $O(N \log N)$, are typically used when implementing block transforms schemes. The Joint Photographics Expert Group (JPEG) have adopted the linear block transform coding approach for its standard using the DCT as the transformation [16].

C. Vector Quantization

The process of quantization maps a signal $x(n)$ into a series of K discrete messages. For the kth message, there exists a pair of thresholds t_k and t_{k+1}, and an output value q_k such that $t_k < q_k \leq t_{k+1}$. For a given set of quantization values, the optimal thresholds are equidistant from the values. The concept of quantizing data can be extended from scalar or one-dimensional data to vector data of arbitrary dimension. Instead of output levels, vector quantization (VQ) employs a set of representation vectors (for the one-dimensional case) or matrices (for the two-dimensional case) [17]–[20], [12]. The set is referred to as the "codebook" and the entries as "codewords." The thresholds are replaced by decision surfaces defined by a distance metric. Typically, Euclidean distance from the codeword is used. The advantage of vector quantization over scalar quantization is that the high degree of correlation between neighboring pixels can be exploited. Even for a memoryless system, the coding of vectors instead of scalars can theoretically improve performance.

In the coding phase, the image is subdivided into blocks, typically of a fixed size of $n \times n$ pixels. For each block, the nearest codebook entry under the distance metric is found and the ordinal number of the entry is transmitted. On reconstruction, the same codebook is used and a simple lookup operation is performed to produce the reconstructed image. The standard approach to calculate the codebook is by way of the Linde, Buzo, and Gray (LBG) algorithm [17]. Initially, K codebook entries are set to random values. On each iteration, each block in the input space is classified,
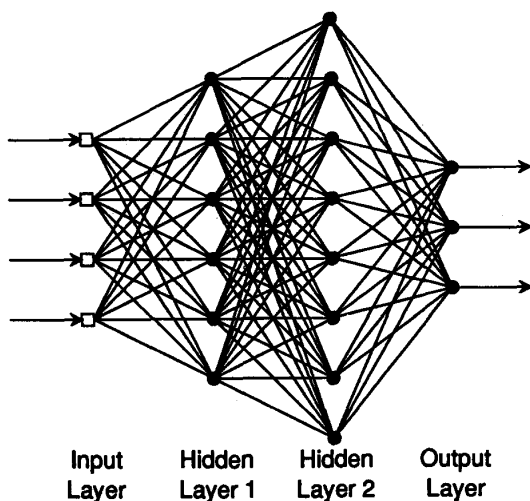
Fig. 2. Multilayer perceptron.

Fig. 3. Model of a neuron.

Fig. 4. Sigmoid nonlinearity.

based on its nearest codeword. Each codeword is then replaced by the mean of its resulting class. The iterations continue until a minimum acceptable error is achieved. This algorithm minimizes the mean squared error over the training set.

While the LBG algorithm converges to a local minimum, it is not guaranteed to reach the global minimum. In addition, the algorithm is very sensitive to the initial codebook. Furthermore, the algorithm is slow since it requires an exhaustive search through the entire codebook on each iteration.

With this brief review of conventional image compression techniques at hand, we are ready to consider the role of neural networks as an image compression tool.

III. PREDICTIVE CODING USING NEURAL NETWORKS

As indicated above, optimal predictors based on a linear weighted sum of the neighboring pixels are relatively easy to design using the statistics of the image. However, if a nonlinear model is more appropriate, the use of a linear predictor will clearly result in a suboptimal solution. Unfortunately, the design of nonlinear predictors is generally not as mathematically tractable as that of linear predictors. Neural networks may provide some useful approaches to the optimal design of nonlinear predictors.

A. Multilayer Perceptrons

When designing a nonlinear predictor, the goal is to find the optimal parameter set $\mathbf{W}_0$ for a given nonlinear function of the previous p inputs, as shown by

$$\hat{x}(n) = f(x(n-1), \ldots, x(n-p), \mathbf{W}) \qquad (8)$$

such that the mean squared value of the prediction error, $E[(\hat{x} - x)^2]$ is minimized. One such class of nonlinear functions is computed using multilayer perceptrons as shown in Fig. 2. The basic computational unit, often referred to as a
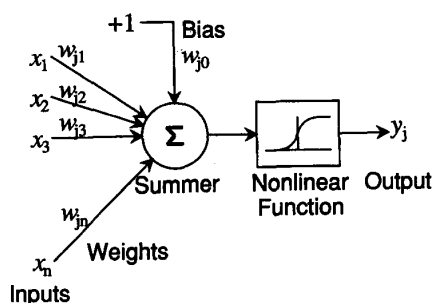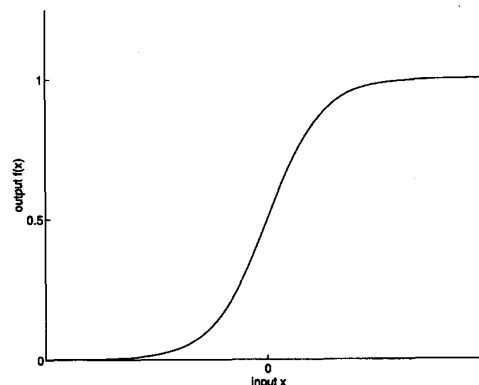
"neuron," consists of a set of "synaptic" weights, one for every input, plus a bias weight, a summer, and a nonlinear function referred to as the activation function as shown in Fig. 3. Each unit computes the weighted sum of the inputs plus the bias weight and passes this sum through the activation function to calculate the output value as

$$y_j = f\left(\sum_i w_{ji}x_i + \theta_j\right) \qquad (9)$$

where x_i is the ith input value for the neuron and w_{ji} is the corresponding synaptic weight. The activation function $f(\bullet)$ maps the potentially infinite range of the weighted sum to a limited, finite range. A common activation function is a sigmoid defined by the logistic function

$$f(v) = \frac{1}{1 + e^{-v}} \qquad (10)$$

as shown in Fig. 4. In a multilayer configuration, the outputs of the units in one layer form the inputs to the next layer. The inputs to the first layer are considered as the network inputs, and outputs of the last layer are the network outputs.

The weights of the network are usually computed by training the network using the backpropagation algorithm. The backpropagation algorithm is a supervised learning algorithm which performs a gradient descent on a squared error energy surface to arrive at a minimum. The key
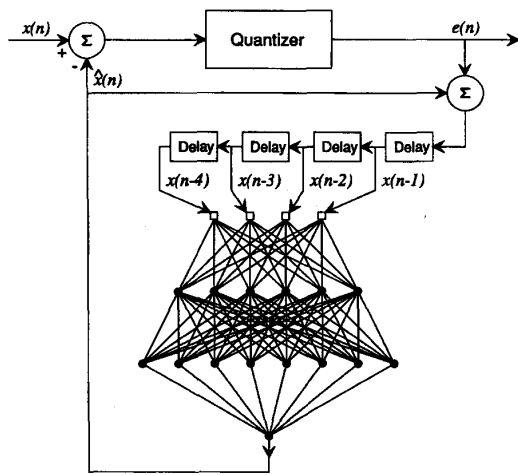
Fig. 5. DPCM using a multilayer perceptron network.

to the use of this method on a multilayer perceptron is the calculation of error values for the hidden units by propagating the error backwards through the network. The local gradient for the jth unit, δ_j, in the output layer is calculated as (assuming a logistic function for the sigmoid nonlinearity)

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \tag{11}$$

where y_j is the output of unit j and d_j is the desired response for the unit. For a hidden layer, the local gradient for neuron j is calculated as

$$\delta_j = y_j(1 - y_j)\sum_k \delta_k w_{jk} \tag{12}$$

where the summation k is taken over all the neurons in the next layer to which the neuron j serves as input. Once the local gradients are calculated, each weight w_{ji} is then modified according to the delta rule

$$w_{ji}(t + 1) = w_{ji}(t) + \eta\delta_j(t)y_i(t) \tag{13}$$

where η is a learning-rate parameter and t is time. Frequently, a modification of (13) is used that incorporates a momentum term that helps accelerate the learning process [1], [5]

$$w_{ji}(t + 1) = w_{ji}(t) + \eta\delta_j(t)y_i(t) + \alpha[w_{ji}(t) - w_{ji}(t - 1)] \tag{14}$$

where α is a momentum term lying in the range $0 < \alpha < 1$.

A multilayer perceptron can be used as a nonlinear predictor. The input consists of the previous p data values and the single output is the predicted value. Between the input and output layers are a number of hidden layers of varying size. Fig. 5 shows such a configuration.

Because of the nonlinear nature of the network, the variance of the prediction error of a neural network can be less than that of a linear predictor, which results in an increased predictive gain for a DPCM system. Dianat *et al.* [21] used a network with a 3-unit input layer, one

hidden layer of 30 units, and one output unit. The input consist of the three immediate causal neighbors. Their experiments showed a 4.1 dB improvement in signal-to-noise ratio (SNR) over the optimal linear predictive coding system using the same input configuration. In addition, they demonstrated an improvement in error entropy from 4.7 bits per pixel (bpp) to 3.9 bpp.

B. Higher-Order Predictors

Another approach is to take advantage of the gradient descent properties of the backpropagation algorithm for the calculation of the optimal predictor for a specific nonlinear model. Manikopoulos [22], [23] has used a nonlinear predictor based on the discrete-time Volterra expansion of a generalized nonlinear predictor model. The justification for using a nonlinear approach is that linear AR image models do not adequately account for sharply defined structures such as edges in images. Therefore, higher-order terms are required for a generalized autoregressive model

$$x(n) = \sum_i w_i x(n - i) + \sum_i \sum_j w_{ij} x(n - i) x(n - j)$$
$$+ \sum_i \sum_j \sum_k w_{ijk} x(n - i) x(n - j) x(n - k)$$
$$+ \cdots + \epsilon_n \tag{15}$$

where $\{\epsilon_n\}$ is a sequence of zero-mean iid random variables.

A single layer neural network was used for the nonlinear predictor. The inputs consist of the previous p samples plus all the higher-order cross terms. Fig. 6 shows the configuration of a 4th order ($p = 4$) system. Improvements in SNR over the optimal linear predictor of 4.17 dB and 3.74 dB were realized for two test images with a four-neighbor 1-D predictor. For a 2-D predictor using nine neighbors, a SNR of 29.5 dB at 0.51 bpp was achieved.

A network similar to that shown in Fig. 6 was proposed in 1989 by Pao [24] who introduced a functional-link network that allows a multilayer perception to be replaced by a single-layer network employing higher-order terms. The learning rate of a single layer higher-order network was shown to be substantially faster than that of an equivalent multilayer perceptron.

C. Cascaded Recurrent Networks

The use of neural network predictors may also be extended to both nonlinear and nonstationary signal models through the use of pipelined recurrent neural networks (PRNN) [5], [25]. The network consists of a number of modules, each of which is a recurrent neural network with N-1 feedback inputs. Fig. 7 shows the architecture of the system. Each module receives as input p delayed version of the input signal and the N-1 feedback inputs. The output of each module serves as input to the next. In this manner, the network effects a *nested nonlinear* function of the inputs. The final network output is a linear weighted sum of the module outputs. The weights of each module $\mathbf{W}_i$ are trained via a gradient descent algorithm and the weights for
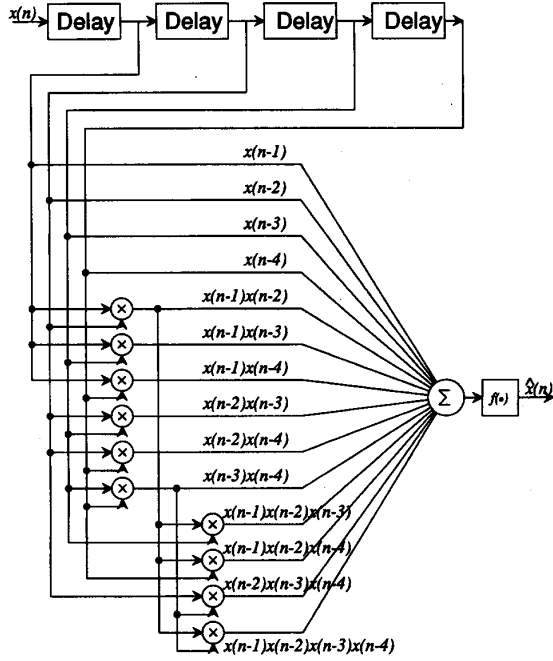
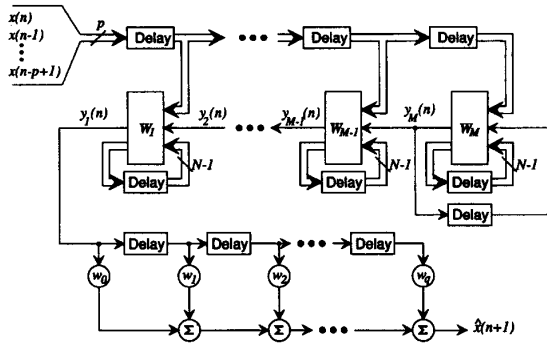Fig. 6. DPCM using a 4th order nonlinear AR model.

Fig. 7. Pipelined recurrent neural network.

the linear sum are computed via the standard least-mean-squares (LMS) algorithm [26]. In experiments using speech signals, Haykin and Li [25] found that the PRNN based predictor can improve the prediction gain by 3 dB over a linear FIR filter. The use of PRNN for image compression is yet to be explored.

IV. TRANSFORM CODING USING NEURAL NETWORKS

A. Linear PCA

One solution to the problems associated with the calculation of the basis vectors through eigendecomposition of the covariance estimate is the use of iterative techniques based on neural network models. These approaches require less storage overhead and can be more computationally efficient.
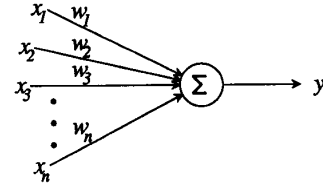

Fig. 8. Simplified linear neuron.

As well, they are able to adapt over long-term variations in the image statistics.

In 1949, Donald Hebb proposed a mechanism whereby the synaptic strengths between connecting neurons can be modified to effect learning in a neurobiological network [27]. Hebb's postulate of learning states that the ability of one neuron to cause the firing of another neuron increases when that neuron consistently takes part in firing the other. In other words, when an input and output neuron tend to fire at the same time, the connection between the two is reinforced.

For artificial neural networks, the neural interactions can be modeled as a simplified linear computational unit as shown in Fig. 8. The output of the neuron, y, is the sum of the inputs $\{x_1, x_2, \ldots, x_N\}$ weighted by the synaptic weights $\{w_1, w_2, \ldots, w_N\}$, or in vector notation,

$$y = \mathbf{w}^T \mathbf{x}. \tag{16}$$

Taking the input and output values to represent "firing rates," the application of Hebb's postulate of learning to this model would mean that a weight w_i would be increased when both values of x_i and y are correlated. Extending this principle to include simultaneous negative values (analogous to inhibitory interactions in biological networks), the weights $\mathbf{w}$ would be modified according to the correlation between input vector $\mathbf{x}$ and the output y.

A simple Hebbian rule updates the weights in proportion to the product of the input and output values as

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha y(t)\mathbf{x}(t) \tag{17}$$

where α is a learning-rate parameter. However, such a rule is unstable since the weights tend to grow without bound. Stability can be imposed by normalizing the weights at each step as

$$\mathbf{w}(t+1) = \frac{\mathbf{w}(t) + \alpha y(t)\mathbf{x}(t)}{\|\mathbf{w}(t) + \alpha y(t)\mathbf{x}(t)\|} \tag{18}$$

where $\| \bullet \|$ denotes the Euclidean norm. This rule has been shown to converge to the largest principal component of the input $\mathbf{x}$ [28]–[31]. Oja linearized (18) using a series expansion to form

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha[y(t)\mathbf{x}(t) - y^2(t)\mathbf{w}]. \tag{19}$$

Equation (19) has also been shown to converge to the largest principal component [5].
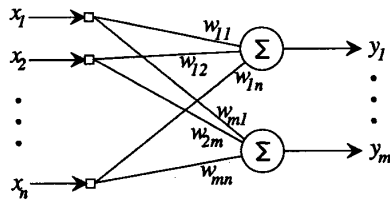
Fig. 9. M principal components linear network.

B. Generalized Hebbian Algorithm

Oja's rule has formed the foundation for extending Hebbian learning to simultaneously find the first M principal components. Fig. 9 shows the architecture of such a system. Each output y_i corresponds to the output of the ith principal component neuron. In vector notation, it can be written as

$$y = Wx \tag{20}$$

with $y \in \mathcal{R}^M$, $W \in \mathcal{R}^{M \times N}$, and $M \leq N$.

Sanger's generalized Hebbian algorithm (GHA) [32]–[34] extends Oja's model to compute the leading M principal components using the fact that the computation of any principal components is identical to that of the first with the data being modified by removing the previous principal components through Gram–Schmidt orthogonalization. In other words, the mth principal component is the first principal components of w_m where

$$w_m = w - W_{m-1}^T W_{m-1} w \tag{21}$$

and W_{m-1} is an $(m - 1) \times N$ matrix whose $m - 1$ rows are the previous $m - 1$ principal components. The orthogonalization is incorpoated into the learning rule to form

$$W(t+1) = W(t) + \alpha(t)(y(t)x^T(t) - LT[y(t)y^T(t)]W(t)) \tag{22}$$

where $LT[\cdot]$ is the lower triangular operator, i.e., it sets all elements above the diagonal to zero. Under the conditions that $\lim_{t \to \infty} \alpha(t) = 0$ and $\sum_{t=0}^{t<\infty} \alpha(t) = \infty$, W converges to a matrix whose rows are the M principal components [32].

For performance evaluation, Sanger implemented the algorithm using 8×8 input blocks and an output dimensionality of 8. The network was trained on a 512×512 image using nonoverlapping blocks with the image being scanned twice. The learning parameter α was fixed in the range [0.1, 0.01]. The coefficients were nonuniformly quantized with the number of bits varying with the sample variance of each coefficient. At a compression of 0.36 bpp, a normalized MSE of 0.043 resulted. When the same matrix W was used to code a second independent image, a compression of 0.55 bpp resulted in a normalized MSE of 0.023. Sanger also applied this algorithm to a texture segmentation problem and has used it to model receptive fields.
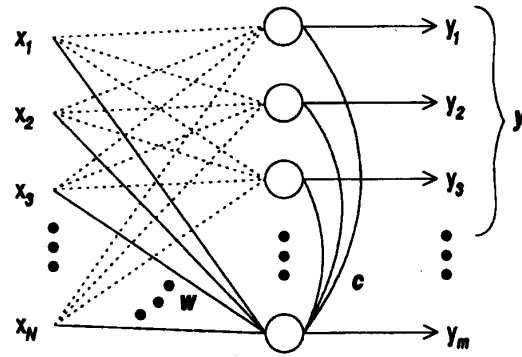

Fig. 10. Network for the APEX algorithm.

C. Adaptive Principal Component Extraction

Sanger's method uses only feedforward connections for calculating the M principal components. An alternative approach proposed by Földiák [35] is to use "anti-Hebbian" feedback connections to decorrelate the components. The justification of this approach was based on earlier work by Barlow and Földiák [36] on the visual cortex. Building on this approach and the work of Oja [28], Kung and Diamantaras [37]–[39] have developed a sequential solution, called adaptive principal components extraction (APEX), in which the output of the mth principal component y_m can be calculated based on the previous $m - 1$ components through

$$y = Wx \tag{23}$$

and

$$y_m = w^T x + c^T y \tag{24}$$

where y is the vector of the first $m - 1$ components, W is the weight matrix for the first $m - 1$ components, w is the weight vector for the mth component and c corresponds to an anti-Hebbian removal of the first $m - 1$ components from the mth component. Fig. 10 shows the architecture of the network.

The learning rule is stated as

$$\Delta w = \alpha(y_m x - y_m^2 w) \tag{25}$$

and

$$\Delta c = -\beta(y_m y - y_m^2 c). \tag{26}$$

Kung and Diamantaras have shown that the weights w converge to the m-th principal component, given that the first $m - 1$ components have already been calculated. As the network converges, the anti-Hebbian weights $c(t)$ converge to zeroes. The optimal learning parameters α and β are calculated as

$$\alpha = \beta = \left(\sum_i^n y_i^2 \right)^{-1} \tag{27}$$

where n is the number of input patterns. This choice of learning-parameters allows the network to adapt to

slowly varying changes in the signal statistics. Further, the additional calculation of a further principal component requires only a linear order, $O(n)$, of multiplications per iteration. For testing, the algorithm was applied to a set of $M=20$ data points of dimension 5. An average squared distance between the principal components extracted from the covariance matrix and those computed using the algorithm was found to be 0.34×10^{-3} for 194 iterations.

Chen and Liu [40] have extended the concept of using feedback connections to extract M principal components simultaneously from the training data, as opposed to the sequential computation of the APEX algorithm. The forward calculation of their network is identical to (24). In addition, the training rule for the orthogonal weights $\mathbf{c}$ is the same as (26). The learning rule for the principal component vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, W_M\}$ is modified to become

$$\Delta \mathbf{w}_i = \alpha\{\mathbf{B}_i[y_m \mathbf{x} - y_m^2 \mathbf{w}] - \mathbf{A}_i \mathbf{w}_i\} \qquad (28)$$

where

$$\mathbf{A}_i = \begin{cases} 0 & , \quad i = 1 \\ \sum_{j=1}^{i-1} \mathbf{w}_i \mathbf{w}_i^T, & i = 2, 3, \ldots, N \end{cases} \qquad (29)$$

and

$$\mathbf{B}_i = \mathbf{I} - \mathbf{A}_i. \qquad (30)$$

The matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ perform the orthogonalization during training. Chen and Liu have shown that the weight vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$ converge to the M principal components while the anti-Hebbian weights $\mathbf{c}_i$ converge to the zero vector.

D. Robust Principal Components Estimation

Xu and Yuille [41] have addressed the problem of robustness in the estimation of the principal components. To account for outliers in the training set they first introduce a binary field which includes data from within the distribution and excludes outliers. As such a function is nondifferentiable, they propose a weighting function based on a Gibbs distribution to account for the degree of deviation from the distribution a data sample may have. By establishing an energy function to be minimized, $J(\mathbf{x}, \mathbf{W})$ the gradient descent learning rule becomes

$$\mathbf{W} = \mathbf{W} - \alpha D_{\beta, \eta}(\mathbf{x}, \mathbf{W}) \nabla J(\mathbf{x}, \mathbf{W}). \qquad (31)$$

where $D_{\beta, \eta}(\mathbf{x}, \mathbf{W})$ is a weighting function defined as

$$D_{\beta, \eta}(\mathbf{x}, \mathbf{W}) = (1 + \exp[\beta(J(\mathbf{x}, \mathbf{W}) - \eta)])^{-1} \qquad (32)$$

which effectively reduces the influence of data points from outside the distribution, i.e., those having a large energy function value $J(\mathbf{x}, \mathbf{W})$. The parameter β is a deterministic annealing parameter which starts as a small value and then increases to infinity. The parameter η determines the region considered as being outside the distribution. As for the choice of the energy function $J(\mathbf{x}, \mathbf{W})$, a number of Hebbian rules can be expressed in terms of the gradient of some energy functions.

E. Discussion of PCA Algorithms

There are a number of advantages which these learning rules have in calculating the M principal components from a data set over standard eigen decomposition techniques. If $M \ll N$, the iterative techniques can be more computationally efficient [42]. As well, because of their iterative nature, they can be allowed to adapt to slowly varying changes in the input stream. A third advantage is that no extra overhead is required to store the data or its higher-order statistics. Finally, if an extra basis were to be required, its computation would be more efficiently performed using the iterative learning rules.

These PCA algorithm using neural network may be categorized into two classes: re-estimation algorithms which use only feedforward connections, and decorrelating algorithms which have both feedforward and feedback connections [43]. The GHA is an example of the former. The learning rule (22) may be restated as

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t) y_j(t)[\mathbf{x}(t) - \hat{x}(t)] \qquad (33)$$

where $\hat{\mathbf{x}}(t)$ is a re-estimator defined by

$$\hat{\mathbf{x}}(t) = \sum_{k=0}^{j} \mathbf{w}_k(t) y_k(t). \qquad (34)$$

The successive outputs of the network are forced to learn different principal components by subtracting estimates of the earlier components from the input before the data are involved in the learning process. In contrast, the APEX algorithm is a decorrelating algorithm. The anti-Hebbian connections decorrelate the successive outputs, resulting in the computation of different principal components.

Recently, there has been some interest in extending the above approaches to a new class of nonlinear PCA networks in which a sigmoidal activation function is added to the model of a neuron. With such a model, it is possible to extract higher-order statistics from the data; however, the resulting basis vectors lose their orthogonality with respect to each other. While nonlinear PCA has been successfully applied to the separation of sinusoidal signals, we feel that due to the loss of orthogonality, their usefulness in image compression may be limited.

F. Autoassociative Transform Coding

Cottrell and Munro [44] and Cottrell, Munro, and Zipser [45] have used an extensional programming approach to image compression. They view the problem of image compression as an encoder problem in which a network is forced to perform an identity mapping through a "narrow" channel. For example, the input and output layers may consist of 8 × 8 units while the hidden layer in the middle has 16 units. In such a scheme, each layer is fully connected. Training consists of presenting a number of randomly selected subimages to the network. Each subimage is presented simultaneously to both the input and output layers and the weights between the layers are
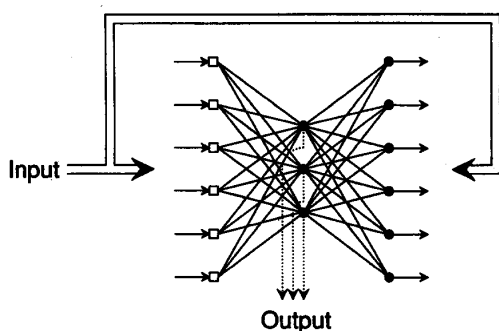
Fig. 11. Autoassociative backpropagation network.

adjusted according to the backpropagation algorithm. The configuration of the network is shown in Fig. 11.

In experiments with real images, Cottrell and Munro achieved compression ratios of 1:8 with acceptable subjective levels of distortion. Linear networks were also shown to produce comparable results to nonlinear networks. In another experiment, each hidden unit was connected to only a 4 × 4 region in the top and bottom layers. This reduction to only 11% of the original connectivity resulted in only a 30% increase in the mean squared error.

The analysis of the resulting network illustrates some very interesting properties. The effect of the network is to produce a set of basis images, one for each hidden unit. It can be shown that these images tend to span the first M principal components of the image, where M is the number of hidden units. When the network is linear, the basis images exactly span the principal components. However, unlike the variances of the principal components coefficients which vary from component to component, the variances of the value of each hidden unit tends to be the same. Furthermore, the error contribution of each appears to be uniformly distributed. In principal component analysis, the error contribution decreases with the size of the eigenvalue.

Cottrell and Metcalfe [46] and Golomb et al. [47] have used the above architecture to successfully compress and restore images of faces. In addition, the values of the hidden units have also been used as input to another network which successfully identified such facial properties as gender and emotional state. Sicuranza et al. [48] also developed a similar network for image compression.

Anthony et al. [49] have used this approach on pulmonary scintigrams containing pulmonary emboli. They compared the total squared error for training and testing on different classes of images. They found that the neural network approach resulted in less squared error compared with conventional PCA when the images used for testing were from the same class of images as those used for training.

Kono et al. [50] developed a modification to this model. They allow a variation in the nonlinearity of limiting function for each neuron. The nonlinearity parameter can be adjusted during training using the backpropagation algorithm simultaneously with the connecting weights. On test

images, the performance of this approach was superior to that of DCT-based compression and that of a neural network without the variable nonlinearity parameter. In the latter case, a 2 dB improvement in SNR was realized.

Another variation proposed by Namphod et al. [51] involves five layers and two sets of training. Initially, the outer two layers are trained with the inner three layers modeled as one layer. Then the inner three are trained while the weights of the outer layers are held constant. Recently, DeMers and Cottrell [52] have proposed a similar structure.

Russo and Real [53] have proposed a learning rule for a linear version of Cottrell's network based on squared error gradient descent. They show that the network converges to a subspace spanned by the M principal components of the training data.

G. Adaptive Transform Coding

As discussed above, the optimal linear block transform for coding images is well known to be the Karhunen–Loève transformation (KLT) which uses the principal components as the bases of the transform. However, the assumption of stationarity in the optimality condition is far from valid for images. The fallacy of this assumption is the reason why the KLT performs poorly at high compression ratios in the vicinity of edges since the image statistics around edges tend to be quite different from the global statistics. Methods such as the KLT, which are globally optimal, are in effect locally suboptimal. Therefore, if processes were made to adapt to local variations in an image, their performance would improve.

An adaptive network which combines Hebbian learning and competitive learning in a topologically ordered map has been proposed, which adapts to mixed data from a number of distributions in a self-organized fashion [54], [55]. Fig. 12 shows the modular architecture for the coding stage of the system. The system consists of a number of independent modules whose outputs are mediated by the subspace classifier. Each module consists of M basis images of dimension $n \times n$ which defines a single linear transformation. The inner product of each basis image with the input image block results in M coefficients per module, represented as an M-dimensional vector $\mathbf{y}_i$. Each module corresponds to one class of input data. The choice of class and therefore the cofficient vector to be transmitted along with its class index is determined by the subspace classifier. The selection is based on the class whose projected vector norm $\|\hat{\mathbf{x}}_i\|$ is maximum. The projected vector $\hat{\mathbf{x}}_i$ is calculated by taking the inverse transformation of the coefficient vector. The image is decoded using the same set of transformations. The class index is used to choose the class for the inverse transformation and the resulting reconstructed image block $\hat{\mathbf{x}}$ is calculated.

It was found that for the same coding rate, the use of the network decreases the MSE by 40–50% over the KLT. Alternatively, for fixed distortion, the compression ratio can almost be doubled [55]. In addition, the network can be used as a segmentor which has the property of illumination invariance, and produces class representations
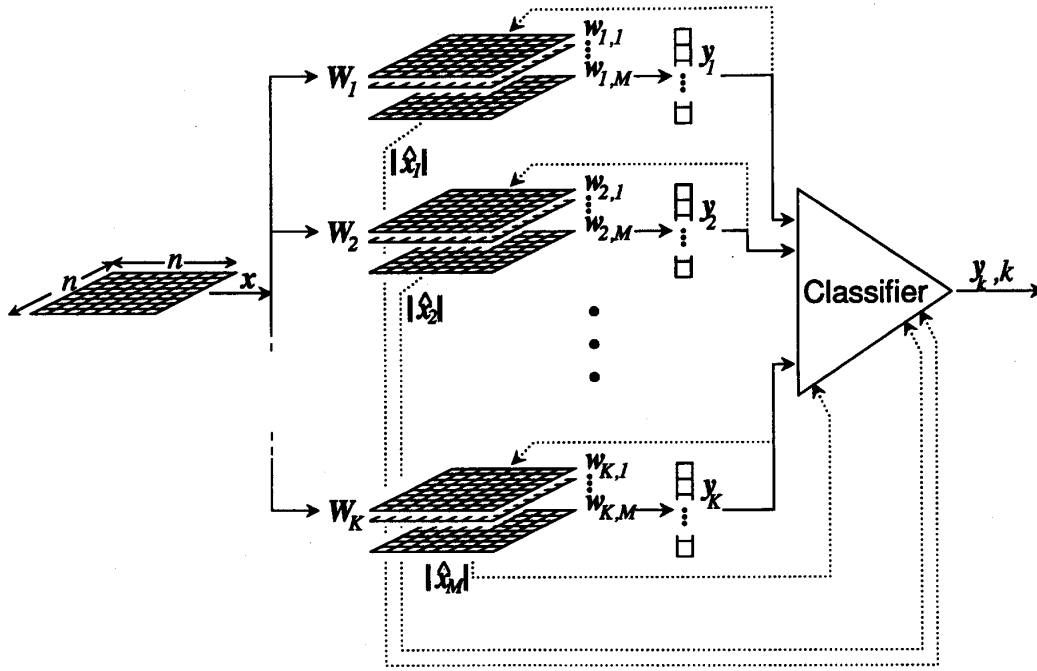
296

Fig. 12. Modular system architecture of OIAL. Input are blocks of $n \times n$ pixels. The K transformations $\mathbf{W}_i$ consist of M basis images of size $n \times n$ and output an M-dimensional vector $\mathbf{y}_i$. The coefficient vector to be sent is chosen by the subspace classifier based on the maximum norm of the projected vector $\|\hat{\mathbf{x}}_i\|$.

that are analogous to the arrangements of the directionally sensitive columns in the visual cortex [55], [56].

V. VECTOR QUANTIZATION USING NEURAL NETWORKS

A. Self-Organizing Feature Map Algorithm

Kohonen's self-organizing feature map (SOFM) [57] has formed a basis for a great deal of research into applying network models to the problem of codebook design in vector quantization. Kohonen introduced the concept of classes ordered in a "topological map" of features. In many clustering algorithms such as k-means each input vector $\mathbf{x}$ is classified and only the "winning" class is modified during each iteration [58]. In the SOFM algorithm, the vector $\mathbf{x}$ is used to update not only the winning class, but also its neighboring classes according to the following rule:

For each vector $\mathbf{x}$ in the training set:

1) Classify $\mathbf{x}$ according to

$$\mathbf{x} \in C_i \text{ if } \|\mathbf{x} - \mathbf{w}_i\| = \min_j \|\mathbf{x} - \mathbf{w}_j\|. \quad (35)$$

2) Update the features $\mathbf{w}_j$ according to

$$\mathbf{w}_j(t+1) = \begin{cases} \mathbf{w}_j(t) + \alpha(t)[\mathbf{x} - \mathbf{w}_j(t)], & C_j \in N(C_i, t) \\ \mathbf{w}_j(t) & C_j \notin N(C_i, t) \end{cases} \quad (36)$$

where $\mathbf{w}$ is the feature vector, α is a learning parameter in the range $0 < \alpha < 1$, and $N(C_i, t)$ is the set of classes which are in the neighborhood of the winning class C_i at time t. The class features $\mathbf{w}_i$ converge to the class means. The neighborhood of a class is defined according to some distance measure on a topological ordering of the classes. For example, if the classes were ordered on a two-dimensional square grid, the neighborhood of a class could be defined as the set of classes whose Euclidean distances from the class are less than some specified threshold. Initially, the neighborhood may be quite large during training, e.g., half the number of classes or more. As the training progresses, the size of the neighborhood shrinks until, eventually, it only includes the one class. During training, the learning parameter α also shrinks down to a small value (e.g., 0.01) for the fine tuning (convergence) phase of the algorithm.

B. Properties of the SOFM Algorithm

The SOFM algorithm has a number of important properties which make it suitable for use as a codebook generator for vector quantization [5].

1) The set of feature vectors are a good approximation to the original input space.
2) The feature vectors are topologically ordered in the feature map such that the correlation between the feature vectors increases as the distance between them decreases.
3) The density of the feature map corresponds to the density of the input distribution so that regions with a

higher probability density have better resolution than areas with a lower density.

C. Comparison of the SOFM and LBG Algorithms

The LBG algorithm for codebook design and the SOFM algorithm are closely related [59]. In fact, the LBG algorithm is the *batch* equivalent of the SOFM algorithm for a neighborhood size of one, $N(C_i) = \{C_i\}$. The LBG algorithm minimizes the mean squared error (MSE) distortion within a class

$$D_{\mathrm{MSE}}(\mathbf{x}, \mathbf{w}_i) = E[\|\mathbf{x} - \mathbf{w}_i\|^2], x \in C_i \qquad (37)$$

by setting the vector $\mathbf{w}_i$ to the mean of the class. The means are calculated in batch mode, i.e., after n training samples. To update the vector $\mathbf{w}_i$ after every training sample, a gradient descent approach based on minimizing (37) can be employed. The updating rule becomes

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \frac{1}{2}\alpha\nabla D_{\mathrm{MSE}}(x, \mathbf{w}_i(t)) \qquad (38)$$
$$= \mathbf{w}_i(t) + \alpha(t)[x - \mathbf{w}_i(t)]$$

which is equivalent to (36) for $N(C_i) = \{C_i\}$. Therefore, both compute the class means, resulting in a minimum MSE codebook.

A number of researchers [60]–[63] have successfully used the SOFM algorithm to generate VQ codebooks and have demonstrated a number of advantages of using the SOFM algorithm over the classical LBG algorithm. The advantages include less sensitivity to initialization of the codebook, better rate distortion performance, and faster convergence. It has also been shown [64], [65] that the codewords, or weights, converge to the mean of the classes. The resulting codewords are optimal in the sense that the average distortion is minimized.

D. Address Predictive Vector Quantization

Another advantage of using the SOFM algorithm for codebook design has been demonstrated for a variation on VQ called Address Predictive Vector quantization (APVQ) [66]. This technique uses an ordered codebook in which adjacent entries are in some sense correlated. The correlation between adjacent codewords and the correlation between neighboring blocks in an image can be exploited to construct a DPCM encoder in which the input signal is the codeword address. This technique can improve the coding gain and allow for lossy address encoding. When the SOFM algorithm was used to compute a codebook in conjunction with APVQ, 37% fewer bits were required to code an image compared to standard VQ [66].

E. Finite State Vector Quantization

The SOFM has also been successfully used in a finite state vector quantization (FSVQ) scheme [67]. In FSVQ, the codeword index i_n is chosen from a state codebook of the current state S_n. The current state is a function of the previous state S_{n-1} and the previous index i_{n-1}. If the state transition function is a good predictor of the next input,

then each state codebook can be much smaller than that required for a memoryless quantizer. Instead of a separate codebook for each state, Liu and Yun [67] use subsets of a single super-codebook as the state codebooks. The super-codebook is computed using the SOFM algorithm and the state codebooks are neighborhoods within the topological map. The current state codebook is simply the neighborhood around the codewords to the previous input blocks adjacent to the current input block in the image. They found that for a given bit rate, using the FSVQ with the SOFM resulted in an increase in SNR of up to 4.2 dB over a memoryless VQ; alternatively, for the same distortion, the bit rate was reduced by more than a half.

F. Learning Vector Quantization

The SOFM algorithm computes a set of vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K\}$ which are used as the codewords for vector quantization. Learning Vector Quantization is a supervised learning algorithm which can be used to modify the codebook if a set of labeled training data is available [57]. For an input vector $\mathbf{x}$, let the nearest codeword index be i and let j be the class label for the input vector. The codeword $\mathbf{w}_i$ is modified as follows.

- If the label agrees with the codeword assignment, i.e., $i = j$, then

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)[\mathbf{x} - \mathbf{w}_i(t)] \qquad (39)$$

- If the label does not agree with the codeword assignment, i.e., $i \neq j$, then

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \alpha(t)[\mathbf{x} - \mathbf{w}_i(t)] \qquad (40)$$

where α is a learning-rate parameter in the range $0 < \alpha < 1$. Typically, the learning-rate parameter is initialized to, say, 0.1 and then decreased monotonically with each iteration. After a suitable number of iterations, the codebook typically converges and the training is terminated.

G. Hierarchical Vector Quantization

One drawback of conventional VQ coding is the computational load needed during the encoding stage as an exhaustive search is required through the entire codebook for each input vector. An alternative approach is to cascade a number of encoders in a hierarchical manner that trades off accuracy for speed of encoding [65], [68]. Fig. 13 illustrates this approach. For a bit rate of 2 b per sample, or equivalently 8 b per input vector, each stage requires only $2^4 = 16$ codewords for a total search space of 32 codewords. For a conventional encoder with the same bit rate, $2^8 = 256$ codewords and required. In addition, when translational invariance is assumed, only one codebook per stage needs to be computed. In computer experiments with one-dimensional signals, Luttrell [68] found that such a hierarchical encoder incurred only a 0.05 dB increase in distortion compared to an equivalent single stage encoder.

Another approach to reduce the search time is to exploit the organization of the codewords in the topologically
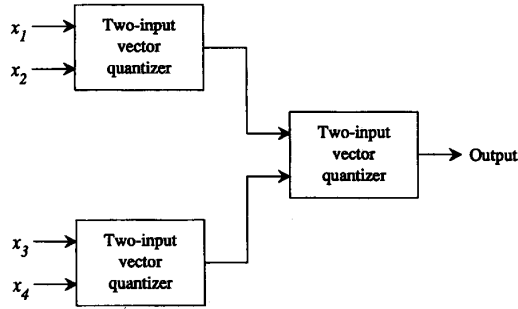
Fig. 13. Two-stage hierarchical vector quantizer using two-input vector quantizers.
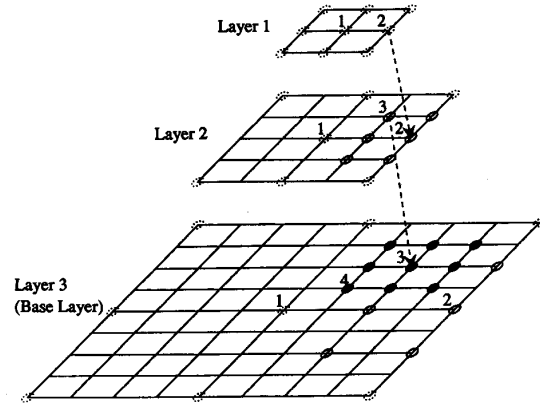

Fig. 14. A two-dimensional codebook to illustrate logarithmic search. Node 1 is root. Node 2 is minimum distance to input vector of neighbors of root node. Node 3 is "winner" of neighbors of node 2 in layer 2. Node 4 is "winner" of neighbors of node 3 in layer 3. Base layer is entire network and shows true locations of abovementioned nodes.

ordered codebook. During the initial period of training in the SOFM algorithm, each input vector modifies a large number of codewords due to the initial large neighborhood function. As training progresses, the number decreases. Some researchers have proposed a training algorithm in which the neighborhood may remain constant but the size of the network grows [69], [70]. In this approach, the size of the network doubles after a number of iterations which grows the network in a hierarchical fashion. The new codewords are inserted between the existing ones and initialized to the mean of their neighbors. This has the effect of drastically reducing the computational requirements during training while still producing a topologically ordered codebook. Luttrell [69] has successfully used this technique to generate VQ codebooks for SAR images.

The resulting codebook can be viewed as a hierarchy of networks [71], [70] with each level half the size of the previous level as illustrated in Fig. 14. The bottom layer contains all the codewords in the network, while each successive layer contains a subset of the codewords of the previous layer. For an input vector, the search begins with the top layer and the winning codeword is found. The neighbors of the winning codeword in the next layer are searched and this process is repeated until the winning codeword in the bottom layer, i.e., the entire codebook, is found. The complexity of the search is logarithmic and is therefore much more efficient than a full search. On experiments with images, Truong [70] found that while this technique dramatically reduced the training and encoding time relative to a full search, the increase in distortion was minor, at most a 0.5 dB decrease in SNR.

Lee and Peterson [64] have developed an approach which dynamically changes the structure and neighborhood relationship between the weights by allowing creation of new weights, merging and deletion of existing weights, and merging of axes in the topological map.

H. Frequency Sensitive Competitive Learning

One of the problems with the LBG algorithm is that the frequency of use of the entries in the codebook can be quite uneven with some codewords being underutilized. In frequency sensitive competitive learning (FSCL) [72]–[74], the distance to the codewords during training is weighted

by an increasing function of the number of wins for each entry during training. In this manner, entries which have had many wins, i.e., which are overutilized, are less likely to the chosen for modification during training. For a Euclidean distance measure, the modified distance measure can be stated as

$$D_{\mathrm{FSCL}}(\mathbf{x}, \mathbf{w}_i) = \mathcal{F}(u_i)\|\mathbf{x} - \mathbf{w}_i\|^2 \qquad (41)$$

where u_i is the count of how many times entry i has won and $\mathcal{F}(\bullet)$ is an increasing function of the win count referred to as the "fairness function."

Ahalt et al. and Krishnamurthy et al. [72], [74] have applied the FSCL algorithm to the design of VQ codebooks for images. Two fairness functions were employed, one equal to the win count

$$\mathcal{F}(u) = u \qquad (42)$$

and the other which decreases to 1 as the training proceeds

$$\mathcal{F}(u) = u^{e^{-t/T}} \qquad (43)$$

where t is the training iteration and T is chosen such that $\mathcal{F}(u) = u^{e^{-1}}$ halfway through the training. They found that the fairness function has a significant effect on the performance of the resulting codebook. The use of (42) resulted in an increase in MSE relative to the LBG codebook, while (43) resulted in MSE approximately the same as for the LBG codebook.

VI. EVALUATION ISSUES

While the scope of this paper has been to review the current research into applying neural network techniques to the task of image compression and summarize the various approaches, we have included, where possible, the claims

as to the performance of the various techniques by their respective authors. However, there is still a need for an objective, unified evaluation of these new approaches. A common set of training images and common testing images from outside the training set would allow the performance of the different algorithms to be compared with each other. In addition, the performance of the state-of-the-art in the "classical" approaches such as linear predictive coding, JPEG using the DCT, and vector quantization on the same test images would provide a set of benchmarks for comparison.

Such an evaluation must address the four issues of training, distortion, bit rate, and complexity.

A. Training

Because these techniques require training to calculate the parameters of the encoder, care must be taken in selecting the common set of training images. They must be representative of the class of images for which the network is to be used. For example, if natural images are to be processed, the training set should include a variety of naturally occurring senses. On the other hand, if artificial images such as synthetic aperture radar (SAR) or magnetic resonance imaging (MRI) are used, then the training set must be representative of that class of images.

The images used for evaluation should also be representative of the class of images used in training but they should not be from the training set. The use of images from outside of the training set allows the *generalization* properties of the networks to the evaluated. For many classical techniques, an image model is assumed. For example, linear predictive coding assumes a *p*th order AR process. For images for which the assumed model is valid and whose statistics are similar, the encoder performs well. When these assumptions are not valid, the performance is degraded. Similarly for neural networks, if a test image has similar attributes to those of the training set, the encoder should perform well and the network is said to generalize well. The degree to which the attributes of an image may deviate from those in the training set before there is a significant degradation in performance provides a measure of the generalization ability of the network.

B. Distortion

With lossy compression, the reconstructed image differs from the original. This difference may result in some visible distortion which may be measured in a number of ways.

1) Mean Squared Error (MSE): A common measure of distortion is the mean squared error (MSE). Its wide-spread use is due mainly to its ease of calculation. Generally, an image with a high MSE has more visible distortion than that with a low MSE. In addition, the minimization of the MSE as an optimality criterion has formed the basis for the development of a large number of successful image compression algorithms.

2) Mean Opinion Score: However, as many researchers have rightly pointed out, despite its popularity as a distor-

tion measure, MSE can be a poor indicator of the subjective quality of reconstruction [14], [11]. As a result, perceptually based criteria may be more appropriate. One example is the *mean opinion score* [11]. A number of subjects view an image and rate its quality on a five point scale of "bad," "poor," "fair," "good," or "excellent." The mean opinion score is simply the average rating assigned by all the subjects.

3) Weighted Mean Squared Error (WMSE): An alternative to the MSE is the weighted mean squared error (WMSE). In this method, the error is weighted by a subjective function which measures the local visibility of distortion. The weighting function can be derived through subjective experiments or models of the human visual system (HVS). The visibility of error can depend on the modulation transfer function (MTF), (the spatial frequency response) of the HVS, the local contrast, and the local texture content [14].

4) End-Use Measures: If the end-use of a class of images is well defined, then the performance under the end-use can be measured. For example, if feature detection is performed on a remote sensing image, then receiver operating characteristics (ROC) curves can be employed [75]. ROC curves plot the relationship between the probability of detection (P_d) and the probability of false alarm (P_{fa}). As the distortion of the compression system increases, P_d will decrease for a given P_{fa}. For medical imaging, the effects of the distortion on the accuracy of diagnosis can be similarly measured [76].

C. Bit Rate

Bit rate can be measured as the average number of bits per pixel for the encoded image. Both the bit rate and the distortion measure must be quoted together for any meaningful comparison of performance.

1) Entropy and Perceptual Entropy: Shannon [77] introduced the concept of entropy as a measure of the average information content of a source. For lossless coding, the lower bound for the minimum number of bits required is given by the entropy of the image. Recently, Jayant introduced the concept of *perceptual entropy* defined as the "fundamental limit to which we can compress the signal with zero (perceived) distortion" [14]. If some distortion is permissible, then the bit rate can be less than the entropy bound. In this case, the rate-distortion relationship must be derived. While this relationship can be analytically derived for the mean squared error distortion measure, for the reasons stated above, the resulting relationship may not give an accurate indication of the performance of an image compression algorithm. For perceptually based distortion measures, the relationship between bit rate and distortion is not so mathematically well defined, and therefore would have to be found experimentally.

2) Overhead: Since a number of neural network approaches to compression are adaptive in nature, the bit rate measure must include any overhead associated with the adaptation mechanism. For example, if a new set of network weights were required at the decoder during processing, the

number of bits in the message notifying the decoder of the change as well as the bits required for the transmission of the new weights, if required, must be accounted for in the average bit rate. In the adaptive transform coding scheme of Section IV–G, the number of bits for the coding of the class index associated with each image block is accounted for in the final bit rate measure.

D. Complexity

The complexity of a compression system is the computational effort required for encoding and decoding images. A typical measure of complexity is the number of floating-point operations (flops) per pixel required to encode and decode an image. Associated with complexity is speed. Speed is a function of both complexity and implementation and may be substantially improved through the use of an efficient implementation of an image compression algorithm with a given computational complexity. For example, if a neural network were to be implemented in a parallel architecture, significant improvements in speed over a serial implementation would be realized due to the inherently parallel design of the neural network.

1) Encoding and Decoding: In point-to-point transmission, an image is encoded once, transmitted, and then decoded once. In such an environment, the complexities of both the encoder and the decoder are equally important. However, this environment does not apply to many current uses of digital imaging. In a broadcast environment or a database retrieval environment, an image is encoded once and decoded *many* times. For these environments, it is the complexity of the decoder that is of greater importance.

2) Training: For neural network approaches, the complexity of the training algorithm must also be examined. The convergence properties are important as they affect the training complexity. They include the conditions of convergence, convergence rate, and the number of iterations required for typical training data. As well, the sensitivity of the network to initial conditions must be examined.

VII. Summary and Discussion

Investigations into the application of neural networks to the problem of image compression have produced some promising results. By their very nature, neural networks are well suited to the task of processing image data. The characteristics of artificial neural networks which include a massively parallel structure, a high degree of interconnection, the propensity for storing experiential knowledge, and the ability to self-organize, parallel many of the characteristics of our own visual system. In contrast, standard approaches to the processing of image data have been based on a serial paradigm of information processing which is more suited to sequential information such as language. As a result, neural network approaches to image compression have been shown to perform as well as or better than standard approaches.

The nonlinear nature of neural networks can be exploited to design nonlinear predictors for predictive coding. Multi-

layer perceptrons trained via the backpropagation algorithm have been shown to increase predictive gain relative to linear predictors. Other networks based on higher-order statistical models and recurrent models have similarly shown improvements over linear predictors.

Hebbian learning has formed the basis for a number of iterative methods of extracting the principal components of image data for use as the basis images in block transform coding. Both the GHA and the APEX algorithms have been shown to converge to the M principal components. These algorithms and their variants have a number of advantages over standard eigendecomposition of the sample autocovariance matrix. When only the first few principal components are required, significant computational savings can be realized. Their iterative nature can allow the basis images to adapt on a block-per-block scale or over long-term variations. As well, memory requirements are reduced as there is no need to store the entire data set or its higher-order statistics. Autoassociative networks have also been successfully used to compress image data using a nonlinear transformation.

The SOFM algorithm has been applied in a number of ways in the area of VQ. Its ability to form ordered topological feature maps in the self-organizing fashion has given it a number of advantages over the standard LBG algorithm for the generation of VQ codebooks. It has been found to be less sensitive to initial conditions, have fast convergence properties and have the ability to produce a lower mean distortion codebook. As well, hierarchical VQ and predictive VQ approaches have made use of the ordered nature of the codebook to reduce both search complexity and distortion.

With the wide variety of approaches to applying neural network methods to the problem of image compression, there is a compelling need for a comprehensive evaluation of the new techniques. Such an evaluation would require a common set of training images and a common set of test images from outside the training set. The performance of the neural network approaches should be compared to that of similar "classical" techniques. The comparisons could be based on measures of distortion including MSE and perceptual distortion measures, bit rate, and the complexity of encoding, decoding and training.

Despite some unanswered questions, many neural network approaches to image compression show much promise. However, the full potential of these approaches will not be realized until they are implemented in their true parallel form. Most of the implementations used in the above research have been based on simulations on serial computers. With the development of VLSI implementations for many neural network architectures, the speed for both training and coding will dramatically increase. Furthermore, the cost savings for VLSI implementation will be appreciable. When such hardware is readily available, then the day in which artificial systems can represent and communicate image data in less than "a thousand words" with the same ease as ourselves will be that much closer to reality.

REFERENCES

[1] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing.* MIT Press: Cambridge, MA, 1986.
[2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine,* vol. 4, pp. 4–22, Apr. 1987.
[3] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 3rd ed., 1988.
[4] D. R. Hush and B. G. Horne, "Progress in supervised neural networks: What's new since Lippmann," *IEEE Signal Processing Magazine,* vol. 10, pp. 8–39, Jan. 1993.
[5] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan, 1994.
[6] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proc. IEEE,* vol. 68, pp. 366–406, Mar. 1980.
[7] A. K. Jain, "Image data compression: A review," *Proc. IEEE,* vol. 69, pp. 349–389, Mar. 1981.
[8] A. Rosenfeld and A. C. Kak, *Digital Picture Processing,* vols. I and II. San Diego CA: Academic, 2nd ed., 1982.
[9] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs, NJ: Prentice-Hall, 1984.
[10] H. G. Musmann, P, Pirsch, and H.-J. Grallert, "Advances in picture coding," *Proc. IEEE,* vol. 73, pp. 523–548, Apr. 1985.
[11] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression.* New York: Plenum, 1988.
[12] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer, 1992.
[13] J. A Storer and M. Cohn, Eds., *Proc. Data Compression Conf.,* Snowbird, UT, Mar./Apr. 1993, IEEE Computer Society Press.
[14] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proc. IEEE,* vol. 81, pp. 1385–1421, Oct. 1993.
[15] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications.* New York: Academic, 1990.
[16] G. K. Wallace, "Overview of the JPEG (ISO/CCITT) still image compression standard," in *Proc. SPIE,* vol. 1244, pp. 220–233, Feb. 1990.
[17] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.,* vol. C-28, pp. 84–95, Jan. 1980.
[18] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine,* vol. 1, pp. 4–29, 1984.
[19] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.,* vol. 36, pp. 957–971, Aug. 1988.
[20] H. Abut, Ed., *Vector Quantization.* New York: IEEE Press, 1990.
[21] S. A. Dianat, N. M. Nasrabadi, and S. Venkataraman, "A nonlinear predictor for differential pulse-code encoder (DPCM) using artificial neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc. '91,* pp. 2793–2796, Toronto, Canada, May 1991.
[22] J. Li and C. N. Manikopoulos, "Nonlinear predictor in image coding with DPCM," *Electron. Lett.,* vol. 26, pp. 1357–1359, Aug. 1990.
[23] C. N. Manikopoulos, "Neural network approach to DPCM system design for image coding," *IEE Proc.-I,* vol. 139, pp. 501–507, Oct. 1992.
[24] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks.* Reading MA: Addison–Wesley, 1989.
[25] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," to be published in *IEEE Trans. Signal Processing,* 1995.
[26] S. Haykin, *Adaptive Filter Theory.* Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1991.
[27] D. O. Hebb, *The Organization of Behavior.* New York: Wiley, 1949.
[28] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biol.,* vol. 15, pp. 267–273, 1982.
[29] ——, "Neural networks, principal components, and subspaces," *Int. J. Neural Syst.,* vol. 1, pp. 61–68, 1989.
[30] ——, "Principal components, minor components, and linear neural networks," *Neural Networks,* vol. 5, pp. 927–935, 1992.
[31] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. and Appl.,* vol. 106, pp. 69–84, 1985.
[32] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feed forward neural network," *Neural Networks,* vol. 2, pp. 459–473, 1989.

[33] ——, "An optimality principle for unsupervised learning," D. S. Touretzky, Ed., *Advances in Neural Information Processing Systems 1,* pp. 11–19, 1989.
[34] ——, "Analysis of the two-dimensional receptive fields learned by the generalized hebbian algorithm in response to random input," *Biol. Cybern.,* vol. 63, pp. 221–228, 1990.
[35] P. Földiák, "Adaptive network for optimal linear feature extraction," in *Int. Joint Conf. on Neural Networks,* vol. 1, pp. 401–405, Washington DC, 1989.
[36] H. Barlow and P. Földiák, "Adaptation and decorrelation in the cortex," R. Durbin, C. Miall, and G. Michison, Eds., *The Computing Neuron.* Reading, MA: Addison–Wesley, pp. 54–72,1989.
[37] S. Y. Kung and K. I. Diamantaras, "A Neural network learning algorithm for adaptive principal component extraction (APEX)," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '90,* pp. 861–864, Alburqurque, NM, Apr. 1990.
[38] K. I. Diamantaras, "Principal component learning networks and applications," Ph.D. dissertation, Princeton Univ., Oct. 1992.
[39] S. Y. Kung, K. I. Diamantaras, and J. S. Taur, "Adaptive principal component extraction (APEX) and applications," *IEEE Trans. Signal Process.,* vol. 42, pp. 1202–1217, May 1994.
[40] H. Chen and R. Liu, "Adaptive distributed orthogonalization process for principal components analysis," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Process. '92,* pp. II 283–296, San Francisco, CA, Mar. 1992.
[41] L. Xu and A. Yuille, "Robust principal component analysis by self-organizing rules based on statistical physics approach," Tech. Rep. 92–93, Harvard Robotics Lab., Feb. 1992.
[42] E. Oja, *Subspace Methods of Pattern Recognition.* Letchworth, UK: Research Studies Press, 1983.
[43] S. Becker and M. Plumbley, "Unsupervised neural network learning procedures for feature extraction and classification," submitted to *Appl. Intell.,* 1993.
[44] G. W. Cottrell and P. Munro, "Principal components analysis of images via back propagation," in *SPIE Vol. 1001 Visual Commun. and Image Process. '88,* pp. 1070–1077, 1988.
[45] G. W. Cottrell, P. Munro, and D. Zipser, "Learning internal representations from grayscale images: An example of extensional programming," in *9th Annu. Conf. of the Cognitive Soc.,* pp. 462–473, July 1987.
[46] G. W. Cottrell and J. Metcalfe, "EMPATH: Face, emotion, and gender recognition using holons," in *Advances in Neural Information Processing Systems,* David S. Touretzky, Ed., vol. 3, pp. 564–571, 1991.
[47] B. A Golomb, D. T. Lawrence, and T. J. Sejnowski, "SEXNET: A neural network identifies sex from human faces," in *Advances in Neural Information Processing Systems,* David S. Touretzky, Ed., vol. 3, pp. 572–577, 1991.
[48] G. L. Sicuranzi, G. Ramponi, and S. Marsi, "Artificial neural network for image compression," *Electron. Lett.,* vol. 26, pp. 477–479, Mar. 1990.
[49] D. Anthony, E. Hines, D. Taylor, and J. Barham, "A study of data compression using neural network and principal component analysis," in *Colloquium on Biomedical Applications of Digital Signal Processing,* pp. 1–5, 1989.
[50] R. Kohno, M. Arai, and H. Imai, "Image compression using a neural network with learning capability of variable function of a neural unit," in *SPIE Vol. 1360, Visual Commun. and Image Proc. '90,* pp. 69–75, 1990.
[51] A. Namphol, M. Arozullah, and S. Chin, "Higher order data compression with neural networks," in *Proc. Int. Joint Conf. on Neural Networks '91,* pp. I 55–59, 1991.
[52] D. DeMers and G. W. Cottrell, "Non-linear dimensionality reduction," in *Advances in Neural Information Processing Systems,* vol. 5, 1993.
[53] L. E. Russo and E.C. Real, "Image compression using an outer product neural network," in *Proc. IEEE. Int. Conf. Acoust., Speech, and Signal Process. '92,* pp. II 377–380, San Francisco, CA, Mar. 1992.
[54] R. D. Dony and S. Haykin, "Optimally integrated adaptive learning," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '93,* pp. I 609–612, Minneapolis, MN, Apr. 1993.
[55] ——, "Optimally adaptive transform coding," to be published in *IEEE Trans. Image Processing,* 1995.
[56] ——, "Self-organizing segmentor and feature extractor," in *Proc. IEEE Int. Conf. on Image Processing,* Austin, TX, Nov. 1994, pp. III 898–902.

302

[57] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.

[58] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[59] S. P. Luttrell, "Self-organization: A derivation from first principle of a class of learning algorithms," in *IEEE Conf. on Neural Networks*, pp. 495–498, Washington DC, 1989.

[60] N. M. Nasrabadi and Y. Feng, "Vector quantization of image based upon a neural-network clustering algorithm," in *SPIE Vol. 1001 Visual Commun. and Image Processing '88*, pp. 207–213, 1988.

[61] —, "Vector quantization of image based upon the Kohonen self-organizing feature map," in *Proc. IEEE Int. Conf. on Neural Networks*, pp. I 101–105, San Diego, CA, July 1988.

[62] J. D. McAuliffe, L. E. Atlas, and C. Rivera, "A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '90*, pp. 2293–2296, Alburquerque, NM, Apr. 1990.

[63] M. Manohar and J. C. Tilton, "Compression of remotely sensed images using self organized feature maps," H. Wechsler, Ed., in *Neural Networks for Perception, vol. 1: Human and Machine Perception,*. San Diego CA: Academic, 1992, pp. 345–367.

[64] T.-C. Lee and A. M. Peterson, "Adaptive vector quantization using a self-development neural network," *IEEE J. on Selected Areas in Commun.*, vol. 8, pp. 1458–1471, Oct. 1990.

[65] J. Li and C. N. Manikopoulos, "Multi-stage vector quantization based on the self-organizing feature maps," in *SPIE Vol. 1199, Visual Commun. and Image Processing IV (1989)*, pp. 1046–1055, 1989.

[66] G. Poggi and E. Sasso, "Codebook ordering techniques for address-predictive VQ," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing '93*, pp. V 586–589, Minneapolis, MN, Apr. 1993.

[67] H. Liu and D. Y. Y. Yun, "Self-organizing finite state vector quantization for image coding," J. Alspector, R. Goodman, and T. X. Brown, Eds., in *Proc. of the Int. Workshop on Applications of Neural Networks to Telecommunications.* Hillsdale, NJ: Lawrence Erlbaum Assoc., pp. 176–182, 1993.

[68] S. P. Luttrell, "Hierarchical vector quantization," *IEE Proc.* (London), vol. 136 (Part I), pp. 405–413, 1989.

[69] —, "Image compression using a neural network," in *Proc. IGARSS '88*, Sept. 1988, pp. 1231–1238, Edinburgh, Scotland.

[70] K. K. Truong, "Multilayer Kohonen image codebooks with a logarithmic search complexity," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '91*, pp. 2789–2792, Toronto, Canada, May 1991.

[71] K. K. Truong and R. M. Mersereau, "Structural image codebooks and the self-organizing feature map algorithm," in *Proc. IEEE Conf. Acoust., Speech, and Signal Process. '90*, pp. 2289–2292, Alburquerque, NM, Apr. 1990.

[72] S. C. Ahalt, P. Chen, and A. K. Krishnamurthy, "Performance analysis of two image vector quantization techniques," in *Proc. IEEE Int. Joint Conf. on Neural Networks*, pp. I 169–175, 1989.

[73] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, pp. 277–290, 1990.

[74] —, "Neural networks for vector quantization of speech and images," *IEEE J. on Selected Areas in Commun.*, vol. 8, pp. 1449–1457, Oct. 1990.

[75] J. P. Egan, *Signal Detection Theory and ROC-Analysis.* New York: Academic, 1975.

[76] P. C. Cosman, C. Tseng, R. M. Gray, R. A. Olshen, L. E. Moses, H. C. Davidson, C. J. Bergin, and E. A. Riskin, "Tree-structured vector quantization of CT chest scans: Image quantity and diagnostic accuracy," *IEEE Trans. Medical Imaging*, vol. 12, pp. 727–739, Dec. 1993.

[77] C. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *IRE Natl. Conv. Rec.*, pp. 142–163, 1959.

Robert D. Dony (Student Member, IEEE) received the B.A.Sc. and M.A.Sc. degrees in 1986 and 1988 in systems design engineering from the University of Waterloo, Ontario, Canada.

In 1988 he joined Imaging Research, Inc., where he was Manager of Imaging Development. He is now working toward the Ph.D. degree in electrical engineering at McMaster University, Hamilton, Ontario, Canada. His research interests include image processing, image compression, neural networks, and both artificial and biological learning systems.

Simon Haykin (Fellow, IEEE) received the B.Sc. (first-class honors) in 1953, the Ph.D. in 1956, and the D.Sc. in 1967, all in electrical engineering, from the University of Birmingham, England.

He is the founding Director of the Communications Research Laboratory and Professor of Electrical and Computer Engineering at McMaster University, Hamilton, Ontario. His research interests include nonlinear dynamics, neural networks, and adaptive filters and their applications. He is the Editor of *Adaptive and Learning Systems for Signal Processing, Communications and Control*, a new series of books for Wiley-Interscience.

In 1980 he was elected Fellow of the Royal Society of Canada, and in 1986 he was awarded the McNaughton Gold Medal of the IEEE.