

# Neural Network Architectures for Vector Prediction

SYED A. RIZVI, MEMBER, IEEE, LIN-CHENG WANG, STUDENT MEMBER, IEEE,  
AND NASSER M. NASRABADI, SENIOR MEMBER, IEEE

A vector predictor is an integral part of a predictive vector quantization (PVQ) coding scheme. However, the performance of a classical linear vector predictor is limited by its ability to exploit only the linear correlation between the blocks. Furthermore, its performance deteriorates as the vector dimension (block size) is increased, especially when predicting blocks that contain edge information. However, a nonlinear predictor exploits the higher-order correlations among the neighboring blocks, and can predict edge blocks with increased accuracy. Because the conventional techniques for designing a nonlinear predictor are extremely complex and suboptimal due to the absence of a suitable model for the source data, it is necessary to investigate new procedures in order to design nonlinear vector predictors. In this paper, we have investigated several neural network architectures that can be used to implement a nonlinear vector predictor, including the Multilayer Perceptron, the Functional Link network, and the Radial Basis Function network. We also evaluated and compared the performance of these neural network predictors with that of a linear vector predictor. Our experimental results show that a neural network predictor can predict the blocks containing edges with a higher accuracy than a linear predictor. However, the performance of a neural network predictor is comparable to that of a linear predictor for predicting the stationary and shade blocks.

## I. INTRODUCTION

The typical block size used in a vector quantizer (VQ) scheme is limited to a low dimension, due to the computational complexity of the VQ encoder. A block size of  $4 \times 4$  is commonly used; therefore, in highly correlated data such as images, a high correlation exists among the neighboring blocks. This interblock correlation can be exploited by incorporating memory into the VQ scheme. Examples of VQ with memory are finite-state VQ (FSVQ) [1], [2], predictive VQ (PVQ) [3]–[5] and predictive residual VQ (PRVQ) [6]. A PVQ scheme uses a predictor that estimates the current block from the previously encoded blocks, and the residual vector (the difference between the original

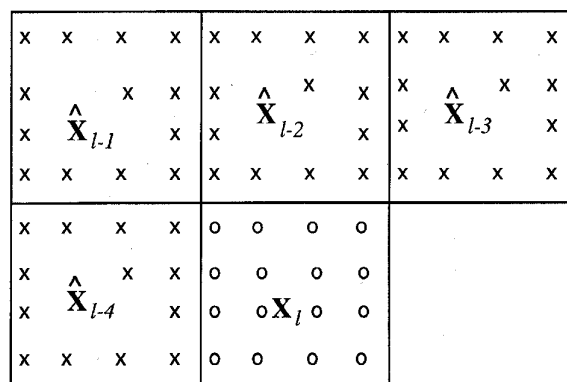


Fig. 1. Geometry of the block (vector) prediction.

and the predicted vector) is then vector quantized using a relatively small codebook. Essentially, the predictor decorrelates the neighboring blocks. A good prediction for the current vector would result in a residual vector that has very little correlation with its neighboring blocks; therefore, this residual code vector could be efficiently encoded using a VQ codebook. The role of the predictor can be viewed from another perspective: A good prediction of the current vector would reduce the variations (dynamic range) of the components of the residual vector. The residual vectors with components that have small variations could be encoded more efficiently than the residual vectors with components that vary a great deal.

The performance of the predictor for the blocks containing edges is an important issue in the design of a vector predictor. These blocks have pixels with a large dynamic range, and are very difficult to encode by a simple VQ. The performance of a coding scheme is usually reported in terms of the mean square error (MSE); however, the MSE is not generally a good criterion for representing the visual quality of the reconstructed images. For example, in low-to-moderately-detailed images, the edge blocks comprise a small fraction (10–15%) of the total number of blocks in the image. Therefore, the poorly reconstructed edge blocks could substantially deteriorate the visual quality

Manuscript received October 1, 1995; revised February 1, 1996.  
S. A. Rizvi and L.-C. Wang are with the Department of Electrical and Computer Engineering, State University of New York at Buffalo, Amherst, NY 14260 USA.  
N. Nasrabadi is with the Department of the Army, U.S. Army Research Laboratory, Fort Belvoir, VA 22060-5838 USA.  
Publisher Item Identifier S 0018-9219(96)07172-1.

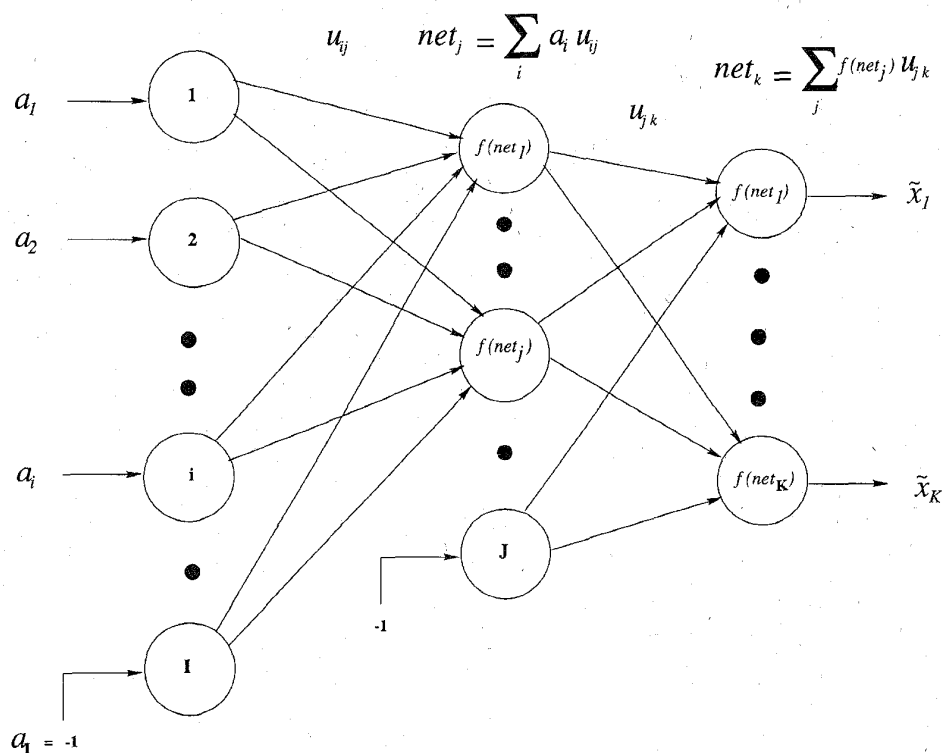


Fig. 2. Three-layer perceptron used as a vector predictor.

of the reconstructed image, even though they have little effect on the overall MSE. From a predictor point of view, a poorly predicted edge block leaves more work for the subsequent VQ operation. On the other hand, the sufficiently accurate prediction of an edge block could generate a residual vector with small variations among its components. The subsequent VQ operation is now more effective in encoding this residual vector than the residual vector generated by a poor predictor. In spite of the fact that the MSE is a poor criterion, it still provides a useful tool for designing a predictor and a quantizer.

Although a linear predictor estimates background and shade blocks (vectors) with sufficiently high accuracy, it still predicts relatively poorly for the blocks containing edges. Basically, linear predictors do not exploit the higher-order structural correlations (e.g., some form of edge continuity or textural signatures). This results in a rather poor performance by linear predictors for blocks containing edges or textural information, and warrants the use of a nonlinear vector predictor that can exploit the higher-order correlations to improve the perceptual quality of the predicted (reconstructed) image.

The optimal prediction of a vector (in a minimum mean-square sense), given a finite number of past vectors, is its conditional expectation [1], but the computation of the conditional expectation requires the knowledge of the joint probability of the *current* vector and the *past* vectors, which is not known. This fact makes the conditional expecta-

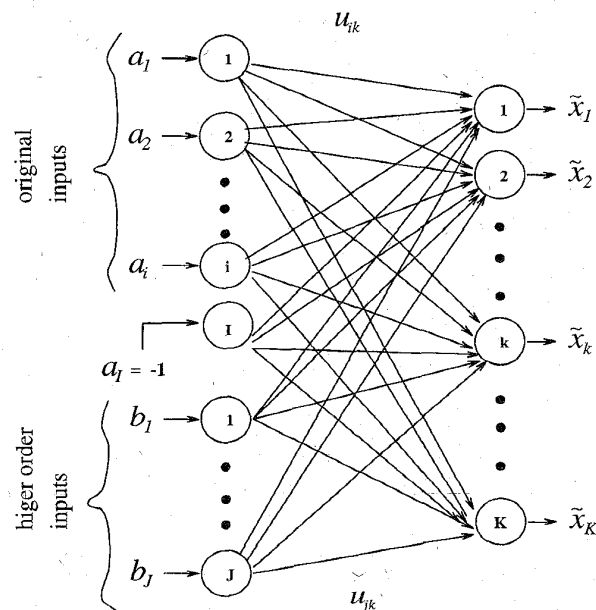


Fig. 3. An FL network.

tion mathematically intractable. Therefore, the methods for designing the nonlinear vector predictors are suboptimal, and they can only attempt to approximate the conditional expectation of the *current* vector.

Although conventional methods for designing a nonlinear predictor are usually very complicated and have received

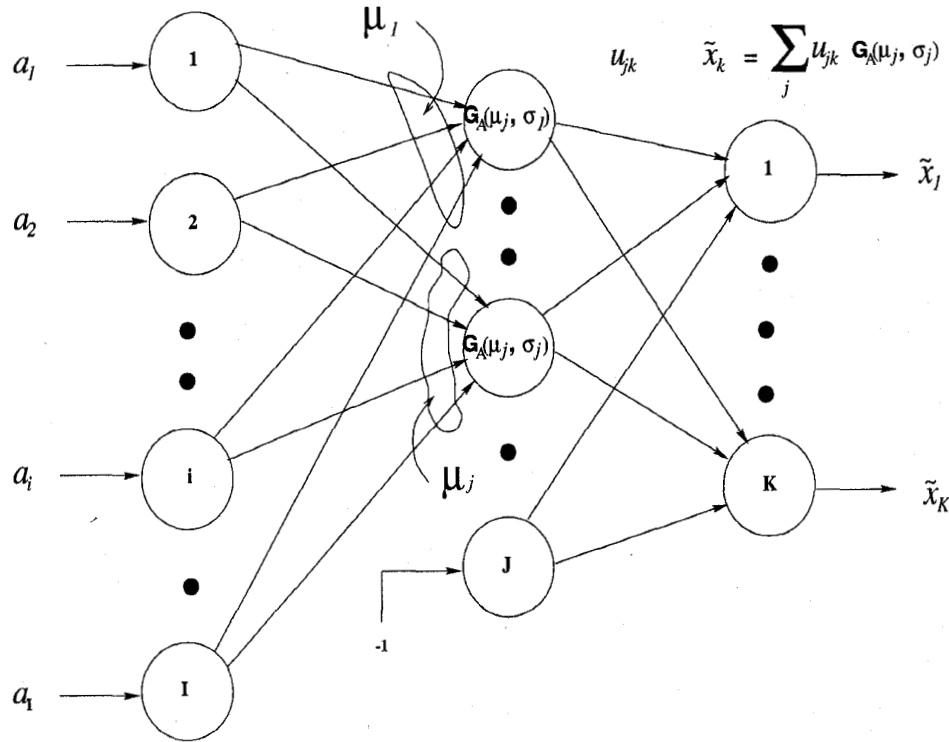


Fig. 4. RBF neural network.

little attention in research literature, there have been a few attempts to design a nonlinear predictor, based on training processes that do not require any modeling of the source data. This work includes a VQ-based nonlinear predictor developed by Wang *et al.* [7], and a neural network-based predictor developed by Tishby [8] for speech processing. In image coding applications, Dianat *et al.* [9] developed a scalar nonlinear predictor for DPCM employing a multilayer perceptron. Mohsenian *et al.* [4], extended this approach for designing a vector predictor. Manikopoulos [10] also proposed a similar neural network predictor for DPCM. The neural network approach provides a good solution to such a mathematically intractable problem, because its design is based on training and, therefore, no statistical assumptions (modeling) are needed for the source data.

For this paper, we investigated several neural network architectures for designing a nonlinear vector predictor. A neural network predictor implements a nonlinear mapping of an  $n$ -dimensional input vector to a  $k$ -dimensional output vector, where  $n$  and  $k$  need not be the same. The  $n$ -dimensional input is made up of a finite number of past vectors so that  $n = m \times k$ , where  $m$  represents the number of past vectors (the size of memory) used for predicting the current vector. Neural network predictors are designed by using a *supervised* training procedure where the network learns the features of the training data (assuming the training set is a good representation of the test data where the neural network will be employed)

by adapting its synapses (weights). Once the training is complete, the *representative features* are stored by the network in the form of weights between the processing units. These *features* are used when a group of past observations (vectors) is presented to *approximate* the current vector based on the features detected in the input data (past vectors). In this way, the neural network approach greatly simplifies the problem of designing a nonlinear vector predictor.

The neural network architectures, that we investigated include the multilayer perceptron (MLP), the functional link (FL) neural network, and the radial basis function (RBF) network. We evaluated the performance of these networks and compared them with that of a linear predictor [1], both in terms of visual quality and the MSE for the predicted image. Experimental results show that all of the neural network predictors outperform the conventional linear predictor.

This paper is organized as follows: Section II briefly reviews the optimal nonlinear predictor; Section III presents several neural network architectures for designing a nonlinear vector predictor; Section IV presents the experimental results; and Section V concludes the paper with recommendations for future research.

## II. OPTIMAL NONLINEAR PREDICTOR

In this section, we briefly review the necessary conditions for an *optimal* nonlinear vector predictor and discuss the problems encountered in designing such a predictor. Note

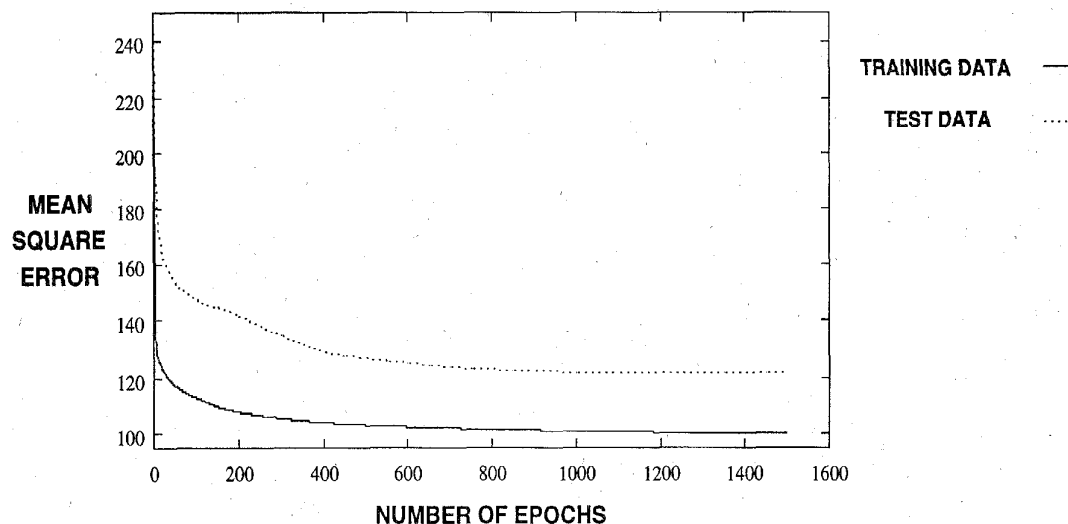


Fig. 5. MSE versus number of epochs for MLP predictor with one hidden layer and 31 hidden neurons.

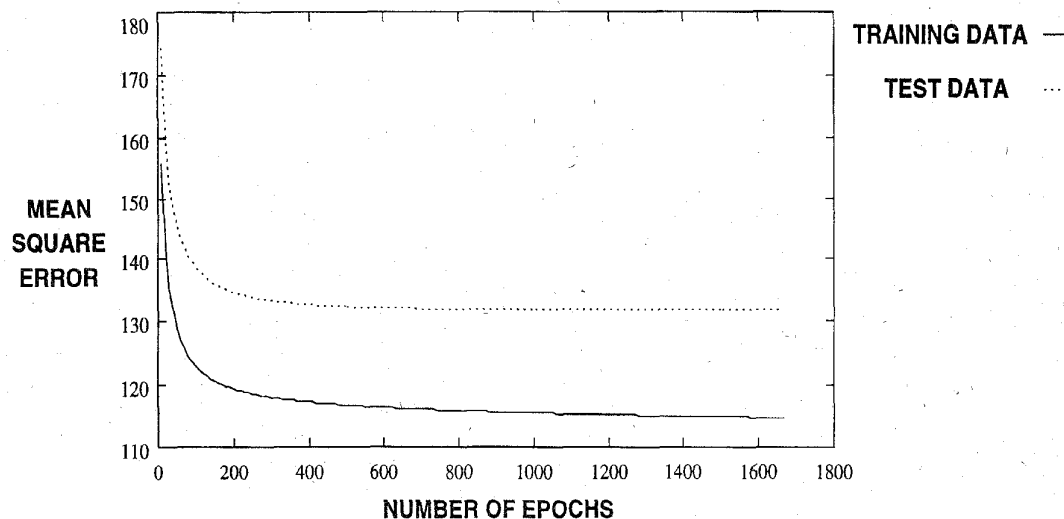


Fig. 6. MSE versus number of epochs for FL predictor consisting of 64 original and 364 higher-order input terms.

that in all of the designs presented in this paper the prediction is based on the four *past* blocks (causal) shown in Fig. 1. This is necessary because a PVQ decoder can only use the previously reconstructed blocks to predict the current block. The input image is partitioned into nonoverlapping contiguous blocks of size  $k = r \times r$  pixels, where each block is represented by a  $K$ -dimensional column vector  $\mathbf{X}_l = [x_k; k = 1, 2, \dots, K]$ . Furthermore, let  $\mathbf{X}_l$  and  $\hat{\mathbf{X}}_l$  represent the current and the predicted blocks, respectively. The predictor can be designed as a mapping  $\Psi(\cdot)$  given by

$$\hat{\mathbf{X}}_l = \Psi(\hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \hat{\mathbf{X}}_{l-3}, \hat{\mathbf{X}}_{l-4}) \quad (1)$$

where the four neighboring blocks in the horizontal, vertical and diagonal directions (shown in Fig. 1) are used to predict  $\hat{\mathbf{X}}_l$ . The sequence  $\{\hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \hat{\mathbf{X}}_{l-3}, \hat{\mathbf{X}}_{l-4}\}$  rep-

resents the *causal* blocks. In the actual implementation of a PVQ encoder, these blocks are the previously reconstructed blocks. Note that the subscripts  $l, l-1, l-2$ , etc., do not represent the order in which the input blocks are presented for encoding. In real implementations, an appropriate buffer is used to keep these blocks in the correct order for generating the input to the predictor. We now briefly review the optimal nonlinear predictor. An optimal nonlinear prediction of the vector  $\mathbf{X}_l$  given the  $m$  previous vectors  $\hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \dots, \hat{\mathbf{X}}_{l-m}$  is given by the conditional expectation [1]

$$\hat{\mathbf{X}}_l = E[\mathbf{X}_l | \hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \dots, \hat{\mathbf{X}}_{l-m}] \quad (2)$$

where  $\hat{\mathbf{X}}_l$  represents the prediction of the vector  $\mathbf{X}_l$  and  $E$  is the expectation operator. The conditional expectation in

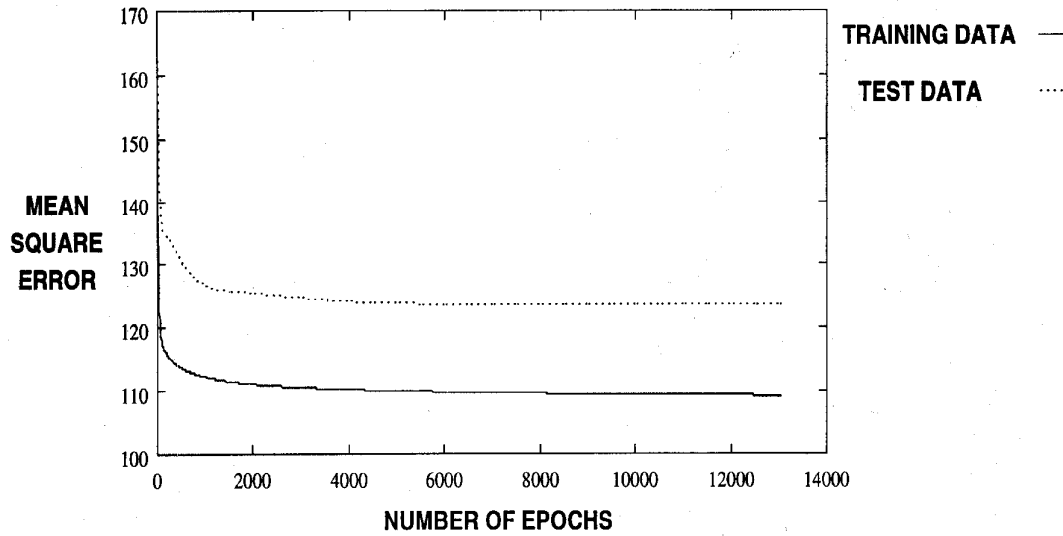


Fig. 7. MSE versus number of epochs for RBF predictor with 30 hidden units (basis functions).

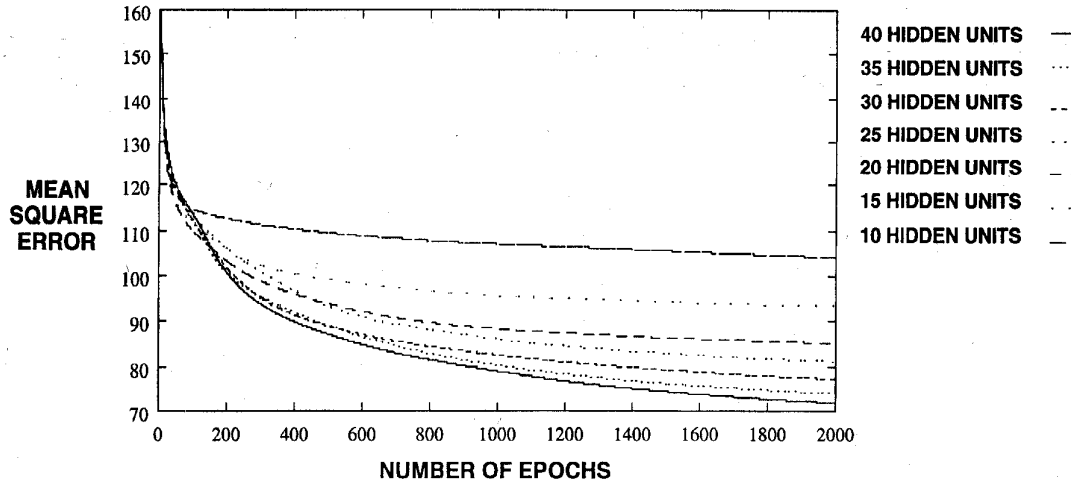


Fig. 8. Performance of the multilayer perceptron with different numbers of hidden units. (Only one training image was used in this experiment.)

(2) is given by

$$E[X_l | \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m}] = \sum_k X_k p((X_l = X_k) | \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m}) \quad (3)$$

where  $p((X_l = X_k) | \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m})$  represents the conditional probability that  $X_l = X_k$  given the  $m$  past vectors  $\hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m}$ . Note that the vectors  $X_l, X_{l-1}, \dots$ , etc., are in general, not independent (prediction would no longer be useful if they were independent).

The conditional probability in (3) can be rewritten as (see (4) at the bottom of the page).

The probabilities on the right-hand side of (4) are not known *a priori* and therefore, (2) is mathematically intractable. Suboptimal solutions could be found using a suitable model for the source data, and then computing the joint/conditional probabilities in (4). However, the lack of precise modeling precludes any useful solution by the direct use of (2). The nonlinear predictor can be viewed from a different perspective. It can be seen from (3) that the best nonlinear predictor is the one that maximizes the

$$p((X_l = X_k) | \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m}) = \frac{p(X_k, \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m})}{p(\hat{X}_{l-1})p(\hat{X}_{l-2} | \hat{X}_{l-1}) \dots p(\hat{X}_{l-m} | \hat{X}_{l-1}, \hat{X}_{l-2}, \dots, \hat{X}_{l-m+1})} \quad (4)$$

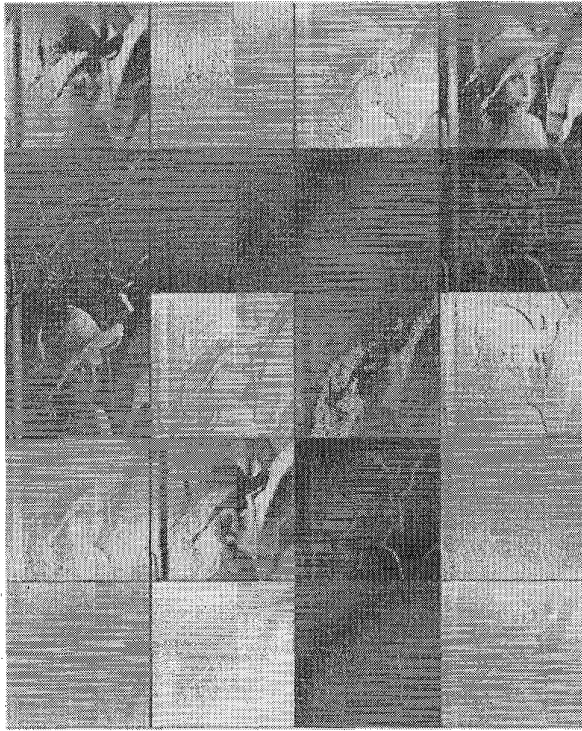


Fig. 9. Output of hidden layer of an MLP predictor with 20 hidden units. The contrast of this image has been increased to display all feature extractors present in the MLP predictor.

conditional probability given by (4). This is equivalent to minimizing the conditional expectation of the prediction error: a nonlinear predictor minimizes the conditional expectation given by

$$E[\|\mathbf{X}_l - \hat{\mathbf{X}}_l\|^2 | \hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \dots, \hat{\mathbf{X}}_{l-m+1}]. \quad (5)$$

A neural network predictor does not minimize the conditional expectation given by (5) directly. However, it does *approximate* the minimization of this conditional expectation [11].

### III. NEURAL NETWORK ARCHITECTURES FOR VECTOR PREDICTION

In this section, we first discuss the use of a multilayer perceptron as a vector predictor followed by the FL and the RBF neural networks.

#### A. Multilayer Perceptron

A neural network vector predictor based on a three-layer perceptron with one hidden layer is shown in Fig. 2. In Fig. 2,  $u_{ij}$  represents the weight associated with the connection from a neuron  $i$  in the first (input) layer to a neuron  $j$  in the second (hidden) layer. Similarly,  $u_{jk}$  represents the weight associated with the connection from a neuron  $j$  in the hidden layer to a neuron  $k$  in the final (output) layer. The block sequence given by  $\{\hat{\mathbf{X}}_{l-1}, \hat{\mathbf{X}}_{l-2}, \hat{\mathbf{X}}_{l-3}, \hat{\mathbf{X}}_{l-4}\}$  along with the biased input  $a_I$  (in Fig. 2), forms the input vector to the network, which is represented by a column vector

$\mathbf{A}_l = [a_i; i = 1, 2, \dots, I]$ , where  $I = (4 \times r \times r) + 1$ . The output of the  $j$ th neuron in the hidden layer  $h_j$ , is given by

$$h_j = f(\text{net}_j) \quad (6)$$

where

$$\text{net}_j = \sum_{i=1}^I a_i u_{ij}. \quad (7)$$

The function  $f(\text{net})$  is a nonlinear sigmoid function given by

$$f(\text{net}) = \frac{1}{1 + e^{-\beta \text{net}}} \quad (8)$$

where  $\beta$  is called the steepness factor. Similarly, the output of the  $k$ th neuron in the output layer  $\tilde{x}_k$  is given by

$$\tilde{x}_k = f(\text{net}_k) \quad (9)$$

where

$$\text{net}_k = \sum_{j=1}^J h_j u_{jk}. \quad (10)$$

The neural network is trained by using the standard back-propagation learning algorithm [12]. The algorithm for designing the vector predictor is as follows.

*Algorithm 1:* Given the training set  $S = [\mathbf{X}_l; l = 1, 2, \dots, M]$  and the test set  $T = [\mathbf{X}_i; i = 1, 2, \dots, N]$ , set the initial weights associated with the output and hidden layer of the multilayer perceptron to small random values. Select the learning rate  $\eta$ , the momentum  $\mu$ , and the maximum number of training epochs  $\tau_{\max}$ . Choose an initial distortion  $D_{\text{test}}^0$  for the test image to be a large number. Set the number of epochs  $\tau = 0$ .

- Step 1: Set  $l \rightarrow 1$ .
- Step 2: Compute the predicted vector  $\hat{\mathbf{X}}_l = [\tilde{x}_k; k = 1, 2, \dots, K]$  (based on the four neighboring blocks shown in Fig. 1) using (6)–(10).
- Step 3: Update the weights associated with the output layer of the multilayer perceptron  $u_{jk}$  as

$$u_{jk}^{t+1} = u_{jk}^t + \eta(x_k - \tilde{x}_k)\tilde{x}_k(1 - \tilde{x}_k)h_j + \mu(u_{jk}^t - u_{jk}^{t-1})$$

where  $t$  is the time index.

- Step 4: Update the weights associated with the hidden layer of the multilayer perceptron  $u_{ij}$  as

$$\begin{aligned} u_{ij}^{t+1} = & u_{ij}^t + \mu(u_{ij}^t - u_{ij}^{t-1}) \\ & + \eta a_i h_j (1 - h_j) \sum_{k=1}^K (x_k - \tilde{x}_k) \\ & \times \tilde{x}_k (1 - \tilde{x}_k) u_{jk}^t. \end{aligned}$$

- Step 5: Set  $l \rightarrow l+1$ . If  $l < M$ , go to Step 2; otherwise, go to next STEP.
- Step 6: Set  $\tau \rightarrow \tau + 1$ . Compute the average distortion  $D_{\text{test}}^\tau$  for the predicted test image as

$$D_{\text{test}}^\tau = \frac{1}{K \times N} \sum_i^N \|\mathbf{X}_i - \hat{\mathbf{X}}_i\|^2.$$

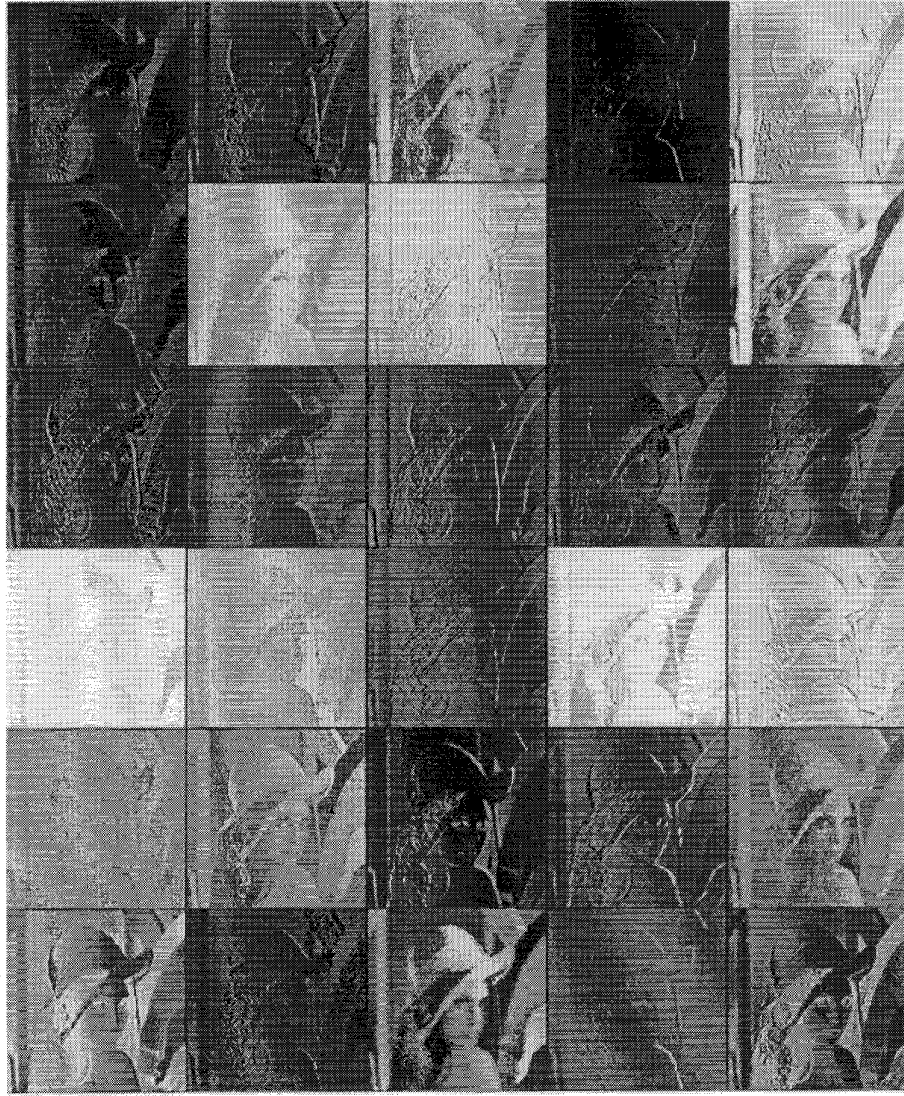


Fig. 10. Output of hidden layer of an MLP predictor with 30 hidden units. The contrast of this image has been increased to display all feature extractors present in the MLP predictor.

If  $D_{\text{test}}^{\tau-1} - D_{\text{test}}^{\tau} > 0$ , go to next STEP; otherwise, STOP.

- Step 7: If  $\tau < \tau_{\text{max}}$ , go to Step 1; otherwise, STOP.

Note that in the updating equations in Steps 3 and 4,  $\mu$  is a constant defined as a number between zero and one that determines the effect of the past weight changes on the current direction of movement in the weight space. The term  $\mu(u_{jk}^t - u_{jk}^{t-1})$  in Step 3 (and the similar term in Step 4) is called the momentum term [13]; this term improves the rate of convergence and smooths the weight changes.

#### B. Functional Link Network

A FL network is a two-layer neural network with an artificially augmented input representation. The augmented input data incorporates higher-order effects and increases the dimension of the input space so that the FL network can handle the linearly nonseparable tasks using a linear

network [14]. The FL network is trained by using the original, as well as additional higher-order, input terms. The simple least-square learning rule is used, since it is a two-layer network. The block diagram of an FL network used for designing the predictor is shown in Fig. 3. The extended input data consists of  $I$  (original) and  $J$  (higher-order) input terms. The higher-order input terms are generated from the original input terms in an independent manner. In this way, the input representation is enhanced; however, no new information is included.

The major task in the design of an FL vector predictor is to find a suitably enhanced representation of the input data. In the FL vector predictor, additional input terms are generated as the union of the product terms  $a_i \times a_j$  for all  $i \neq j$ . Using this model for an  $n$ -dimensional input space, the total number of additional input terms is given by  $\frac{n(n-1)}{2}$ . Obviously, the number of additional input terms





Fig. 11. Portion of image "Man" (magnification: two times): (a) original image, (b) predicted image by linear predictor, (c) predicted image by MLP predictor with 20 hidden units, and (d) predicted image by MLP predictor with 30 hidden units.

grows very quickly with an increasing value of  $n$ . For example, for  $n = 64$ , a total of 2016 additional input terms can be used. However, this would make the FL vector predictor impractical for implementation. Therefore, the number of higher order terms must be reduced significantly in order to design a practical FL vector predictor. This is achieved by using only the higher-order terms that are generated by the pixels in the original input space that are relatively close to the pixels in the current block to be predicted.

The higher-order input terms are obtained as the product of two pixels (Fig. 1), where one pixel comes from the closest pixel column/row to the predicted block, and the other

pixel comes from the three closest pixel columns/rows. The construction rule is defined as follows

$$p(x, 4) \times p(x, y) \quad 2 \leq x \leq 8, \quad 2 \leq y \leq 3 \quad (11)$$

$$p(x, 4) \times p(z, 4) \quad 2 \leq x, \quad z \leq 8, \quad x \neq z \quad (12)$$

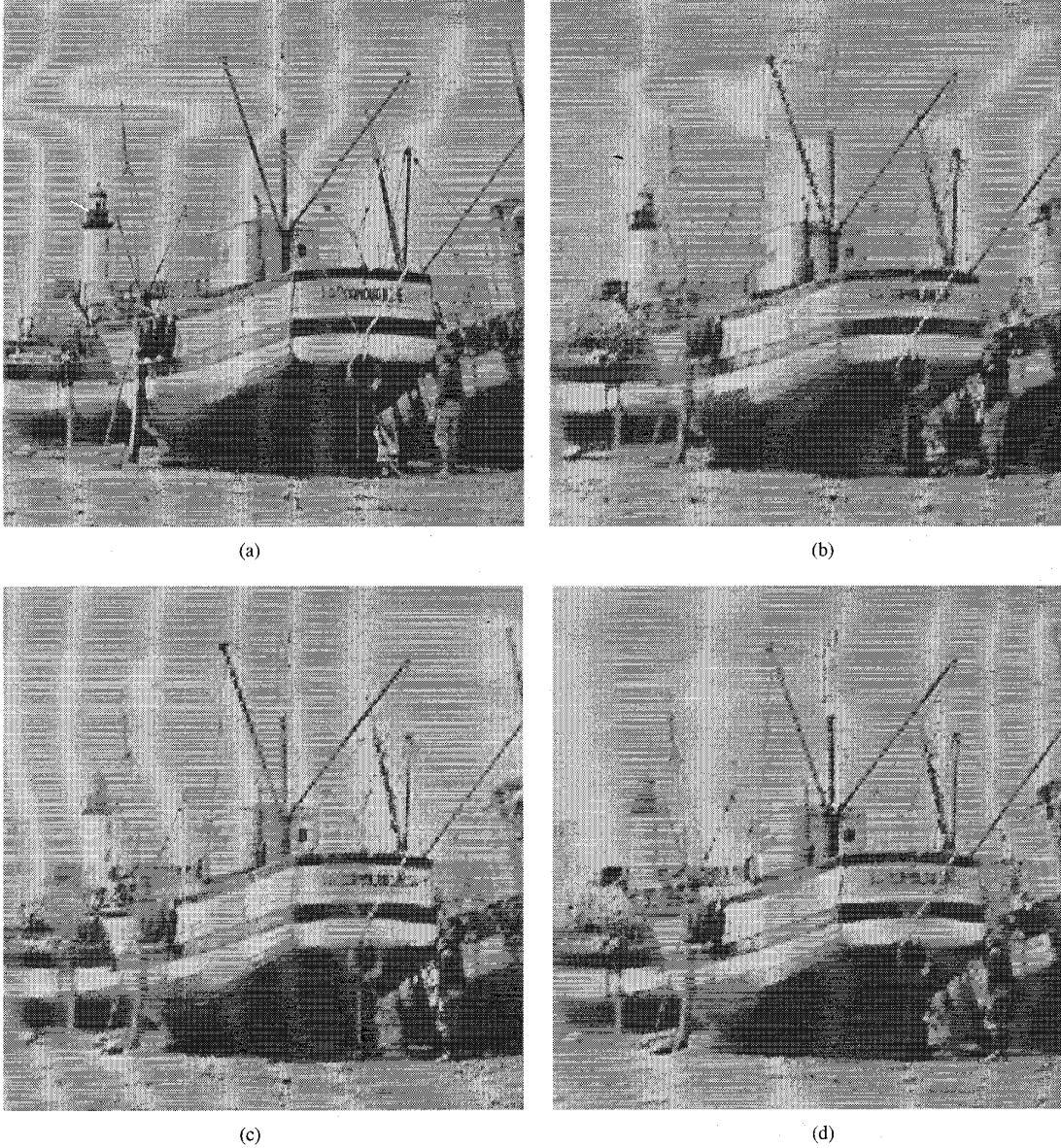
$$p(4, y) \times p(x, y) \quad 2 \leq y \leq 11, \quad 2 \leq x \leq 3 \quad (13)$$

$$p(4, y) \times p(4, z) \quad 2 \leq y, \quad z \leq 11, \quad y \neq z \quad (14)$$

where  $p(x, y)$  represents a pixel at location  $(x, y)$  (see Fig. 1). Now, we present an algorithm for designing the vector predictor by using the FL network.

*Algorithm 2:* Given the training set  $S = [\mathbf{X}_i; i = 1, 2, \dots, M]$  and the test set  $T = [\mathbf{X}_i; i = 1, 2, \dots, N]$ , set





**Fig. 12.** “Boats.” (a) Original image, and predicted test image using (b) linear predictor, (c) FL predictor, and (d) RBF predictor.

the initial weights associated with the output layer of the FL network to small random values. Select the learning rate  $\eta$  and the maximum number of training epochs  $\tau_{\max}$ . Choose an initial distortion  $D_{\text{test}}^0$  for the test image to be a large number. Set the number of epochs  $\tau = 0$ .

- Step 1: Set  $l \rightarrow 1$ .
- Step 2: Compute the predicted vector  $\tilde{\mathbf{X}}_l = [\tilde{x}_k; k = 1, 2, \dots, K]$  (based on the four neighboring blocks shown in Fig. 1) as

$$x_k = \sum_i a_i u_{ik} + \sum_j b_j u_{jk}.$$

- Step 3: Update the weights  $u_{ik}$  and  $u_{jk}$  as

$$u_{ik}^{t+1} = u_{ik}^t + \eta(x_k - \tilde{x}_k)a_i$$

and

$$u_{jk}^{t+1} = u_{jk}^t + \eta(x_k - \tilde{x}_k)b_j$$

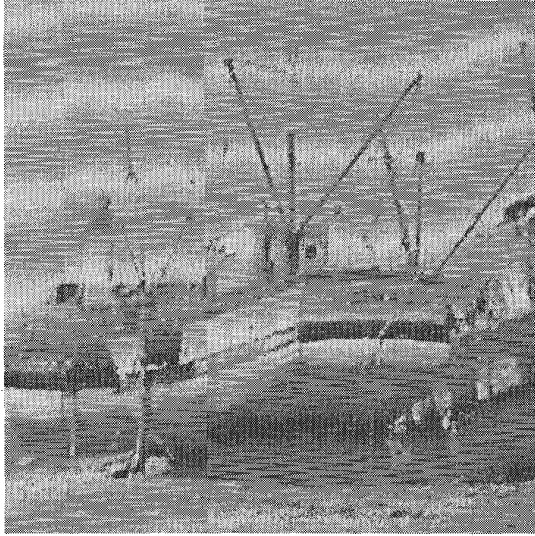
where  $t$  is the time index.

- Step 4: Set  $l \rightarrow l + 1$ . If  $l < M$ , go to STEP 2; otherwise, go to next STEP.
- Step 5: Set  $\tau \rightarrow \tau + 1$ . Compute the average distortion  $D_{\text{test}}^\tau$  for the predicted test image as

$$D_{\text{test}}^\tau = \frac{1}{K \times N} \sum_i \|\mathbf{X}_i - \tilde{\mathbf{X}}_i\|^2$$

if  $D_{\text{test}}^{\tau-1} - D_{\text{test}}^\tau > 0$ , go to next STEP; otherwise, STOP.

- Step 6: If  $\tau < \tau_{\max}$  go to Step 1; otherwise, STOP.



(e)

Fig. 12. (Continued.) (e) Predicted test image "Boats" using MLP predictor.

We should point out that the outputs of the FL predictor do not use sigmoid function. Furthermore, an FL predictor reduces to a linear predictor if the higher-order input terms are not used.

### C. The RBF Network

The structure of a RBF network with  $I$  inputs,  $J$  basis functions, and  $K$  outputs is shown in Fig. 4, where  $I = 64$ ,  $J = 30$ , and  $K = 16$ . We used the generalized Gaussian RBF network to design a vector predictor, instead of the regularization RBF network. The generalized Gaussian RBF network allows the use of a reduced number of hidden neurons; however, it gives a suboptimal solution. The network implements the function

$$F_k(A) = \sum_{j=1}^J u_{jk} \varphi_j(A) \quad (15)$$

for the  $k$ th output neuron, where  $A = [a_i; i = 1, 2, \dots, I]$  represents the  $I$ -dimensional input vector and  $u_{jk}$  represents the linear weight associated with the connection from the  $j$ th neuron in the hidden layer to the  $k$ th neuron in the output layer.  $\varphi_j(A)$  is the basis function associated with the  $j$ th neuron in the hidden layer [15]. In a Gaussian basis function,  $\varphi_j(A)$  is implemented by

$$\begin{aligned} \varphi_j(A) &= G_A(\mu_j, \sigma_j) \\ &= \frac{1}{(2\pi)^{I/2} |\sigma_j|^{1/2}} \\ &\quad \times \exp \left[ -\frac{1}{2} (A - \mu_j)^T \sigma_j^{-1} (A - \mu_j) \right] \end{aligned} \quad (16)$$

where  $T$  denotes the transpose and  $\mu_j$  and  $\sigma_j$  are the  $I$ -dimensional vectors representing the centers and the spreads, respectively, of the RBF  $\varphi_j(\cdot)$ .

The learning process of an RBF network is supervised and consists of adjusting the three free parameters: 1) the linear weights, 2) the centers, and 3) the spreads. The cost function of the  $k$ th output neuron for the training pattern  $A$  is defined as

$$\varepsilon_k = \frac{1}{2} e_k^2 \quad (17)$$

where

$$e_k = d_k - F_k(A) = d_k - \sum_{j=1}^J u_{jk} G_A(\mu_j, \sigma_j) \quad (18)$$

where  $d_k$  and  $F_k(A)$  are the desired and the computed output of the  $k$ th output neuron for training pattern  $A$ , respectively. In the experiments performed in this paper, the initial values for these free parameters are generated randomly. The adaptation formulas are summarized below

$$u_{jk}^{t+1} = u_{jk}^t + \eta_u e_k^t G_A(\mu_j^t, \sigma_j^t) \quad (19)$$

$$\mu_j^{t+1} = \mu_j^t - \eta_\mu 2b_j^t \frac{1}{\sigma_j^t} [A - \mu_j^t] G_A(\mu_j^t, \sigma_j^t) \quad (20)$$

$$\frac{1}{\sigma_j^{t+1}} = \frac{1}{\sigma_j^t} + \eta_\sigma b_j^t [A - \mu_j^t]^2 G_A(\mu_j^t, \sigma_j^t) \quad (21)$$

$$b_j^t = \sum_{k=1}^K e_k^t u_{jk}^t \quad (22)$$

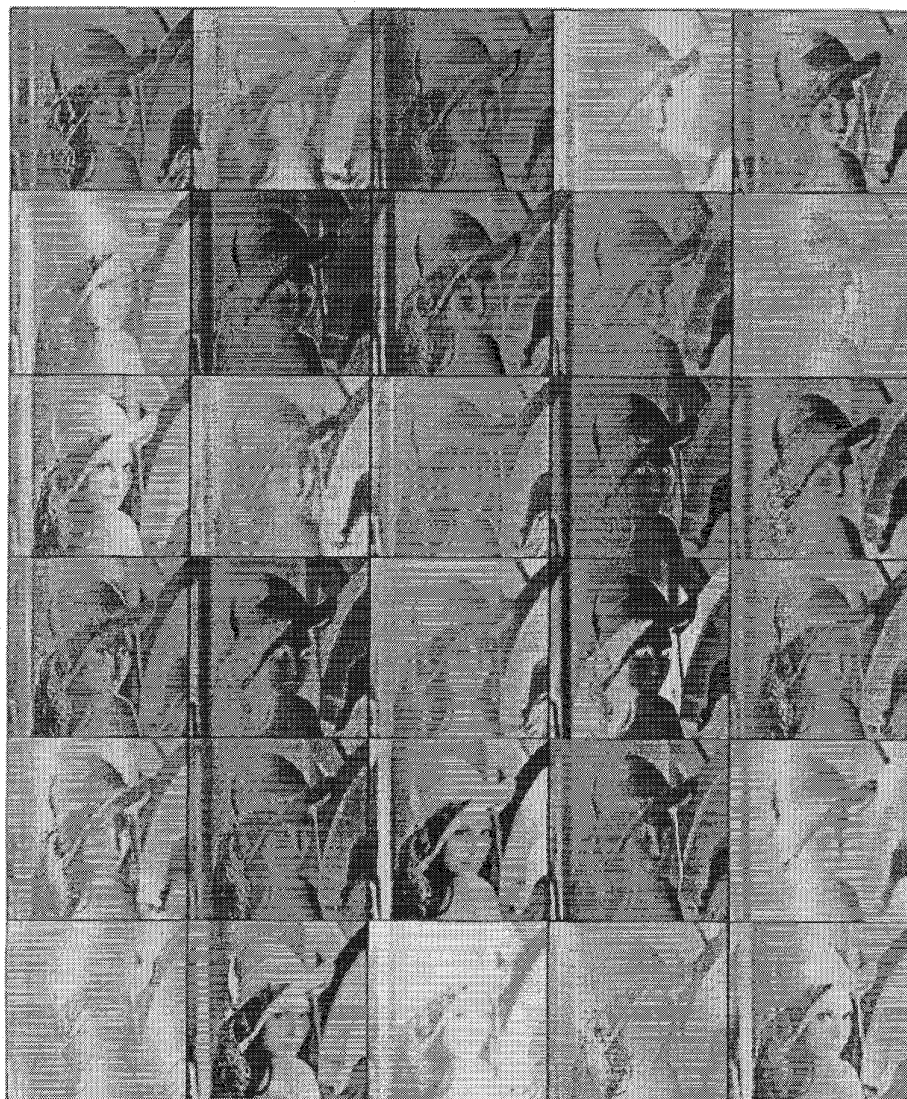
where  $\eta_u$  is the learning rate for the weights associated with the output layer of the RBF predictor, and  $\eta_\mu$  and  $\eta_\sigma$  are the learning rates for the centers and the spreads of the RBF, respectively. We now present the algorithm for designing an RBF vector predictor.

*Algorithm 3:* Given the training set  $S = [\mathbf{X}_l; l = 1, 2, \dots, M]$  and the test set  $T = [\mathbf{X}_i; i = 1, 2, \dots, N]$ , set the initial weights associated with the output neurons and the initial centers and spreads of the Gaussian RBF's to small random values. Select the appropriate learning rates  $\eta_u$ ,  $\eta_\mu$ ,  $\eta_\sigma$  and the maximum number of training epochs  $\tau_{\max}$ . Choose the initial distortion  $D_{\text{test}}^0$  for the test image to be a large number. Set the number of epochs  $\tau = 0$ .

- Step 1: Set  $l \rightarrow 1$ .
- Step 2: Compute the predicted vector  $\tilde{\mathbf{X}}_l = [\tilde{x}_k; k = 1, 2, \dots, K]$  (based on the four neighboring blocks shown in Fig. 1) using (15) and (16).
- Step 3: Update the linear weights associated with the output layer of the RBF network using (19).
- Step 4: Update the centers and spreads of the Gaussian RBF's using (20)–(22).
- Step 5: Set  $l \rightarrow l + 1$ . If  $l < M$ , go to STEP 2; otherwise go to next STEP.
- Step 6: Set  $\tau \rightarrow \tau + 1$ .

Compute the average distortion  $D_{\text{test}}^\tau$  for the predicted test image as

$$D_{\text{test}}^\tau = \frac{1}{K \times N} \sum_i^N \|\mathbf{X}_i - \tilde{\mathbf{X}}_i\|^2.$$



**Fig. 13.** Output of hidden layer of RBF predictor with 30 hidden units. (The contrast of this image has been increased to display all the feature extractors present in the RBF predictor.)

**Table 1** Performance of Neural Network and Linear Predictors on Training and Test Data in Terms of Average PSNR and Average Error Entropy

Predictor	Training Data		Test Data	
	Average PSNR (dB)	Average Error Entropy (bpp)	Average PSNR (dB)	Average Error Entropy (bpp)
MLP	27.79	4.792	26.53	5.060
RBF	27.76	4.725	26.53	5.010
FL	27.50	4.731	26.36	5.004
Linear1	27.39	4.734	26.30	5.003
Linear2	27.36	4.736	27.27	5.000

If  $D_{\text{test}}^{\tau-1} - D_{\text{test}}^{\tau} > 0$ , go to next STEP; otherwise, STOP.

- Step 7: If  $\tau < \tau_{\text{max}}$ , go to Step 1; otherwise, STOP.

#### IV. EXPERIMENTAL RESULTS

Computer simulations were performed to implement the vector predictors proposed in this paper. We used five images of resolution  $512 \times 512$  pixels with eight bits per

pixel (bpp) as our training set. We used the image “Lena” (outside the training set) as the test image for stopping the training process. We used a block size of  $4 \times 4$ , with a total number of 80 010 training vectors, in all of the experimentation reported in this paper, unless otherwise stated. We also designed two linear vector predictors, one using the LMS method presented in [16], and the other using the covariance method presented in [1]. We also

**Table 2** Performance of Neural Network and Linear Predictors on Several Training and Test Images

Image	MLP PSNR (dB)	RBF PSNR (dB)	FL PSNR (dB)	Linear1 PSNR (dB)	Linear2 PSNR (dB)
Crowd*	24.92	24.81	24.74	24.64	24.58
Man*	25.49	25.34	24.95	24.84	24.83
Lena**	27.27	27.21	26.93	26.89	26.85
Marie**	28.62	28.69	28.37	28.29	28.21
Couple**	23.71	23.70	23.77	23.72	23.74

\*Inside training data

\*\*Outside training data

implemented a VQ-based nonlinear predictor developed in [7]. In all of the vector predictors presented in this paper (including the linear vector predictors), the prediction is based on the four *past* blocks shown in Fig. 1. Therefore, the number of input elements in all of the vector predictors, except the FL vector predictor, is 64 (65 input elements, when the biased input is included). In the case of the FL vector predictor, higher-order input terms are needed and, therefore, the number of inputs to an FL vector predictor is relatively large (64 original terms, one bias term, and 364 higher-order terms). The MSE calculation is used as a measure of performance for the training procedure. We define

$$\text{MSE} = \frac{1}{K \times L} \left( \sum_{l=1}^L (\|X_l - \hat{X}_l\|^2) \right) \quad (23)$$

where  $L$  refers to the number of input vectors in the test image and  $K$  is the vector dimension.

Figs. 5–7 show the training profile of the neural network predictors. As depicted in Figs. 5–7, the prediction error (MSE) is decreased at the beginning of the training for both the training data and the test data. At a certain point in the training, the prediction error begins to increase for the test image; however, it continues to decrease for the training data; we refer to this point as the *saturation* point. Further training of the neural network beyond this saturation point overtunes the network in the direction of the training data. The performance of the neural network on the test data deteriorates if training is continued after this saturation point; therefore, the training procedure for the network is terminated when this saturation point is reached. In the experiment performed in this paper, we observed that the RBF predictor reaches its saturation point after a large number of training epochs (e.g., about 11 000 epochs). An epoch consists of one pass of the whole training data through the network; the MLP and FL predictors reach their saturation points at 1229 and 1360 epochs, respectively. However, after a few hundred epochs, there is very little decrease in the error for both the training and test data. This suggests that the RBF predictor has a relatively high resilience to overtune in the direction of the training data.

The vector predictor based on the MLP has one hidden layer with 31 neurons (including the biased neuron). The number of hidden layers and the number of neurons in each hidden layer is determined experimentally. There is a trade-

off between the computational complexity, the training time, and the performance gain. In the MLP vector predictor that we presented in this paper, only one hidden layer is used to reduce the computational complexity and the training time.

Fig. 8 shows the MSE versus the number of epochs for the MLP predictor, with different numbers of hidden neurons. As depicted in the figure, the performance of the predictor improves significantly as the number of hidden neurons is increased from ten to 20 neurons. At this point, adding more hidden neurons to the network results in a slight improvement in performance. This behavior of the MLP predictor can be explained by considering Figs. 9 and 10. These figures show the output at the hidden layer of a three-layer MLP predictor, where the hidden layers contain 20 and 30 neurons, respectively. As shown in the figures, each of the hidden neurons acts as a feature extractor, extracting some specific features from the input data. These features are passed to the output layer, which constructs the predicted image. The exact nature of these feature extractors are not known; however, Figs. 9 and 10 show that the most obvious operations performed by the hidden neurons are low-pass filtering, high-pass filtering, and directional edge detection. This could explain the superior prediction of a block containing edges by a neural network predictor.

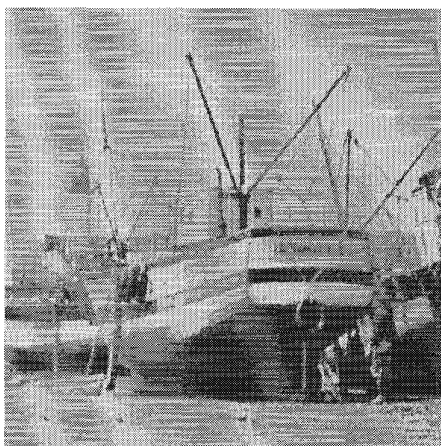
Fig. 11 shows a magnified view of the shoulder of the test image “Man” predicted by a linear predictor, an MLP predictor with 20 hidden neurons, and an MLP predictor with 30 hidden neurons, respectively. A comparison of Figs. 9 and 10 shows that an MLP predictor with 20 hidden neurons has *sufficient* feature extractors for a satisfactory prediction. Adding more hidden neurons at this point results in an increase in feature extractors, where some are duplicated. For example, two more low-pass filters appear in Fig. 10 than in Fig. 9. The additional low-pass filters are responsible for the smoothing of the background and the shade areas in the predicted image. There is also an increase in edge operators (with some duplications). These redundant edge operators contribute very little to the further improvement of the quality of the blocks containing edges.

Fig. 12 shows the original and the predicted test image “Boats” (outside the training process) using linear, FL, RBF, and MLP predictors, respectively. As shown in Fig. 12, the MLP vector predictor predicts the edges with better accuracy than the other vector predictors. The linear predictor estimates the background and the shade areas satisfactorily; however, it performs rather poorly in predicting the edges. The FL predictor gives a slight improvement over the linear predictor. (As mentioned earlier, an FL predictor reduces to a linear predictor if no higher-order terms are used.) In the image predicted by the FL predictor, the quality of the background areas is similar to that of the image predicted by the linear predictor. The effect of higher-order terms in the FL predictor can be seen in the form of slightly better prediction of the edge

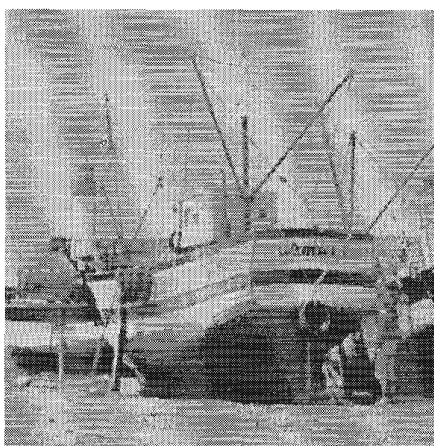


**Table 3** The Performance of PVQ Using Four Different Vector Predictors

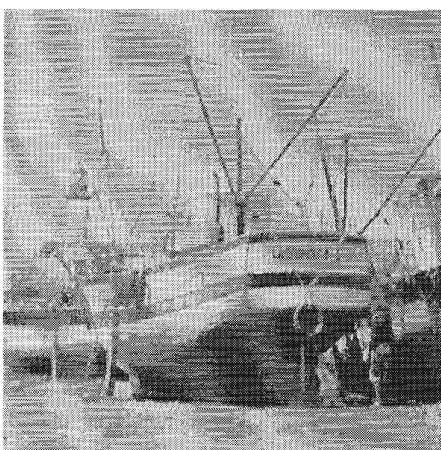
Image	Codebook Size	MLP PSNR (dB)	RBF PSNR (dB)	FL PSNR (dB)	Linear PSNR (dB)
Man	64	29.19	29.12	29.14	29.15
	128	29.88	29.90	29.82	29.81
	256	30.63	30.58	30.45	30.50
Lena	64	31.40	31.31	31.28	31.35
	128	32.20	32.14	32.08	32.06
	256	33.02	32.87	32.73	32.84
Boat	64	29.25	29.11	29.16	29.16
	128	29.96	29.90	29.83	29.85
	256	30.65	30.53	30.46	30.59



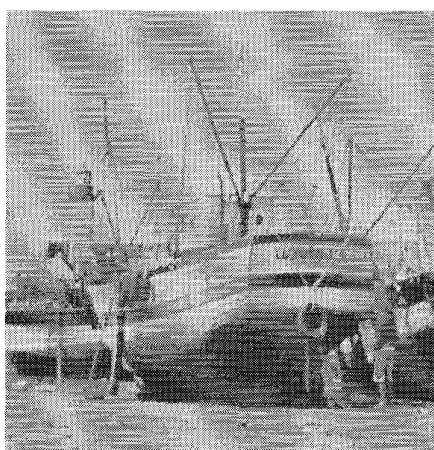
(a)



(b)



(c)



(d)

**Fig. 14.** Encoded image boats.

blocks. However, overall there is no significant difference in the images predicted by the linear and the FL predictors, which could be attributed to the limited number of higher-order terms used in our FL predictor. An FL predictor with a large number of higher-order input terms would be of little practical significance because of its computational complexity.

The performance of the RBF predictor in Fig. 12(d) suggests that the RBF predictor is a compromise between the linear and the MLP predictors. This assertion is further supported by observing the output of the hidden neurons of

the RBF predictor. Fig. 13 shows that most of the hidden neurons act as low-pass filters; however, there are a few high-pass or bandpass feature extractors, when compared with the MLP predictor. In the image predicted by the RBF predictor, the background and the shade areas have good visual quality. The quality of the edge areas in the predicted image, on the other hand, is not as good as in the image predicted by the MLP predictor. This behavior of the RBF predictor can be attributed to the fact that the RBF predictor performs a combination of linear and nonlinear processing. This accounts for better prediction of

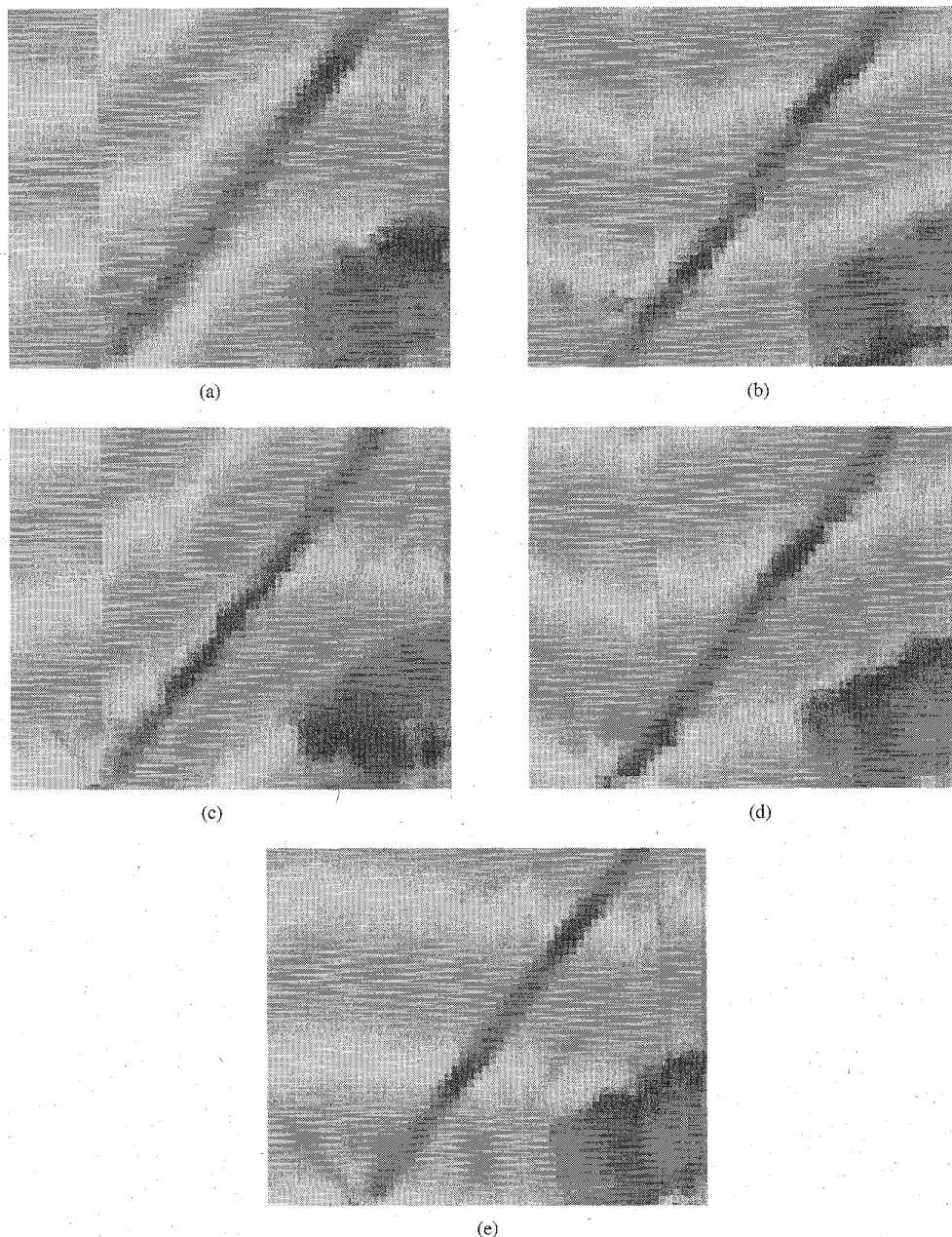


Fig. 15. Portion of encoded image boats.

the background, as well as the edge areas, compared to that of the linear predictor.

We also implemented the nonlinear, VQ-based predictor developed in [7] for comparison with our neural network predictors. Unfortunately, this theoretically sound approach involves a substantial computational complexity for implementing a fourth-order vector predictor with a block size of  $4 \times 4$  that was implemented in this paper. However, our experimental results show that the VQ-based nonlinear vector predictor could lead to an optimal nonlinear predictor, if computational complexity is not a constraint. Specifically, we designed the VQ-based predictor with codebook sizes 64, 256, and 512. The performance of these predictors was

23.09 dB, 23.73 dB, and 23.99 dB, respectively, for the test image "Lena" (outside the training data). This shows about 1 dB of improvement in the predicted image when the codebook size is increased from 64 to 512. However, the predictor with a codebook size 512 took several minutes on a SUN SparcStation10 to predict a  $512 \times 512$  image, so we did not pursue the design of the VQ-based predictor further. Accordingly, the results for a VQ-based nonlinear predictor are not included in the comparison.

Table 1 presents the performance comparison of the neural network predictors presented in this paper, along with the two linear predictors designed in [1] and [16]. The comparison is given in terms of the peak signal-to-noise

ratio (PSNR) given by

$$\text{PSNR} = 10 \log \frac{255^2}{\text{MSE}} \quad (24)$$

averaged over the training and test data. Table 1 also shows the average error entropy for the training and test data. In the experiments performed in this paper, the predictor "Linear1" was designed by using the covariance method presented in [1], which is the *optimal* linear predictor for the training data. The predictor "Linear2" was designed using the LMS method [16], which could be regarded as a *suboptimal* or *nearly optimal* linear predictor for the training data. The superior performance of the neural network predictors over the *optimal* linear predictor implies that the neural network predictors *do approximate* the *optimal* nonlinear predictor. Due to the lack of a precise model for the source data, the linear and neural network predictors attempt to achieve optimality for the training data alone. They could not be regarded as optimal for test data that is outside of the training data set. In most cases, however, the training data is chosen as a good representation of the actual test data on which the predictor will operate; therefore, it is expected that the predictors will generalize and perform reasonably well for all of the test data. Table 1 shows that all of the neural network predictors perform better than the linear predictors.

Table 2 shows the performance of the neural network and linear predictors on several images selected from inside, as well as outside, the training data. In Table 2, the images "Crowd" and "Man" were taken from inside the training data. As we mentioned earlier, the test image "Lena" was not used for training; however, it was used as the test image to stop the training process. We also used two additional test images: "Marie" and "Couple." These images were *completely* outside the training process. Test image "Marie" has similar statistics to those images used in the training process; however, the test image "Couple" has quite different statistics from the training data. Table 2 shows that the neural network predictors performed better than the linear predictor for the test image "Marie"; whereas the performance of all the predictors was almost the same for the test image "Couple." This implies that the neural network predictors perform well for the data which has statistics similar to the training data. Therefore, this property of the neural network predictors makes them attractive for designing dedicated predictors for different kinds of data, forming an adaptive system in order to achieve better overall performance.

In this paper, we also present an application of these vector predictors on image coding. We performed computer simulation to evaluate performance of the vector predictors proposed in this paper to a PVQ coding scheme. We designed several PVQ's with different vector predictors using codebooks sizes 64, 128, and 256. Accordingly, the fixed rates for these PVQ's are 0.375 bpp, 0.4375 bpp, and 0.5 bpp, respectively. We also used the five images that we used to train the vector predictors for designing the

codebooks. Table 3 shows the performance comparison of PVQ's with different vector predictors. In terms of PSNR, the PVQ with MLP vector predictor had slightly better performance. Fig. 14 shows the image "Boats" encoded by four PVQ's with different vector predictors, and a codebook size 128. This figure shows that the overall quality of the four encoded images is almost the same; all of them have nice background and edge preservation. However, on some critical edge portions (such as the poles on image "Boats"), the PVQ's with MLP and RBF vector predictors create better quality. The pole portion of the test image "Boats" encoded by a PVQ with a linear vector predictor has blocking artifacts and some white spots around the poles. Finally, a magnified view of the pole region of the test image "Boats," encoded by the PVQ's with different vector predictors, is shown in Fig. 15.

## V. CONCLUSION

In this paper, we presented several neural network architectures for designing the nonlinear predictors. We also presented the algorithms for designing the neural network predictors, based on the MLP, the RBF network, and the FL network. The neural network approach provides a solution to the problem of designing a nonlinear vector predictor. The neural network predictors are found to have improved performance, in comparison to that of an optimal linear predictor, both numerically and perceptually. We demonstrated that the neural network predictors predict the blocks that contain edges with greater accuracy than a linear predictor. This capability of the neural networks can be better taken advantage of by using a modular approach in which a dedicated predictor is used to learn particular statistics of the data. The use of neural network predictors as expert modules in an adaptive vector prediction scheme is the focus of the future research.

## REFERENCES

- [1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston: Kluwer, 1992.
- [2] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. 36, pp. 957-971, Aug. 1988.
- [3] H. M. Hang and J. W. Woods, "Predictive vector quantization of images," *IEEE Trans. Commun.*, vol. COM-33, pp. 1208-1219, Nov. 1985.
- [4] N. Mohsenian, S. A. Rizvi, and N. M. Nasrabadi, "Predictive vector quantization using a neural network approach," *Opt. Engineering*, vol. 32, no. 7, pp. 1503-1513, July 1993.
- [5] S. A. Rizvi and N. M. Nasrabadi, "Predictive vector quantization using constrained optimization," *IEEE Signal Process. Lett.*, vol. 1, no. 1, pp. 15-18, Jan. 1994.
- [6] —, "Predictive residual vector quantization," *IEEE Trans. Image Process.*, vol. 4, pp. 1482-1495, Nov. 1995.
- [7] S. Wang, E. Paksoy, and A. Gersho, "Nonlinear prediction of speech with vector quantization," in *Proc. Int. Conf. Spoken Language Process.*, Kobe, Japan, Nov. 1990, pp. 29-32.
- [8] N. Tishby, "A dynamical systems approach to speech processing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Albuquerque, NM, Apr. 1990, pp. 365-368.
- [9] S. Dianat, N. M. Nasrabadi, and S. Venkataraman, "A nonlinear predictor for DPCM encoder using an artificial neural network," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Toronto, Canada, 1991.



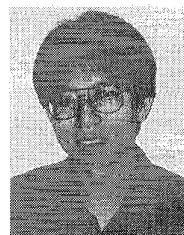
- [10] C. N. Manikopoulos, "Neural network approach to DPCM system design for image coding," *IEE Proc.—I*, vol. 139, no. 5, pp. 501–507, Oct. 1992.
- [11] H. White, "Learning in artificial neural networks: A statistical perspective," *Neur. Computation*, vol. 1, pp. 425–464, 1989.
- [12] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA: MIT Press, 1986.
- [13] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neur. Networks*, vol. 1, no. 4, pp. 295–307, 1988.
- [14] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, MN: West, 1992.
- [15] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
- [16] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.



**Syed A. Rizvi** (Member, IEEE) received the B.Sc. degree (honors) from University of Engineering and Technology, Lahore, Pakistan, and the M.S. and Ph.D. degrees from the State University of New York, Buffalo, in 1990, 1993, and 1996, respectively, all in electrical engineering.

From 1992 to 1995, he was a Teaching Assistant at SUNY Buffalo. Since 1995, he has been a Research Associate with the U.S. Army Research Laboratory, Adelphi, MD, where his responsibilities include development of coding algorithms for FLIR images and video using wavelet decomposition and vector quantization; and application of neural networks to FLIR automatic target recognition. His research interests include image and video coding, applications of artificial neural networks in image processing, and computer vision.

Dr. Rizvi is a member of SPIE.



**Lin-Cheng Wang** (Student Member, IEEE) received the B.S. degree from National Sun Yat-Sen University, Taiwan, in 1985, and the M.S. degree from National Taiwan University, Taiwan, in 1987, both in electrical engineering.

From 1987 to 1989, he was an Officer Lecturer in the Chinese Naval Academy. From 1989 to 1993, he was with United Microelectronics Corp., Taiwan, where he worked as a VLSI designer. He is currently a Ph.D. student in the Department of Electrical and Computer Engineering, State University of New York at Buffalo. Since 1995, he has been working with U.S. Army Research Laboratory, Adelphi, MD, as a Research Associate. His current research interests are the applications of neural networks in FLIR automatic target recognition and in image coding.



**Nasser M. Nasrabadi** (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in electrical engineering from Imperial College of Science and Technology (University of London), London, U.K., in 1980 and 1984, respectively.

From 1986 to 1991 he was an Assistant Professor in the Department of Electrical Engineering at Worcester Polytechnic Institute, Worcester, MA. Since 1994 he has been a Visiting Senior Research Scientist with the U.S. Army Research Laboratory working on image compression and automatic target recognition. Since 1991, he has also been an Associate Professor in the Electrical and Computer Engineering Department at State University of New York at Buffalo. His current research interests are in image and video compression, packet video, automatic target recognition, and neural networks applications to image processing. He served as an Associate Editor for the *IEEE TRANSACTIONS ON IMAGE PROCESSING* from 1993 to 1996. He is currently an Associate Editor for the *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*.