Optimal Nonlinear Adaptive Prediction and Modeling of MPEG Video in ATM Networks Using Pipelined Recurrent Neural Networks

Po-Rong Chang, *Member, IEEE*, and Jen-Tsung Hu

***Abstract*—This paper investigates the application of a pipelined recurrent neural network (PRNN) to the adaptive traffic prediction of MPEG video signal via dynamic ATM networks. The traffic signal of each picture type (I, P, and B) of MPEG video is characterized by a general nonlinear autoregressive moving average (NARMA) process. Moreover, a minimum mean-squared error predictor based on the NARMA model is developed to provide the best prediction for the video traffic signal. However, the explicit functional expression of the best mean-squared error predictor is actually unknown. To tackle this difficulty, a PRNN that consists of a number of simpler small-scale recurrent neural network (RNN) modules with less computational complexity is conducted to introduce the best nonlinear approximation capability into the minimun mean-squared error predictor model in order to accurately predict the future behavior of MPEG video traffic in a relatively short time period based on adaptive learning for each module from previous measurement data, in order to provide faster and more accurate control action to avoid the effects of excessive load situation. Since those modules of PRNN can be performed simultanously in a pipelined parallelism fashion, this would lead to a significant improvement in the total computational efficiency of PRNN. In order to further improve the convergence performance of the adaptive algorithm for PRNN, a learning-rate annealing schedule is proposed to accelerate the adaptive learning process. Another advantage of the PRNN-based predictor is its generalization from learning that is useful for learning a dynamic environment for MPEG video traffic prediction in ATM networks where observations may be incomplete, delayed, or partially available. The PRNN-based predictor presented in this paper is shown to be promising and practically feasible in obtaining the best adaptive prediction of real-time MPEG video traffic.**

***Index Terms*—MPEG, nonlinear autoregressive moving average (NARMA), pipelined recurrent neural network (PRNN).**

I. Introduction

FUTURE broad-band integrated services digital netorks (B-ISDN's) based on the asynchronous transfer mode (ATM) principle are designed to support a wide variety of multimedia services with diverse statistical characteristics and quality of service (QOS) requirements at cell and call levels. Among the various kinds of services, video service is becoming an important component of multimedia communications. A number of video coding schemes have been proposed for the compression of video signals that generate a high bit-rate stream via the ATM broad-band networks. A well-known coding scheme called the Motion Picture Expert Group (MPEG) is emerging as a major international standard for multimedia applications [1]. The MPEG specification was developed specifically to allow the transmission of VCR-quality digital images at a data rate of approximately 1–1.5 Mbits/s. IS01172 is also sometimes referred to as MPEG-1. Another specification known as MPEG-2 has been developed to support higher resolution images. The MPEG video compression algorithm relies on two basic techniques: block-based motion compensation for reduction in temporal redundancy and discrete cosine transform (DCT) for spatial redundancy. MPEG is able to reduce the raw image data by 20- to 50-fold. Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intrapictures (I pictures) are coded without reference to other pictures and points for random access. Predictive (P) pictures are coded with reference to previous I or P frames. Bidirectionally predictive (B) pictures are coded with reference to a previous I or P frame, as well as the future I or P frame. As a result, the MPEG encode produces a variable bit-rate (VBR) compressed video bit stream in which instantaneous bit rates vary widely with scene content. Moreover, the correlation between consecutive pictures is very low since MPEG video generates very bursty traffic that changes dynamically on a picture-by-picture basis depending on coding mode. The ATM technology utilizes the nature of the traffic to effectively allocate the network resources via statistical multiplexing. Although statistical multiplexing provides efficient use of the network resource (e.g., bandwidth) and enough flexibility to support multiple connections with different bit rates, the effective statistical multiplexing requires detailed information about the traffic of MPEG video. This information is used to design traffic smoothing, bandwidth allocation, and congestion control algorithms [2]–[5]. Thus, video source modeling for traffic prediction is extremely important for high-speed packet switching and ATM network design. Since the channel capacity for each video source is allocated dynamically in these networks, source models are especially important for providing the prediction of the required channel capacity.

Source models based on the queueing theory under the assumption of random packet arrival were effective for earlier packet networks carrying data generated by computers. The

Manuscript received April 10, 1996; revised October 7, 1996. This work was supported in part by the National Science Council of Taiwan, R.O.C under Contract NSC 86-2221-E-009-056.

The authors are with the Department of Communication Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C.

Publisher Item Identifier S 0733-8716(97)04199-1.

random arrival assumption of MPEG video signals, however, may no longer be valid. In addition, another shortcoming of the queueing models is that only steady-state results are tractable. Therefore, two classes of video models have been proposed. A model based on discrete-state continuous-time Markov processes was proposed by Maglaris *et al*. [6] for analytical use in video sources with relatively uniform activity level (no scene changes). This model was extended by Sen *et al*. [7] to encompass scene changes. However, the parameters to fit the model to general video sources is difficult. Another model based on autoregressive (AR) processes was proposed by Nomura *et al*. [8] and Ohta [9] to simulate the characteristics of a single video source, which can capture the autocorrelation property of the video signal. A number of researchers [10], [11] have shown that the AR model closely matches an actual video signal because coefficients of the AR model can easily be derived from the characteristics of real video data. However, the AR model is not suitable for the video signal with a very bursty traffic like MPEG video and is also not appropriate for the statistical multiplexing of VBR video signals via an ATM network. To tackle this difficulty, Wu *et al*. [12] have proposed a model for the simulation of MPEG video signals, which is based on the AR model compensated by a projection function. Another method of characterizing the VBR video signals via ATM networks was proposed by Grunenfelder *et al*. [13]. This model is based on an autoregressive moving average (ARMA) process followed by a zero-memory nonlinearity and provides satisfactory modeling accuracy for the VBR video signals.

The above-mentioned modeling methods for characterizing the VBR video signals are used for off-line applications to both the simulation and analysis. However, many flow control mechanisms that dynamically regulate traffic flows according to changing network conditions require the capability of predicting future behavior of the video traffic in a relatively short time period (sampling period) in order to provide faster and more accurate control actions to avoid the effects of excessive load situation. Jung and Meditch [2] have applied the AR model to the prediction for each picture type of real MPEG video signals. An alternative method based on neural networks has been proposed to further improve the prediction accuracy of the traffic statistics of the VBR video. Several examples of using neural network approaches for on-line traffic prediction of VBR video signals can be found in the literature [5], [14], [15]. Neves *et al*. [4], [5] and Tarraf and Habib [14] applied the multilayered perceptron neural networks with backpropagation training to the traffic prediction in the flow control and traffic enforcement mechanism for ATM networks, respectively. Recently, Chong *et al*. [16] used the high-order pi–sigma neural network for the bandwidth prediction of VBR video over an ATM network. All of them have shown that satisfactory traffic prediction accuracy can be achieved by those neural networks. However, those neural networks suffer from drawbacks of slow convergence and unpredictable solutions during learning. To overcome this difficulty, an alternative architecture to the traffic prediction of MPEG video with the flexibility to adapt to a changing ATM network environment is based on recurrent neural networks (RNN's). An RNN is well suited for the adaptive prediction of a nonstationary time series [17]. Several algorithms have been proposed for the training of the RNN's. The most widely known algorithm is the real-time recurrent learning (RTRL) algorithm, proposed by Williams and Zipser [18], that can be used to update the synaptic weights of the RNN in real time.

In Section II, we will show the traffic for each picture type of MPEG video signal can be characterized by a general nonlinear autoregressive moving average (NARMA) process. According to the theroy of prediction [19], [20], the minimum mean-squared error traffic predictor is the conditional mean which can be expressed in terms of the functional expression of the NARMA process. However, the explicit functional expression of the NARMA model is actually unknown. Conner *et al*. [21] have introduced a recurrent neural network (RNN) implementation to approximate the NARMA-based conditional mean predictor, and have shown the superior accuracy of its prediction. Section III describes the procedure for the approximation of a NARMA-based optimal traffic predictor using the recurrent network. However, it is impossible to achieve the RNN-based prediction within an acceptably small measurement time interval for the purpose of adapting to the dynamic environment since a sufficiently large number of neurons are required to maintain the prediction accuracy, but also increase its computational complexity. To tackle this difficulty, in Section IV, a pipelined recurrent neural network (PRNN), proposed by Haykin and Li [22], is introduced to implement the NARMA-based optimal traffic predictor with low complexity. Moreover, we would like to apply the learning-rate annealing schedules [23] to the PRNN in order to improve its convergence performance further. Finallly, in Section V, we present an experimental study of the PRNN applied to the traffic prediction of a typical MPEG video signal.

II. Source Characteristics and Traffic Prediction Model of MPEG Video Signals

The MPEG video international standard [1] specifies the coded representation of the video data. The video source coding is based on motion-compensated hybrid DCT coding which employs two basic techniques: motion compensation for the reduction of temporal redundancy and DCT transform compression for the reduction of spatial redundancy. In order to achieve the highly efficient compression and to meet the conflicting requirements of random acess, the input video is divided into units of group-of-pictures (GOP's) consisting of an intra (I) picture, coded without reference to other pictures, an arrangement of predictive (P) pictures, coded with reference to previous (I or P) pictures, and bidirectionally predictive (B) pictures, coded with reference to an immediate previous (I or P) picture, as well as an immediate future (P or I) picture. The I picture at the beginning of a GOP serves as a basic entry point to facilitate random seek or channel switching, and also provides coding robustness to transmission error, but is coded with only moderate compression to reduce the spatial redundancies. P pictures are coded more efficiently using motion-compensated prediction from a past I or P picture, and are generally used as a reference for further prediction.

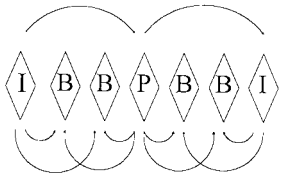


Fig. 1. Example of MPEG GOP.

B pictures provide the highest degree of compression, but require both past and future reference pictures for motion compensation. It should be mentioned that B pictures are never used as references for prediction. A GOP is defined by its length N_{GOP}, which is the distance between I pictures. An example of a GOP in MPEG is presented in Fig. 1, where I, P, and B denote picture encoding in the intrapicture mode, predicted mode, and bidirectionally predicted mode, respectively. The GOP sequence of Fig. 1 with $N_{\mathrm{GOP}} = 6$ is $IBBPBBI$.

The macroblocks of 16 $\times$ 16 pixels are the basic coding units for the MPEG algorithm. Each macroblock is divided into four blocks, where each block contains 8 $\times$ 8 pixels. The main extension from monochrome video to color is the addition of two 8 $\times$ 8 chrominace blocks to the macroblock. A row of macroblocks that makes up a horizontal strip in the image is called a slice, and a number of slices are combined to form a picture. The coding mode of each macroblock within a specific picture depends on its picture type. For I pictures, a discrete cosine transform (DCT) is performed on each block. The resulting two-dimensional block of DCT coefficients is quantized and scanned in a zig-zag order to convert it into a one-dimensional string of quantized DCT coefficients. Run-length coding is used for the quantized coefficient data. The predicted pictures (P and B) use motion-compensated prediction of the contents of the macroblock based on past or future reference pictures. This prediction is subtracted from the actual data in the current macroblock to form an error signal. The prediction error is coded like the intracoded macroblocks.

None of the analytical models available today can adequately represent VBR MPEG video traffic. Here, we choose a 5-s (150 picture) video traffic signal from a "Bike" scene of 352 $\times$ 240 pixels as a testbed for our study. It is coded by an MPEG compression technique. Usually, the main unit of measure for video traffic has been the number of bits generated per picture. In ATM network application, the bits generated by the VBR MPEG encoder are packetized into 53-byte ATM cells with a user payload size of 44 bytes plus 9 bytes of protocol overhead. Fig. 2 shows the MPEG coded Bike video traffic sequence measured in cells per picture, denoted by $s(n)$ at time instant n. Observing this figure, the MPEG video encoder generates very bursty traffic dynamically on a picture-by-picture basis. As one can see, a periodic peak is from I pictures, the medium traffic is from P pictures, and the lowest traffic is from B pictures. The dynamic behavior of Fig. 2 is a general characteristic of the MPEG video traffic sequence, which is independent of the GOP sequence because the traffic depends only on the coding mode or picture type, not on the sequence. Jung and Meditch [2] have shown that MPEG video sequences of different GOP sequences have similar characteristics. In this paper, we develop an adaptive prediction scheme which is able to fully characterize the general feature and behavior of MPEG video sequences.

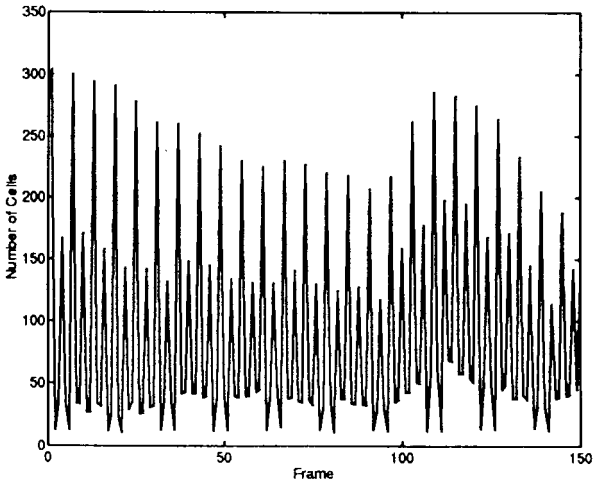


Fig. 2. Traffic signal of MPEG video: "Bike" Scene.

A. Optimal Traffic Prediction Model for MPEG Video Signals Based on Nonlinear ARMA Models

The main purpose of constructing the traffic models for the bit-rate variability of video is to create an aid for designing the future ATM communication networks that will carry multiplexed video signals. The autoregressive (AR) process is one of the simplest models which has proven to be very effective in predicting video signals with high correlation [8], [9]. However, this model is suitable for less bursty traffic like teleconferencing video, but not for very bursty traffic like MPEG video. Additionally, it is not particularly appropriate for evaluating statistical multiplexing. To tackle this difficulty, Grunenfelder *et al*. [13] describe a method of characterizing VBR video sources in ATM networks as an ARMA followed by a zero-memory nonlinearity (ZMNL). They have demonstrated that the ARMA process is suited to VBR video codes and ATM traffic. Actually, their model is a NARMA process. In this section, we would like to investigate the general NARMA models of MPEG video signals. Since each picture type has its own traffic characteristics, it is suggested that each picture type (I, P, or B) is modeled individually and can be expressed as

$$\begin{aligned} s_z(n) = {} & h_z(s_z(n-1), s_z(n-2), \cdots, s_z(n-p_z), \\ & e_z(n-1), e_z(n-2), \cdots, e_z(n-q_z)) + e_z(n) \\ & z \in \{\mathrm{I}, \mathrm{P}, \mathrm{B}\} \end{aligned} \tag{1}$$

where $s_z(n)$ is the traffic variable for picture type of z at time instant n, $h_z(\cdot)$ is an unknown smooth nonlinear function, and $e_z(n)$ is the unmodeled system error at time instant n. It is assumed that $E(e_z(n)|s_z(n-1), s_z(n-2), \cdots) = 0$, and that the variance of $e_z(n)$ is σ_z^2. In addition, the nonlinear function $h_z(\cdot)$ has a nonzero constant term if $s_z(n)$ is not a zero-mean process. Equation (1) is called a $\mathrm{NARMA}\,(p_z, q_z)$, where p_z and q_z are positive integers [20].

For simplified analysis, we neglect the index "z" of (1) in the remainder of this paper. From the NARMA (p,q) model of (1), the minimum mean-squared error predictor based on the infinite past of observations is the conditional mean [19], [20]

$$\begin{aligned}\hat{s}(n) &= E(s(n)|s(n-1), s(n-2), \cdots) \\ &= E[h(s(n-1), \cdots, s(n-p), e(n-1), e(n-2), \\ &\quad \cdots, e(n-q))|s(n-1), s(n-2), \cdots]. \end{aligned} \tag{2}$$

Moreover, assume that the NARMA (p,q) model is invertible so that there is a function $g(\cdot)$ such that

$$s(n) = g(s(n-1), s(n-2), \cdots) + e(n). \tag{3}$$

In other words, the current traffic variable $s(n)$ may be expressed as a function in terms of the infinite past of traffic variables $s(n-1), s(n-2), \cdots$. Thus, the $e(n-j)$ in (1) becomes a function of $s(i), -\infty < i \leq (n-j)$

$$\begin{aligned} e(n-j) &= \phi_{n-j}(s(i), -\infty < i \leq (n-j)), \\ & \qquad j = 1, 2, \cdots, q. \end{aligned} \tag{4}$$

Since $e(n-j)$ are specified by (4) in terms of present and past (relative to time $n-j$) traffic variables, the conditional mean predictor of (2) becomes

$$\begin{aligned}\hat{s}(n) &= h(s(n-1), s(n-2), \cdots, s(n-p), \\ & \qquad e(n-1), \cdots, e(n-q)). \end{aligned} \tag{5}$$

Note that the predictor has a mean-squared error σ^2.

However, one cannot compute both (4) and (5) directly because there are, in practice, only the finite past observations available in the calculation. Therefore, Connor *et al.* [21] have shown that it is reasonable to approximate the conditional mean predictor of (5) by the following recursive algorithm:

$$\begin{aligned}\hat{s}(n) &= h(s(n-1), s(n-2), \cdots, s(n-p), \\ & \qquad \hat{e}(n-1), \cdots, \hat{e}(n-q)) \end{aligned} \tag{6}$$

where the prediction errors $\hat{e}(j)$ are defined by

$$\hat{e}(j) = s(j) - \hat{s}(j), \qquad j = n-1, n-2, \cdots, n-q \tag{7}$$

with appropriate initial conditions discussed in [20].

Unfortunately, the main problem of performing the recursive algorithm (6) and (7) is that the explicit form of the nonlinear function $h(\cdot)$ is actually unknown. Connor *et al.* [21] have shown that the recursive formulation of the conditional mean predictor can be approximated by a class of recurrent neural networks within an acceptable accuracy. In the next section, we will investigate the applications of recurrent networks to the determination of the optimal NARMA predictor.

III. Determination of Optimal NARMA Predictor Using Recurrent Neural Networks

RNN's are neural networks that have feedback. RNN's are highly nonlinear dynamical systems which exhibit a rich and complex dynamical behavior. Moreover, the RNN allows any neuron in the network to be connected to any other neuron in the network. They have been proven better than traditional signal processing methods in modeling and predicting nonlinear and chaotic time series [17]. Connor *et al.* [21] have indicated that RNN's are well suited for the nonlinear prediction of NARMA process. There are a number of types of recurrent networks that have been proposed by several researchers [24], [25]. In this paper, we apply the most widely known architecture, proposed by Williams and Zipser [18], to time series prediction. A network with this particular architecture is also called the fully connected recurrent network. Consider a Williams–Zipser RNN consisting of a total of N neurons with p external input connections. Let $\boldsymbol{x}(n)$ denote the $p \times 1$ external input applied to the network at discrete time n, and let $\boldsymbol{y}(n+1)$ denote the corresponding $N \times 1$ vector of neuron outputs produced one step later at time $n+1$. The input vector $\boldsymbol{x}(n)$ and one-step delayed output vector $\boldsymbol{y}(n)$ are concatenated to form the $(p+N) \times 1$ vector $\boldsymbol{u}(n)$, whose ith element is denoted by $u_i(n)$. Let X denote the set of indexes i for which $u_i(n)$ is an external input, and let Y denote the set of indexes i for which $u_i(n)$ is the output of a neuron. The indexes on $\boldsymbol{y}$ and $\boldsymbol{x}$ are chosen to correspond to those of $\boldsymbol{u}$, so that

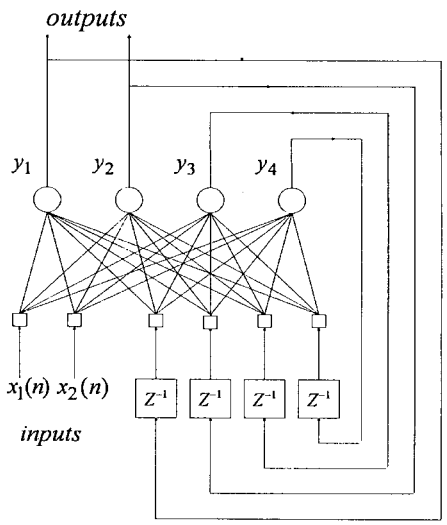


Fig. 3. Architectural graph of a fully connected recurent network for $p = 2$ and $N = 4$.

$$u_i(n) = \begin{cases} x_i(n), & \text{if } i \in X \\ y_i(n), & \text{if } i \in Y. \end{cases} \tag{8}$$

Generally, the RNN has two distinct layers, i.e., a concatenated input–output layer and a processing layer. This is illustrated in Fig. 3 for $p = 2$ and $N = 4$. The network is fully interconnected in that there are a total of pN forward connections and N^2 feedback connections. Let W denote the $N \times (p+N)$ synaptic weight matrix. An element w_{ji} of this matrix represents the weight of the connection from the ith node to the jth neuron. The net internal activity of neuron j at time n, for $j \in Y$, is computed by

$$v_j(n) = \sum_{i \in X \cup Y} w_{ji}(n) u_i(n) \tag{9}$$

where $X \cup Y$ is the set union of sets X and Y. At time $n+1$, the output of neuron j is computed by passing $v_j(n)$ through

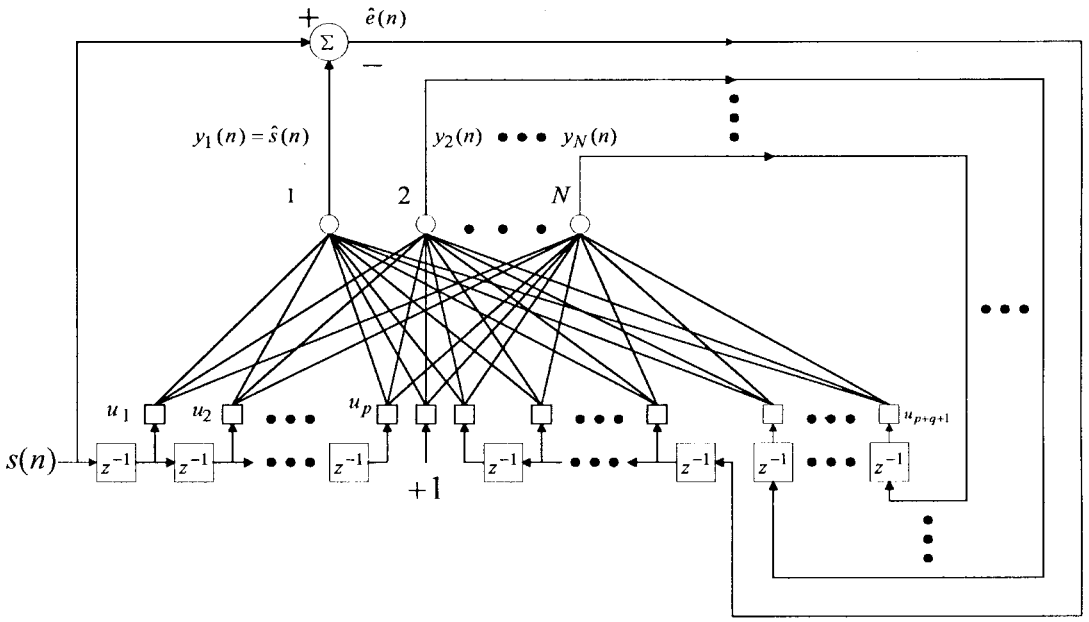


Fig. 4. Recurrent neural network implementation for $\mathrm{NARMA}(p,q)$ prediction model with prediction error input and N neurons.

a sigmoidal function $\varphi(\cdot)$ to obtain

$$y_j(n+1) = \varphi(v_j(n)) = \frac{1}{1+\exp(-v_j(n))}. \tag{10}$$

To implement the $\mathrm{NARMA}(p,q)$ prediction model by recurrent neural networks, additional $(q-1)$ input nodes should be added to the network. The RNN implementation a of $\mathrm{NARMA}(p,q)$ predictor is illustrated in Fig. 4. Let the first neuron output $y_1(n)$ be equal to the conditional mean predictor $\hat{s}(n)$. The prediction error at time $n, \hat{e}(n)$ is obtained by subtracting $\hat{s}(n)$ from the external input $s(n)$. By inputting $\hat{e}(n)$ into a tapped-delay-line filter with q delay elements, it yields q input nodes at time $n-1, u_{p+q+2-i}(n-1) = \hat{e}(n-i) = s(n-i) - \hat{s}(n-i), 1 \le i \le q$, where $u_{p+q+2-i}, 2 \le i \le q$ are the additional input nodes. Note that the indexes in the network inputs are arranged according to Fig. 4. Similarly, p external inputs to the network $x_i(n-1), 1 \le i \le p$ can be created by inputting $s(n)$ into a tapped-delay-line filter with p delay elements. Thus, $x_i(n-1) = s(n-i), 1 \le i \le p$. In addition, the remaining $(N-1)$ neuron outputs $y_i(n), 2 \le i \le N$ are fed back to its input. Finally, to accommodate a bias for each neuron, besides the $p+q+N-1$ inputs, we have included one other input whose value is always maintained at $+1$. Based on the above discussion, the explicit form of network input at time $n-1, u_i(n-1)$ can be expressed as follows:

$$u_i(n-1) = \begin{cases} x_i(n-1) = s(n-i), & 1 \le i \le p \\ +1, & i = p+1 \\ \hat{e}(n+i+2-p-q), & p+2 \le i \le p+q+1 \\ y_{i-p-q-1}(n-1), & p+q+2 \le i \le p+q+N. \end{cases} \tag{11}$$

Hence, the recurrent neural network with prediction error input provides a nonlinear approximation $\hat{h}(\cdot)$ to $h(\cdot)$ of (6) given by

$$\begin{aligned} \hat{s}(n) &= \hat{h}(s(n-1), \cdots, s(n-p), \hat{e}(n-1), \cdots, \hat{e}(n-q)) \\ &= y_1(n) = \varphi(v_1(n-1)) \\ &= \varphi\left(\sum_{i=1}^{p+q+N} w_{1i} u_i(n-1)\right) \\ &= \varphi\left[\sum_{i=1}^{p} w_{1i} s(n-i) + w_{1,p+1} \right. \\ &\qquad + \sum_{i=p+2}^{p+q+1} w_{1i} \hat{e}(n+i+2-p-q) \\ &\qquad \left. + \sum_{i=p+q+2}^{p+q+N} w_{1i} y_{i-p-q-1}(n-1)\right]. \end{aligned} \tag{12}$$

The synaptic weights w'_{ji}'s are estimated from a set of training samples $D = \{s(n)\}_{n=1}^{N_T}$, thereby obtaining an estimate $\hat{h}(\cdot)$ of $h(\cdot)$, where N_T denotes the number of training samples. Estimates are obtained by minimizing the sum of the squared prediction errors $\mathcal{E} = \Sigma_{n=1}^{N_T} \hat{e}^2(n)$. However, the formulation of the RNN with prediction error inputs of (12) cannot apply the well-known RTRL algorithm [18] to determine the appropriate synaptic weights directly. Therefore, we would like to reformulate the recursive formulation of (6) and (7) as a new function H in terms of the delayed traffic variables $s(n-i), 1 \le i \le p$ and the past approximate conditional mean prediction values $\hat{s}(n-j), 1 \le j \le q$, as follows:

$$\begin{aligned} \hat{s}(n) &= h[s(n-1), \cdots, s(n-p), (s(n-1) - \hat{s}(n-1)), \\ &\qquad \cdots, (s(n-q) - \hat{s}(n-q))] \\ &= H(s(n-1), s(n-2), \cdots, s(n-p), \hat{s}(n-1), \\ &\qquad \cdots, \hat{s}(n-q)). \end{aligned} \tag{13}$$

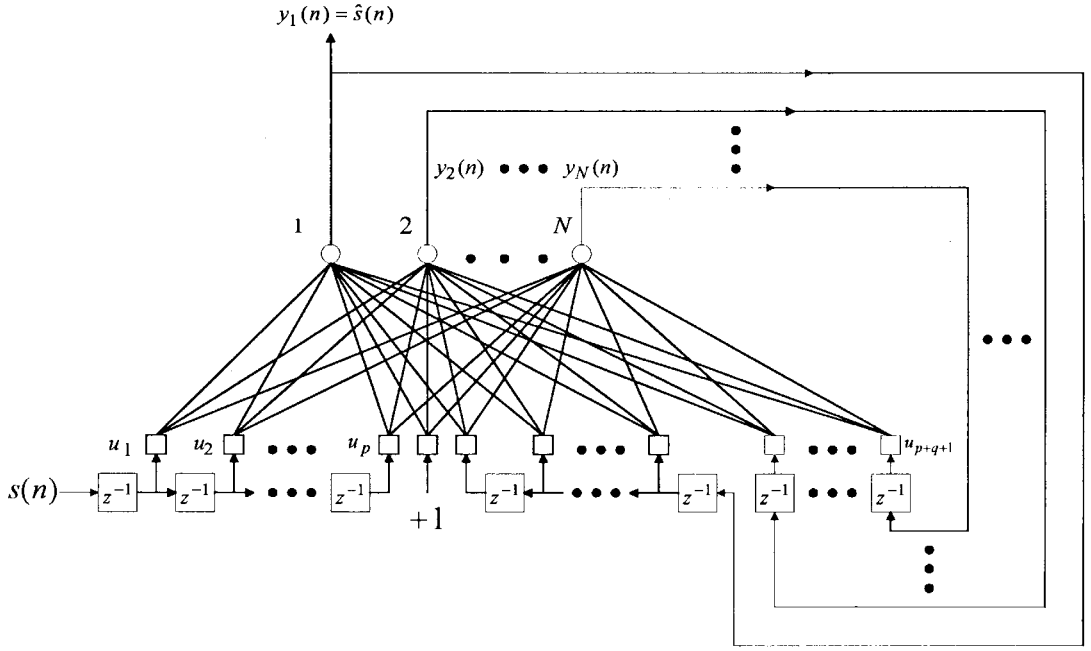


Fig. 5. Recurrent neural network implementation for $\mathrm{NARMA}\,(p, q)$ prediction model with normal input and N neurons.

Indeed, the expression of (13) is an alternative formulation of the conditional mean predictor for the $\mathrm{NARMA}\,(p, q)$ process. For the RNN implementation of the $\mathrm{NARMA}\,(p, q)$ predictor model of (13), the recurrent network topology is shown in Fig. 5, and its inputs are given by

$$u_i(n-1) = \begin{cases} s(n-i), & 1 \le i \le p \\ +1, & i = p+1 \\ \hat{s}(n+i+2-p-q), & p+2 \le i \le p+q+1 \\ y_{i-p-q-1}(n-1), & p+q+2 \le i \le p+q+N. \end{cases} \tag{14}$$

The resulting network topology is well suited to the RTRL algorithm. Similarly, an RNN nonlinear approximation $\hat{H}(\cdot)$ to $H(\cdot)$ of (13) is written as

$$\begin{aligned} \hat{s}(n) &= \hat{H}(s(n-1), \cdots, s(n-p), \hat{s}(n-1), \cdots, \hat{s}(n-q)) \\ &= \varphi\left[\sum_{i=1}^{p} w_{1i} s(n-i) + w_{1,p+1} \right. \\ &\qquad + \sum_{i=p+2}^{p+q+1} w_{1i} \hat{s}(n+i+2-p-q) \\ &\qquad \left. + \sum_{i=p+q+2}^{p+q+N} w_{1i} y_{i-p-q-1}(n-1)\right]. \end{aligned} \tag{15}$$

Li [26] has shown that a recurrent neural network of (15) with a sufficiently large number of neurons and appropriate weights can be found by performing the RTRL algorithm such that the sum of the squared prediction errors $\mathcal{E} < \epsilon$ for an arbitrary $\epsilon > 0$. In other words, $||H - \hat{H}||_D < \epsilon$, where $||\cdot||_D$ denotes the L_2 norm with respect to the training set D. Moreover, the RNN has the ability to generalize learning to what has never been seen [18]. This is called the generalization from learning. This is particularly useful for learning a dynamic environment for traffic prediction of a VBR video signal via ATM networks where observations may be incomplete, delayed, or partially available. Thus, $||H - \hat{H}||_{\tilde{D}} < \epsilon$, where $\tilde{D}$ denotes the set union of D and a set of data that does not belong to D, and $D \subseteq \tilde{D}$. The RTRL algorithm is capable of nonlinear adaptive prediction of nonstationary signals, and does not require *a priori* knowledge of time dependence among the input data. However, a major limitation of the RTRL algorithm is that its computational complexity is proportional to $O(N^4)$, where N is the total number of neurons in the network. Since a sufficiently large number of neurons are required to maintain the prediction accuracy of the RNN-based NARMA predictor, it seems infeasible to achieve the RNN-based prediction within an acceptably small time interval. To tackle this difficulty, Haykin and Li [22] proposed a new RNN structure called the PRNN that is an extension of the conventional RTRL algorithm. The design of such a network is based on the principle of divide and conquer, that is, a complex RNN with a large number of neurons can be divided into a number of simpler small-scale RNN modules with less computational complexity. In the following section, we apply the PRNN structure to improve the computational performance of the NARMA conditional mean predictor. It should be mentioned that this section provides the fundamentals of generalization from learning for PRNN since, from Figs. 5 and 6, PRNN is an extended pipeline structure of RNN-based NARMA predictor with normal input.

IV. Low-Complexity Pipelined Recurrent Neural Networks for Optimal Adaptive NARMA Predictors with Learning-Rate Annealing Schedules

The PRNN shown in Fig. 6 is composed of q identical modules, each of which is designed as a fully connected

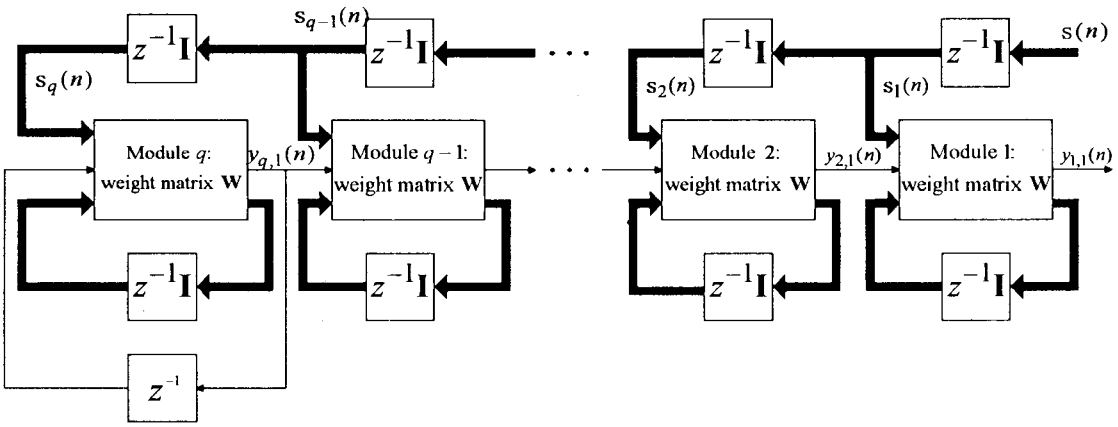


Fig. 6. Pipelined recurrent neural network with q modulus.

recurrent network with N neurons. Each module has $N-1$ neuron outputs fed back to its input, and the remaining neuron output (the first neuron output) is applied directly to the next module. In the case of module q, a one-unit delayed version of the module's output is assumed to be fed back to the input. Information flow into and out of the modules proceeds in a synchronized fashion. Fig. 7 shows the detailed structure of module i with N neurons and p external inputs. All of the modules have exactly the same number of external inputs and internal feedback signals. Note that for module q, its module output acts as an external feedback signal to itself. In addition, all of the modules of PRNN are designed to have exactly the same $(p+N+1)\times N$ synaptic weight matrix W. The updated value of the synaptic weight matrix W is computed using the RTRL algorithm [18]. Haykin and Li [22] have demonstrated that the PRNN is able to provide satisfactory accuracy of the nonlinear adaptive prediction of a nonstationary signal and time series process. An important feature of the PRNN is its high computational efficiency. Specifically, the total computational requirement of processing a single sample on a PRNN is $O(qN^4)$ arithmetic operations. However, this is to be contrasted with the computational requirement of a corresponding structure involving the use of a conventional RNN with qN neurons, that is, $O(q^4N^4)$ arithmetic operations. Thus, the computational savings made possible by the use of PRNN can indeed be enormous for large q. In order to further improve the convergence performance of the RTRL algorithm for PRNN, we would like to redesign the RTRL algorithm using the learning-rate annealing schedules [23]. First, an overview of PRNN is summarized as follows.

For the ith module, its external input at the nth time instant is described by the $p\times 1$ vector

$$\boldsymbol{s}_i(n)=[s(n-i),s(n-(i+1)),\cdots,s(n-(i+p-1))]^T \quad (16)$$

where p is the nonlinear prediction order according to the NARMA(p,q) process. The other input vector applied to module i is the $N\times 1$ feedback vector

$$\boldsymbol{r}_i(n)=[y_{i+1,1}(n),\boldsymbol{r}_i'(n)]^T, \qquad i=1,2,\cdots,(q-1) \quad (17)$$

where $y_{i+1,1}(n)$ is the first neuron's output in the adjacent module $i+1$ and $\boldsymbol{r}_i'(n)$ denotes the internal feedback signal

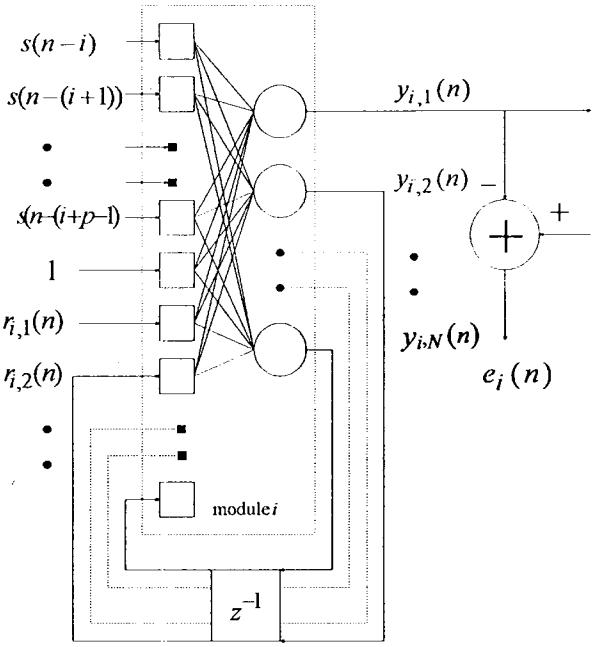


Fig. 7. Detailed architecture of module i of the PRNN.

consisting of the one-step delayed output signals of neurons $2,3,\cdots,N$ in module i and can be written as

$$\boldsymbol{r}_i'(n)=[y_{i,2}(n-1),\cdots,y_{i,N}(n-1)]^T, \quad i=1,2,\cdots,(q-1). \quad (18)$$

Note that $\boldsymbol{r}_i'(n)$ denotes feedback signals that originate from module i itself. The last module of the PRNN operates as a standard fully connected recurrent neural network. Thus, $\boldsymbol{r}_q(n)$ is written as

$$\boldsymbol{r}_q(n)=[y_{q,1}(n-1),y_{q,2}(n-1),\cdots,y_{q,N}(n-1)]^T. \quad (19)$$

Additionally, the fixed input $+1$ is included for the provision of a threshold applied to each neuron in module i. Based on the above discussion, an input vector consisting of total $(p+N+1)$ input signals applied to module i is represented by

$$\boldsymbol{u}_i=[\boldsymbol{s}_i^T(n),1,\boldsymbol{r}_i^T(n)]^T, \qquad i=1,2,\cdots,q. \quad (20)$$

Thus, the lth element of $\boldsymbol{u}_i$ is represented by

$$u_{il}=\begin{cases} s(n-(i+l-1)), & 1\le l\le p, 1\le i\le q \\ 1, & 1\le l\le p, 1\le i\le q \\ y_{i+1,l-(p+1)}(n), & l=p+2, 1\le i\le q-1 \\ y_{i,l-(p+1)}(n), & l=p+2, i=q \\ y_{i,l-(p+1)}(n), & p+3\le l\le p+1+N, \\ & 1\le i\le q. \end{cases} \tag{21}$$

Thus the output $y_{ik}(n)$ of neuron k in module i is given by

$$y_{ik}(n)=\varphi(v_{ik}) \tag{22}$$

where the net internal activity v_{ik} is calculated by

$$\begin{aligned} v_{ik} &= \sum_{l=1}^{p+N+1} w_{kl}u_{il} \\ &= \sum_{l=1}^{p} w_{kl}s(n-(i+l-1))+w_{k,p+1} \\ &\quad + \sum_{l=p+2}^{p+N+1} w_{kl}r_{i,l-(p+1)}(n). \end{aligned} \tag{23}$$

Finally, the PRNN prediction at time instant n is defined by the output of the first neuron of the first module as shown by

$$\hat{s}(n)=\hat{s}_1(n+1)=y_{1,1}(n). \tag{24}$$

Note that $\hat{s}_1(n)=\hat{s}(n-1)$.

The pipeline recurrent network is characterized by a nested nonlinearity. Since all of the neurons have a common nonlinear activation function $\varphi(\cdot)$, the functional dependence of the output $y_{1,1}(n)(=\hat{s}(n))$ of the network can be expressed as follows:

$$\begin{aligned} \hat{s}(n) &= y_{1,1}(n)=\varphi(\boldsymbol{s}_1(n), y_{2,1}(n)) \\ &= \varphi(\boldsymbol{s}_1(n), \varphi(\boldsymbol{s}_2(n), y_{3,1}(n))) \\ &= \varphi(\boldsymbol{s}(n-1), \varphi(\boldsymbol{s}(n-2), \varphi(\boldsymbol{s}(n-3), \cdots, \\ &\quad \varphi(\boldsymbol{s}(n-q), y_{q,1}(n-q)), \cdots))) \end{aligned} \tag{25}$$

where we have omitted the dependence on the synaptic weight matrix W that is common to all of the q modules, and $\boldsymbol{s}_i(n)=\boldsymbol{s}(n-i)$. The expression of (25) is indeed the nested nonlinearity that gives the PRNN its enhanced computing power compared to the conventional recurrent network. Moreover, the scheme of nested nonlinear functions described in (25) is unusual in the classical approximation theory. Indeed, it is a universal approximator in the sense that a PRNN with appropriate training can approximate any nonlinear ARMA process to any desired degree of accuracy, provided that sufficiently many hidden neurons are available [26].

Certainly, $y_{i,1}(n)$ is interpreted as the one-step prediction of $s_i(n)$ computed by the ith module whose functional dependence can be described by a complete dependence form shown as follows:

$$\hat{s}_i(n+1)=y_{i,1}(n)=\varphi(W, \boldsymbol{s}_i(n), \boldsymbol{r}_i(n)) \tag{26}$$

where W is the $N\times(p+N+1)$ synaptic weight matrix of module i, and $\boldsymbol{r}_i(n)$ is the input vector defined in (17) and (19). The desired response for module i at time instant n is $s_i(n+1)=s(n-i+1)$. Hence, the prediction error for module i is given by

$$\begin{aligned} \hat{e}_i(n) &= s_i(n+1)-\hat{s}_i(n+1)=s_i(n+1)-y_{i,1}(n) \\ &= s(n-i+1)-y_{i,1}(n). \end{aligned} \tag{27}$$

Thus, an overall cost function for the PRNN is defined by

$$\mathcal{E}(n)=\sum_{i=1}^{q}\lambda^{i-1}\hat{e}_i^2(n) \tag{28}$$

where λ is an exponential forgetting factor that lies in the range of $0<\lambda\le 1$. The inverse of $1-\lambda$ is a measure of the memory of the PRNN. Adjustments to the synaptic weight matrix W of each module is made to minimize $\mathcal{E}(n)$ in accordance with the RTRL algorithm. In order to further speed up the convergence rate, in the next section, a RTRL algorithm incorporated with learning-rate annealing schedules is presented to achieve the goal.

A. Real-Time Recurrent Learning (RTRL) Algorithm with Learning-Rate Annealing Schedules

For the case of a particular weight w_{kl}, its incremental change $\Delta w_{kl}(n)$ made at time n according to the method of steepest descent is given by

$$\Delta w_{kl}(n)=-\eta\frac{\partial\mathcal{E}(n)}{\partial w_{kl}} \tag{29}$$

where η is the learning-rate parameter. From (27)–(29), we note that

$$\begin{aligned} \frac{\partial\mathcal{E}(n)}{\partial w_{kl}} &= 2\sum_{i=1}^{q}\lambda^{i-1}\hat{e}_i(n)\frac{\partial\hat{e}_i(n)}{\partial w_{kl}} \\ &= -2\sum_{i=1}^{q}\lambda^{i-1}\hat{e}_i(n)\frac{\partial y_{i,1}(n)}{\partial w_{kl}}. \end{aligned} \tag{30}$$

In (30), the partial derivative $(\partial y_{i,1}(n)/\partial w_{kl})$ is calculated using a modification of the RTRL algorithm [18]. A quadruply indexed set of variables $\{\pi_{kl}^{ij}(n)\}$ is introduced to characterize the RTRL algorithm, and each element of the set is given by

$$\begin{aligned} \pi_{kl}^{ij}=\frac{\partial y_{ij}(n)}{\partial w_{kl}} \qquad & 1\le i\le q, 1\le j,k\le N \\ & 1\le l\le p+1+N. \end{aligned} \tag{31}$$

Note that $\pi_{kl}^{i1}=(\partial y_{i,1}/\partial w_{kl})$. The RTRL algorithm is used to recursively compute the values of π_{kl}^{ij} for every time step and all appropriate i, j, k, and l as follows:

$$\pi_{kl}^{ij}(n+1)=\varphi'(v_{ij})\left\{\sum_{m=1}^{N}w_{jm}(n)\pi_{kl}^{im}(n)+\delta_{kj}u_{il}(n)\right\} \tag{32}$$

with initial conditions

$$\pi_{kl}^{ij}(0)=0 \tag{33}$$

where δ_{kj} is a Kronecker delta equal to one when $j=k$ and zero otherwise; u_{il} and v_{ij} are defined in (21) and (23),

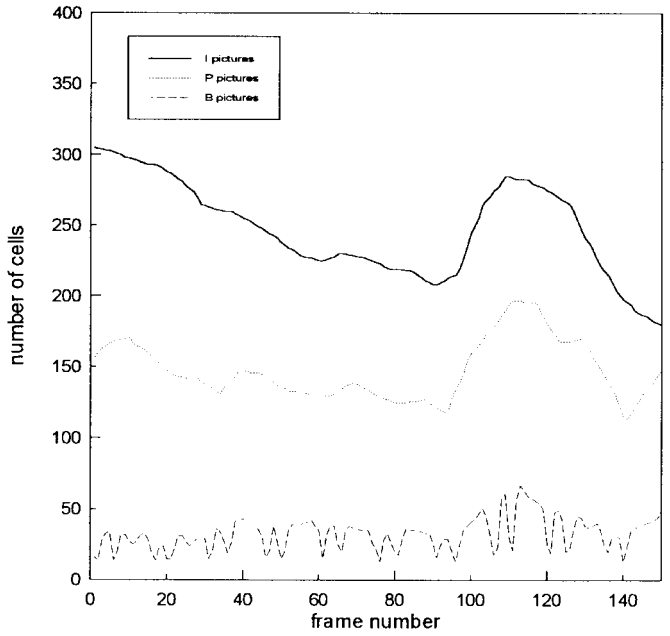


Fig. 8. Traffic signals of the decomposed three pictures (I, P, and B) resulting from MPEG video "Bike" scene.

respectively. From (10), we find that the derivative $\varphi'(\cdot)$ is given by

$$\varphi'(v_{ij}) = y_{ij}(n)(1 - y_{ij}(n)). \tag{34}$$

Hence, it is possible to determine the value of $\pi_{kl}^{i1}(n)$ at time instant n by the recursion of (32) and (33). As a result, from (29)–(31), the change applied to the (k,l)th element of the synaptic weight matrix is calculated by the following equation:

$$\Delta w_{kl}(n) = 2\eta \sum_{i=1}^{q} \lambda^{i-1} \hat{e}_i(n) \pi_{kl}^{i1}(n). \tag{35}$$

The weight w_{kl} is updated in accordance with

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n). \tag{36}$$

The difficulties encountered with the RTRL algorithm may be attributed to the fact that the learning-rate parameter η is maintained constant throughout the computation, that is, $\eta(n) = \eta_0$ for all n. To overcome this difficulty, we would like to use the learning-rate annealing schedule commonly used in the LMS algorithm [23] to accelerate the learning process of the RTRL algorithm. The most well-known annealing schedule is the search-then-converge schedule [23], defined by

$$\eta(n) = \frac{\eta_0}{1 + \dfrac{n}{\tau}} \tag{37}$$

where τ denotes the search time constant. In the early stages of adaptation involving time step n small compared to the search time constant τ, the learning-rate parameter $\eta(n)$ is approximately equal to η_0, and the algorithm operates essentially as the standard constant learning-rate RTRL algorithm. For time step n large compared to τ, the learning-rate parameter $\eta(n)$ approximates as (c/n), where $c = \tau\eta_0$. The algorithm now operates as a traditional stochastic approximation algorithm [23],

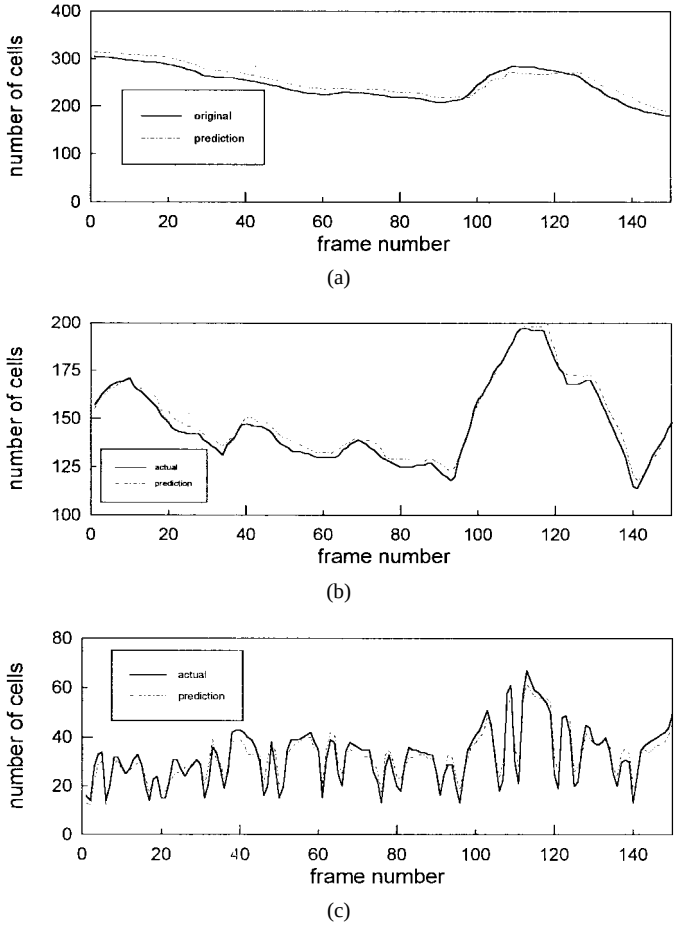


Fig. 9. Comparison of actual traffic signal and the prediction obtained by the PRNN-based optimal prediction for three picture types: (a) I pictures, (b) P pictures, and (c) B pictures.

and the synaptic weights converge to their optimum values. It thus appears that the search-then-converge scheme combines the desirable features of the standard RTRL algorithm in an LMS manner and traditional stochastic approximation algorithms.

B. Initialization of the PRNN

For the initialization of the synaptic weight matrix W, a traditional epochwise training method [22] is applied to one module of a recurrent neural network operating with N_0 samples of the input signal. A set of training pairs $\{\boldsymbol{s}(n), d(n)\}_{n=1}^{N_I}$ is constructed to perform the initialization, where $N_I = N_0 - p$, the desired signal $d(n)$ is equal to $s(n+p)$, and the input vector $\boldsymbol{s}(n)$ is defined by

$$\boldsymbol{s}(n) = [s(n + (p-1)), \cdots, s(n)]^T. \tag{38}$$

A cost function $\mathcal{E}_I$ of the initialization, obtained by summing $\hat{e}^2(n)$ over a time interval $[1, N_I]$, is defined by

$$\mathcal{E}_I = \frac{1}{N_I} \sum_{n=1}^{N_I} \hat{e}^2(n) \tag{39}$$

where $\hat{e}(n) = s(n+p) - y_1(n)$ and $y_1(n)$ denotes the first neuron output of the module. Similarly, the change applied

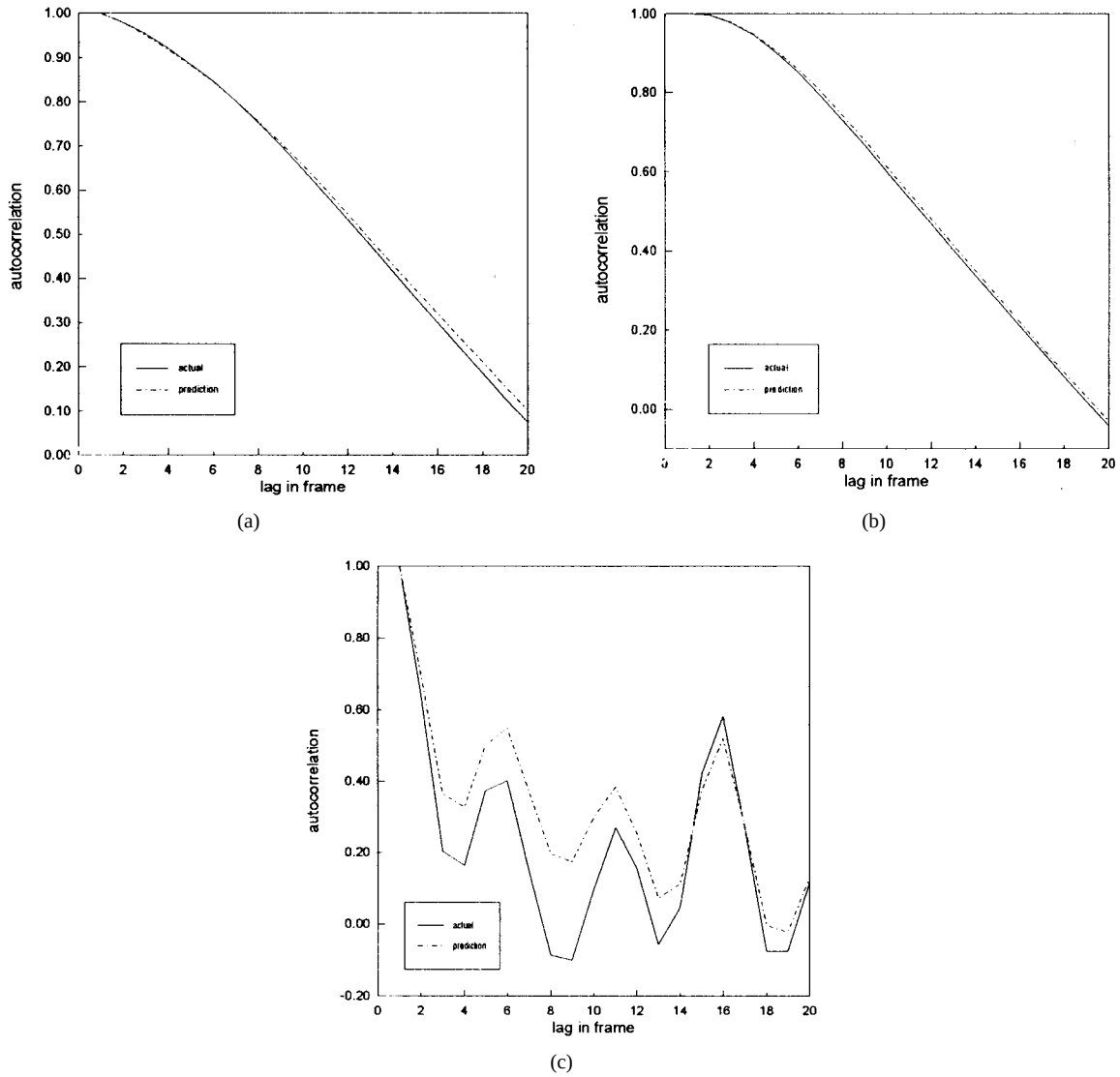


Fig. 10. Comparison of autocorrelation function for actual I, P, and B picture sequences (solid line) and their corresponding PRNN-based predictions (dashed line).

to w_{kl} is

$$\Delta w_{kl} = -\eta \frac{\partial \mathcal{E}_I}{\partial w_{kl}} \tag{40}$$

where

$$\begin{aligned} \frac{\partial \mathcal{E}_I}{\partial w_{kl}} &= -\frac{2}{N_I} \sum_{n=1}^{N_I} \hat{e}(n) \frac{\partial y_1(n)}{\partial w_{kl}} \\ &= -\frac{2}{N_I} \sum_{n=1}^{N_I} \hat{e}(n) \pi_{kl}^1(n) \end{aligned} \tag{41}$$

where $\pi_{kl}^1(n) = (\partial y_1(n)/\partial w_{kl})$ and $\pi_{kl}^j(n) = (\partial y_j(n)/\partial w_{kl})$, $1 \le j,k \le N, 1 \le l \le p+N+1$.The values of the triply indexed variables $\pi_{kl}^j(n)$ can be computed by the RTRL algorithm with a learning-rate annealing scheme by inputting repeatedly the N_I training pairs to the RNN module until $\mathcal{E}_I$ is less than a permitted value ϵ_I. Usually, the permitted error ϵ_I is chosen as a value which is about 1% of the mean-square error of the input signal $s(n)$. Note that this module operates as a fully connected RNN with $\boldsymbol{r}(n) = \boldsymbol{y}(n-1)$.

C. Description of the PRNN Algorithm for Optimal Adaptive NARMA Prediction Based on Satisfaction-or-Pass Strategy

Suppose now that a set of observation samples $D = \{s(n)\}_{n=1}^{N_T}$ is generated by a $\mathrm{NARMA}(p,q)$ process. These observation samples are referred to as an input signal $s(n)$ to the PRNN. The sampling time interval T_s should be selected such that the input signal captures the correlation properties of the traffic variable for each picture type of the MPEG video signal. The PRNN is developed to predict the value of the traffic variable for each picture type based upon sampled values taken from the previous measurement period. To further improve the prediction accuracy of the PRNN-based predictor, we will apply the strategy of satisfaction-or-pass to modify the above-mentioned PRNN learning algorithm. For example, the adjustment of the synaptic weight matrix is repeatedly evaluated by performing (32) and (34)–(36) within a limited sampling time interval until the overall cost function for the PRNN $\mathcal{E}(n)$ of (28) is less than ϵ (satisfaction) which is a permitted error tolerance for the

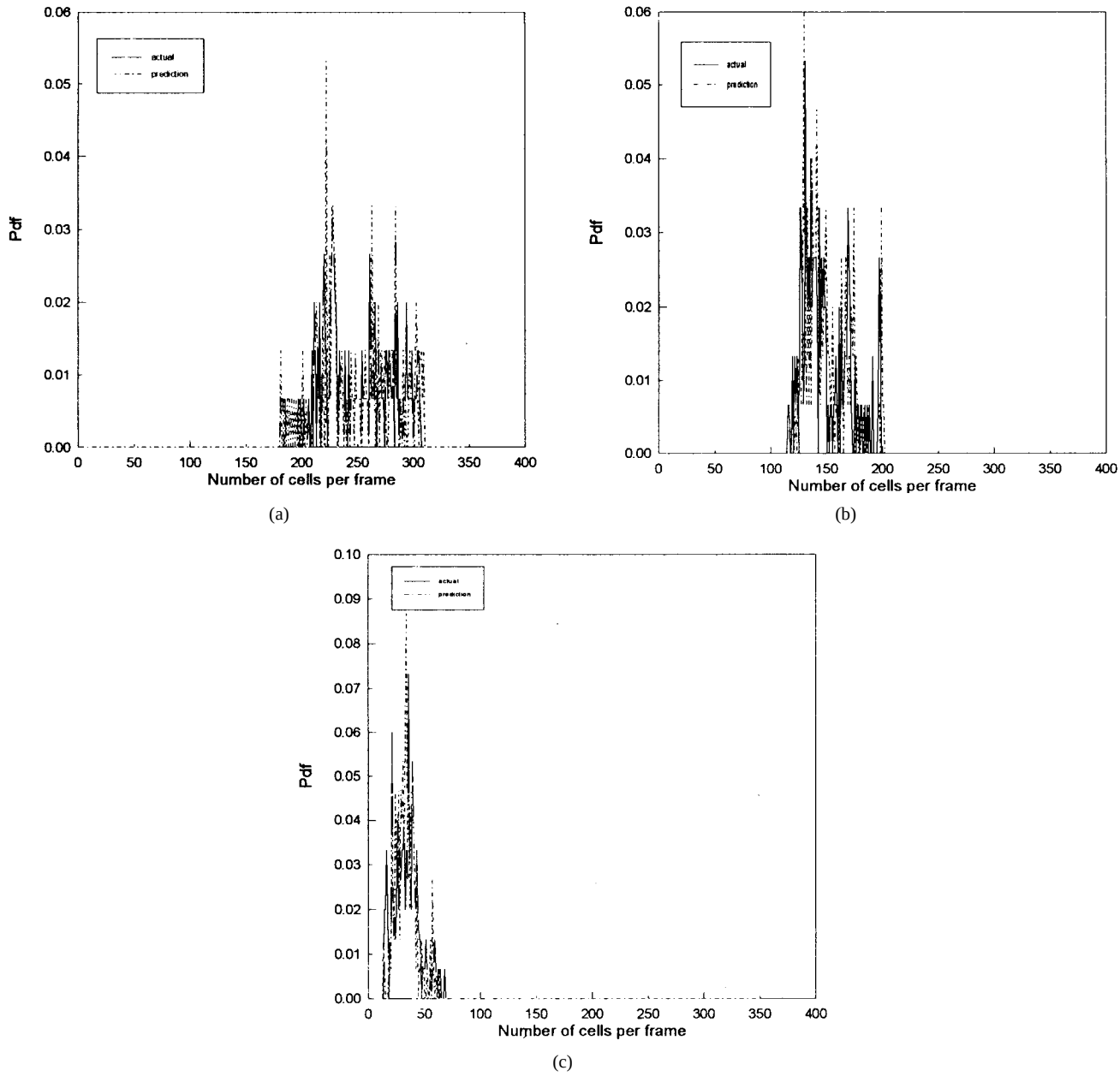


Fig. 11. Comparison of probability density distribution for actual I, P, and B picture sequences (solid line) and their corresponding PRNN-based predictions (dashed line).

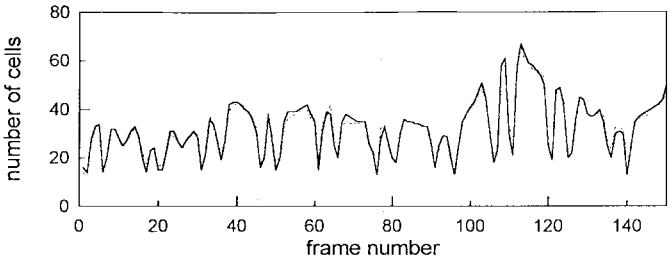
network prediction. However, sometimes, $\mathcal{E}(n)$ may not fall within the prescribed value of ϵ in the limited short time period. Thus, when the number of adjustments exceeds a permitted number m_p, the adjustment is terminated (pass) even $\mathcal{E}(n)$ is still larger than ϵ. For this case, the permitted number of adjustments within the sampling time interval is equal to m_p, where $m_p < (T_s/T_c)$ and T_c denotes the computation time of processing a weight adjustment on the PRNN. m_p is also called the permitted number for a single sample. Thus, a one-step traffic prediction based on the updated value of the synaptic weight matrix is performed. The adjustments will be started again when the next new traffic sample is applied to the network. Here, it is suggested that the value of ϵ is set to about 1–10% of the amplitude of the traffic signal $s(n)$. Note that the computational requirement of processing a single weight adjustment on the PRNN is $O(qN^4)$. Thus, the maximum total computational requirement of processing a single sample becomes $O(m_p qN^4)$ when the "pass" occurs for the sample. In order to reduce the computational complexity, we can apply the very-large-scale integration (VLSI) circuit and parallel computer techniques [27] to implement the training algorithm of PRNN in order to significantly reduce the computation time T_c. For instance, the computational complexity of an adjustment on the PRNN using a parallel computer consisting of q processors can be reduced to $O(N^4)$ since each module is performed independently using its corresponding processor.

V. Simulation Results

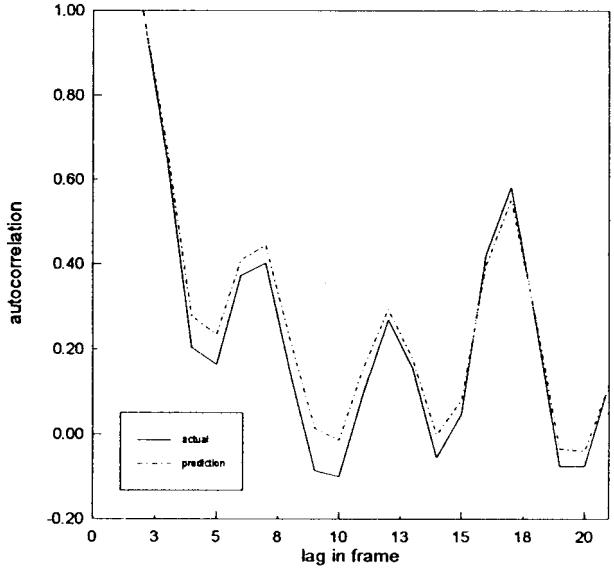
To verify the effectiveness of the PRNN-based optimal traffic predictor, a typical MPEG video source "Bike" is

adopted in this section as an example. A PVRG-MPEG software codec developed at Stanford University was used for simulation. The "Bike," which lasts for about 5 s, is composed of I frames, P frames, and B frames. The source characteristics of the "Bike" MPEG video may be found in [2]. Fig. 8 presents the decomposed traffic signals from the "Bike" scene which clearly shows that the original signal can be readily decomposed into three (I, P, and B) components. The traffic of I or P pictures changes rapidly as shown from frame 100–frame 120 in the figure because the correlation is low during a motion or scene change period. Although B pictures exhibit the lowest traffic, the traffic of B pictures changes very frequently, and generates a very complicated traffic curve. The optimal traffic predictor for each picture type (I, P, or B) is performed by the PRNN individually because each picture type has its own traffic characteristics. The parameters of the proposed PRNN-based optimal traffic predictor for each picture type were assumed to be identical in the calculations: the nonlinear predictor order (p_I, p_P, or p_B) is selected as four, and the number of modules (q_I, q_P, or q_B) is chosen as five. The forgetting factor λ of (28) is set to 0.9. The initial learning rate η_0 and the search time constant τ of (37) are set to 0.9 and $(m_p/3)$, respectively, where m_p denotes the permitted number of synaptic weight adjustments per sample and is set to 1000. It should be mentioned that the choice of value assigned to the number of neurons per module (N_I, N_P, or N_B) is dependent on the degree of variation in the traffic curve for each picture type. According to the above suggestion and observing Fig. 8, N_I, N_P, and N_B are chosen as 3, 3, and 4, respectively. Hence, the total number of neurons in the PRNN is 15 (or 15 or 20) for I (or P or B) pictures. For example, for I or B pictures, the computational complexity of an adjustment on the PRNN is on the order of 5×3^4 (= 405),whereas the computational complexity of an adjustment on the conventional recurrent network is on the order of $5^4 \times 3^4$ ($\approx 5 \times 10^4$).Thus, the PRNN reduces the computational complexity by more than two orders of magnitude. Moreover, by using a five-processor parallel computer, computational complexity per adjustment of PRNN is reduced to $O(3^4\ (= 81))$. One other design parameter that needs to be specified is the number of pretraining samples N_0 in the initialization of the PRNN. The main purpose of pretraining is merely to determine an adequate set of initial synaptic weights. To economize on pretraining time, the size of pretraining samples N_0 is limited to a specified small value. Here, N_0 is set to 25.

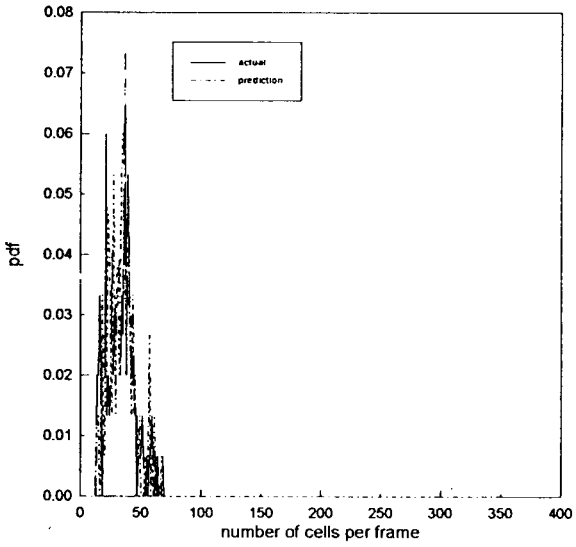
Fig. 9(a) shows a plot of 150 samples of the traffic signal of I pictures versus the number of frames. The continuous curve is the actual traffic signal of I pictures, and the dashed curve is the one-step prediction performed by the nonlinear adaptive PRNN-based predictor with $N_I = 3$. Similarly, a comparsion of the prediction and the actual traffic signal is shown in Fig. 9(b) for P pictures or Fig. 9(c) for B pictures. The predictions are very close to their corresponding actual traffic signals. In addition, the prediction errors for the traffic signal of I or P or B pictures are still maintained within a specific small value, even though the scene changes occurs during a time interval between frame 100 and frame 120.



(a)



(b)



(c)

Fig. 12. Comparison of actual traffic signal and its autocorrelation and probability density for B pictures (solid line) and their corresponding PRNN-based predictions (dashed line) with $N_B = 6$.

The performance of the predictor can be characterized by a performance index called the relative rms prediction error. The relative rms prediction error is obtained by averaging the squared prediction error relative to the squared value of its actual traffic over a time window from the initial time point to 150 T_s, where T_s denotes the frame (sampling) time interval.

The expression of the relative rms prediction error is given by

$$E_{\rm rms} = \sqrt{\frac{1}{150}\sum_{n=1}^{150}\left(\frac{\hat{e}(n)}{s(n)}\right)^2}. \quad (42)$$

Moreover, in order to achieve the best prediction, the permitted error ϵ is chosen to be a smaller value which is equal to 3% of $s(n)$. As a result, the relative (absolute) rms prediction errors for I, P, and B pictures are 0.963% (2.38 cells), 2.34% (3.11 cells), and 6.73% (3.77 cells), respectively. From the above results, it turns out that the number of satisfactions is greater than the number of passes in an rms sense for I pictures because $E_{\rm rms} = 0.953\% < 3\%$. However, for B pictures, the number of passes is greater than the number of satisfactions. In contrast to ϵ of 3%, if ϵ may be selected to be a larger value, for example, 10% of $s(n)$, this would result in fast prediction action, but less prediction accuracy. In addition, from the universal approximation property of PRNN, the value of $E_{\rm rms}$ of B pictures will become an extremely small number when the number of neurons per module N_B is sufficiently large. But it also increases the computational complexity. For example, the value of $E_{\rm rms}$ of B pictures is reduced to 1.632% when N_B is increased to six. Thus, the computational complexity per adjustment of PRNN is on the order of 5×6^4 ($\approx 6 \times 10^3$) using a single-processor computer or 6^4 ($\approx 1.3 \times 10^3$) using a five-processor parallel computer. For this case, it is seen that the number of satisfactions is greater than the number of passes for B pictures with $N_B = 6$ since $E_{\rm rms} = 1.632\% < 3\%$. In summary, it should be noted that the tradeoff between the prediction accuracy and computational efficiency is dependent on a proper selection of ϵ, N, and m_p. In our case, we choose $N_B = 4$ for the purpose of achieving the real-time traffic prediction, even though this would sacrifice more in prediction accuracy. Furthermore, we choose $\epsilon = 3\%$ to ensure its relative rms prediction error within an acceptably small range. Nomura *et al*. [8] and Ohta [9] have indicated that the autocorrelation function is one of the simplest measures used to characterize the temporal variation behavior of the video sequence. Fig. 10 depicts the autocorrelation obtained from real MPEG video data and the prediction that is generated by PRNN. The PRNN-based optimal prediction model demonstrates a better performance in curve fitting for I and P pictures, as shown in Fig. 10(a) and (b), respectively. It is clear from Fig. 10(a) and (b) that the autocorrelation decreases monotonically, and its shape is similar for I and P pictures. In constrast to I and P pictures, Fig. 10(c) shows that the autocorrelation decreases monotonically for the first three frames, but that the correlations increase slightly again at the fourth frame. More precisely, the autocorrelation has a ripple effect in its curve after the fourth frame. This is probably because bike motion has a natural period of four frames during the motion. Meanwhile, the PRNN produces a prediction which is able to capture the autocorrelation property of B pictures and demonstrates an acceptable approximation accuracy. The most basic measures for characterizing bit-rate variations are measures that characterize the statistical distribution. Fig. 11 shows the probability density distribution of bit-rate variation for each of the three picture types of MPEG video. The span for the distribution is between the minimum frame size or number of cells per frame and the maximum frame size (= 320 cells). These distributions are bell shaped, and they may be modeled as a normal distribution. The shape of the distribution is similar for all three picture types with different means and variances. As a goodness-of-fit test, the probability distribution function (pdf) of the predicted traffic signal for each picture type is very close to that of the original one.

Finally, we would like to show that the best approximation property of PRNN-based prediction can be achieved when the number of neurons per module of the PRNN is sufficiently large. As an example mentioned above, the prediction performance of a PRNN-based predictor for B pictures has been significantly improved when N_B is larger than 6. Fig. 12 demonstrates better performance in curve fitting for the traffic curve, autocorrelation, and probability density distribution when N_B is equal to six.

VI. Conclusion

This paper has presented a new optimal prediction model based on a pipelined recurrent neural network which is capable of predicting the MPEG video traffic by using a modification of the RTRL algorithm. The modified RTRL algorithm with learning-rate annealing schedules is well suited for the PRNN to learn highly dynamic situations such as the status of MPEG video traffic. This algorithm enhances the flexibility of the PRNN-based prediction model to adapt to the traffic bit-rate fluctuations via the dynamic network environment. Simulation results have shown that the PRNN approach provides accurate real-time prediction for MPEG video traffic. This verifies the effectiveness of the best approximation capability of the PRNN.

References

[1] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, Apr. 1991.

[2] S. Jung and J. S. Meditch, "Adaptive prediction and smoothing of MPEG video in ATM networks," in *Proc. IEEE Int. Conf. Commun. (ICC'95)*, Washington, DC, June 1995, pp. 832–836.

[3] P. Pancha and M. ElZarki, "Bandwidth-allocation schemes for variable bit-rate MPEG sources in ATM networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 190–198, June 1993.

[4] J. E. Neves, L. B. Almeida, and M. J. Leitco, "B-ISDN connection admission and routing strategy with traffic prediction by neural networks," in *Proc. SUPERCOM–ICC'94*, New Orleans, LA, May 1994.

[5] J. E. Neves, M. J. Leitco, and L. B. Almeida, "Neural networks in B-ISDN flow control: ATM traffic prediction or network modeling," *IEEE Commun. Mag.*, vol. 33, pp. 50–56, Oct. 1995.

[6] B. Maglaris *et al*., "Performance models of statistical multiplexing in packet video communications," *IEEE Trans. Commun.*, vol. 36, pp. 834–843, July 1988.

[7] P. Sen *et al*., "Models for packet switching of variable bit rate video sources," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 865–869, June 1989.

[8] N. Nomura, T. Fujii, and N. Ohta, "Basic characteristics of variable rate video coding in ATM environment," *IEEE J. Select. Areas Commun.* vol. 7, pp. 752–760, June 1989.

[9] N. Ohta, *Packet Video: Modeling and Signal Processing.* Norwood, MA: Artech House, 1994.

[10] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 49–59, Mar. 1992.

[11] F. Yegenoglu, B. Jabbari, and Y. Q. Zhang, "Motion-classified autogressive modeling of variable bit rate video," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 3, pp. 42–53, Feb. 1993.

[12] J. L. C. Wu, Y. W. Chen, and K. C. Jiang, "Two models for variable bit rate MPEG sources," *IEICE Trans. Commun.*, vol. E78-b, pp. 717–745, May 1995.
[13] R. Grunenfelder *et al*., "Characterization of video codecs as autoregressive moving average processes and related queueing system performance," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 284–293, Apr. 1991.
[14] A. A. Tarraf, I. W. Habib, and T. N. Saadawi, "A novel neural network traffic enforcement mechanism for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 1089–1096, Aug. 1994.
[15] ———, "Intelligent traffic control for ATM broad-band networks," *IEEE Commun. Mag.*, vol. 33, pp. 76–82, Oct. 1995.
[16] J. Chong, S. Q. Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR over ATM," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 12–23, Jan. 1995.
[17] G. Kechriotis and E. Manolakos, "Using recurrent neural networks for nonlinear and chaotic signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1993, pp. 465–469.
[18] R. T. Williams and D. E. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.
[19] J. S. Meditch, *Stochastic Optimal Linear Estimation and Control*. New York: McGraw-Hill, 1969.
[20] G. E. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA: Holden-Day, 1976.
[21] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, pp. 240–254, Mar. 1994.
[22] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Processing*, vol. 43, pp. 526–535, Feb. 1995.
[23] C. Darken and J. Moody, "Toward faster stochastic gradient search," in *Neural Information Processing System 4*, J. E. Moody *et al*., Eds. San Mateo, CA: Morgan Kaufmann, pp. 1009–1016.
[24] L. Li and S. Haykin, "A cascaded recurrent neural networks for real-time nonlinear adaptive filtering," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1993, pp. 857–862.
[25] *IEEE Trans. Neural Networks* (Special Issue on Dynamic Recurrent Networks), vol. 5, Mar. 1994.
[26] L. K. Li, "Approximation theory and recurrent networks," in *Proc. Int. Joint Conf. Neural Networks*, vol. 2, Baltimore, MD, pp. 266–271.
[27] *IEEE Trans. Neural Networks* (Special Issue on Neural Network Hardware), vol. 4, May 1993.

Po-Rong Chang (M'87) received the B.S. degree in electrical engineering from the National Tsing-Hua University, Taiwan, R.O.C., in 1980, the M.S. degree in telecommunication engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1982, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1988.

From 1982 to 1984, he was a Lecturer in the Chinese Air Force Telecommunication and Electronics School for his two-year military service. From 1984 to 1985, he was an Instructor of Electrical Engineering at National Taiwan Institute of Technology, Taipei. From 1989 to 1990, he was a Project Leader in charge of the SPARC chip design team at ERSO of Industrial Technology and Research Institute, Chu-Tung, Taiwan. Currently, he is a Professor of Communication Engineering at National Chiao-Tung University. His current interests include fuzzy neural networks, wireless multimedia systems, and virtual reality.

Jen-Tsung Hu received the B.S. and M.S. degrees in communication engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1995 and 1996, respectively.

Currently, he is working at AT&T in the United States.