An Adaptable Neural-Network Model for Recursive Nonlinear Traffic Prediction and Modeling of MPEG Video Sources

Anastasios D. Doulamis, *Member, IEEE,* Nikolaos D. Doulamis, *Member, IEEE,* and Stefanos D. Kollias, *Member, IEEE*

***Abstract*—Multimedia services and especially digital video is expected to be the major traffic component transmitted over communication networks [such as internet protocol (IP)-based networks]. For this reason, traffic characterization and modeling of such services are required for an efficient network operation. The generated models can be used as traffic rate predictors, during the network operation phase (online traffic modeling), or as video generators for estimating the network resources, during the network design phase (offline traffic modeling). In this paper, an adaptable neural-network architecture is proposed covering both cases. The scheme is based on an efficient recursive weight estimation algorithm, which adapts the network response to current conditions. In particular, the algorithm updates the network weights so that 1) the network output, after the adaptation, is approximately equal to current bit rates (current traffic statistics) and 2) a minimal degradation over the obtained network knowledge is provided. It can be shown that the proposed adaptable neural-network architecture simulates a recursive nonlinear autoregressive model (RNAR) similar to the notation used in the linear case. The algorithm presents low computational complexity and high efficiency in tracking traffic rates in contrast to conventional retraining schemes. Furthermore, for the problem of offline traffic modeling, a novel correlation mechanism is proposed for capturing the burstness of the actual MPEG video traffic. The performance of the model is evaluated using several real-life MPEG coded video sources of long duration and compared with other linear/nonlinear techniques used for both cases. The results indicate that the proposed adaptable neural-network architecture presents better performance than other examined techniques.**

I. Introduction

THE demands of multimedia services and especially of digital video is expected to rapidly increase in the following years, due to the development of low-cost devices for capturing and generating multimedia information [1], [2]. Examples of such services include high-definition TV (HDTV), videophone or video conferencing applications, home education, video on demand services, content-based image/video retrieval from large databases and video browsing applications [3]–[8]. Since digital video demands large bandwidth requirements, several coding algorithms have been proposed in the literature to accomplish efficient video compression. Among the most popular is the MPEG standard [3], mainly due to its generic structure, able to support a broad range of applications [9]. However, even in compressed domain, the bandwidth requirements of digital video still remain high and, thus, its transmission, in a cost effective and quality guaranteed manner, is a difficult task. For this reason, appropriate traffic management schemes are developed so that efficient transmission video information over telecommunication networks, like the Internet, is accomplished in the sense that an acceptable quality of service is guaranteed to the users.

For implementing appropriate traffic management schemes, statistical characterization and modeling of the transmitted information is required. In general, two cases are discriminated. The first concerns the development of statistical models able to 1) capture traffic statistics; 2) simulate traffic behavior; and 3) estimate network resources with high accuracy. We call this process *offline traffic modeling* in the rest of this paper, since the generated models *do not* track the actual rates, but they are applied "*offline*" to simulate video traffic. Instead, the second case regards *the network operation* phase, where the models are applied to predict future rates based on previous *actual* samples of the traffic. The second case is called *online traffic modeling* in the rest of this paper, since the models are applied "*online*" during video transmission.

Many applications can benefit for offline–online traffic modeling. In the offline case, traffic models can be used as *video generators*, to select appropriate network parameters during the *network design phase*, such as utilization, and/or number of multiplexed sources that achieve an acceptable video quality. In this framework, the reliability of the network can be evaluated. For example, we can estimate the probability of refusing a new video call or the probability of network overload. On the other hand, online traffic models are very useful for traffic management algorithms and congestion control schemes, which prevent the network from possible overload. Video on demands services, video streaming over Internet Protocol (IP) networks, wireless transmission of video sources, telemedicine applications, home education, or interactive television are some typical examples of services which require such a kind of modeling.

Several video models have been proposed in the literature dealing with either the offline or the online case. As far as the offline modeling is concerned, the first attempts were by Haskell and Limb who proposed and simulated statistical multiplexing for picturephone encoders [10], [11]. A discrete-state continuous-time Markov chain was proposed in [12] for variable bit rate (VBR) teleconference streams, while in [13] a discrete autoregressive process of order 1 [DAR(1)] has been found more suitable for the same type of video sources,

Manuscript received March 12, 2001; revised May 29, 2002.

The authors are with the Department of Electrical and Computer Engineering, National Technical University of Athens, 15773 Athens, Greece (e-mail: adoulam@cs.ntua.gr).

Digital Object Identifier 10.1109/TNN.2002.806645

but of longer duration. For more complex video streams, more complicated models have been proposed. In [14], a motion classified autoregressive (AR) model has been presented, the parameters of which are determined using a Markov chain associated with different motion activity periods, in case of a full motion VBR video stream. An M-state discrete-time discrete-state Markov model has been proposed in [15], with an additional state to represent scene change. Scene modeling has been adopted in [16] for video films whose the autocorrelation functions present a long-range dependence and in [17] for VBR broadcast video traffic. However, the aforementioned models cannot be directly applied to MPEG coded video sources since different coding methods result in different traffic statistics [2]. Some statistical properties and basic characteristics of MPEG coded video steams have been recently analyzed, such as the higher average rate of Intra frames than of Inter ones or the periodicity existing in the autocorrelation function of MPEG sequences [18]–[21]. A multilayer Markovian modeling of MPEG-1 video sources followed by nonlinearities has been recently proposed in [22].

However, all the aforementioned models cannot be applied to the problem of online traffic modeling since they are oriented to capturing only traffic statistics. Several works have been proposed in the literature dealing with the problem of online traffic modeling using either linear or nonlinear models. Linear approaches are mainly implemented in a recursive framework and they are suitable for simple traffic traces [2], [23]. Instead, prediction of more complicated traffic, such as video streams, is based on nonlinear models implemented using neural networks [24], [25]. In particular, in [24], a feedforward neural network has been applied, the parameters of which (network weights) remain constant throughout transmission. As a result, the model is not suitable for statistically varying process, like the MPEG video traffic. In [25], a recurrent neural network has been used for predicting a "smooth" video traffic, such as videoconferencing sequences. Instead, in complicated traffic, where highly traffic rates are encountered, the model performance is deteriorated. Furthermore, they are not suitable for the problem of offline traffic modeling.

The main difficulty of modeling a nonlinear input–output relationship is the estimation of the unknown nonlinear function of the model. A simple way to perform this is to use a simplified mathematical model, such as functions of exponential type, and then to estimate the model parameters to fit the data. However, these approaches present satisfactory results only in case that the data follow the preassumed function type. Otherwise, a significant deterioration of the model accuracy is observed. Furthermore, in a complicated real-world environment it is difficulty to find a simple analytical mathematical model for describing the input–output relationship. MPEG video traffic lies in this category due to the complexity of the coding algorithm used to compressed video data and the complicated content of the stream, which may include several camera effects, such as zooming or panning or scenes of high activity. Neural networks provide a generic framework for modeling a nonlinear function at any accuracy by appropriately estimating network structure and parameters (weights) [26].

Models both for online and offline traffic modeling and prediction are presented in this paper, based on an adaptive neural-network architecture. The proposed scheme is based on an efficient recursive estimation of neural-network weights for adapting network output to current conditions. In particular, the weight updating is performed in an optimal way so that 1) the network response is approximately equal to current conditions (traffic rates) and 2) a minimal degradation over the previous network knowledge is accomplished. The proposed adaptive neural-network architecture simulates a recursive implementation of a nonlinear autoregressive model (RNAR), which is suitable for complex and nonstationary processes, such as the MPEG video traffic. In contrast to conventional neural-network training algorithms, where generally require long training periods, the computational complexity of the proposed scheme is very small and can be applied to real-time applications, such as the online traffic prediction of MPEG video sources. Furthermore, it guarantees that the network response is close to current traffic statistics, instead of conventional retraining methods where the weight updating process can be trapped to local minima, deteriorating the network performance.

The weight adaptation is performed at time instances, where the model response (network performance) is not satisfactory. These time instances are detected by an activation mechanism. Furthermore, in case of offline traffic modeling, a novel correlation mechanism is proposed so that the correlation among the three types of frames (Intraframe (I), Predictive (P), and Bi-directionally predictive (B)) of the MPEG stream is retained in the generated sequence. This is an important issue for modeling of MPEG video sources, since it affects the burstness of video traffic, which has a significant influence on the network resources, such as the frame losses. More specifically, if the rates of I, P, and B frames are generated independently, severe underestimate of the network resources will be accomplished since the burstness of the actual traffic cannot be appropriately estimated. Moreover, high frame rates, which mainly affect frame loss probabilities, are further refined based on a generalized regression neural network (GRNN) architecture. Experimental results and comparisons with other linear and nonlinear models both for traffic prediction and modeling are presented to show the good performance of the proposed scheme both as traffic rate predictor and network resource estimator.

This paper is organized as follows: Section II refers to the problem of online traffic modeling. In particular, in Section II-A, the basic characteristics of MPEG video sources are presented, while in Section II-B, a nonlinear autoregressive (NAR) model based on a neural-network architecture is described for predicting traffic rates. The weight adaptation algorithm used to update network performance to current conditions is discussed in Section III. The problem of offline traffic modeling is addressed in Sections IV and V. Experimental results and comparative study with other linear–nonlinear approaches using real life MPEG-2 coded video sources are presented in Section VI both for online and offline traffic modeling. Finally, Section VII concludes the paper.

II. Online Traffic Modeling

In this section, the problem of online traffic modeling is investigated. As mentioned above, online traffic modeling is

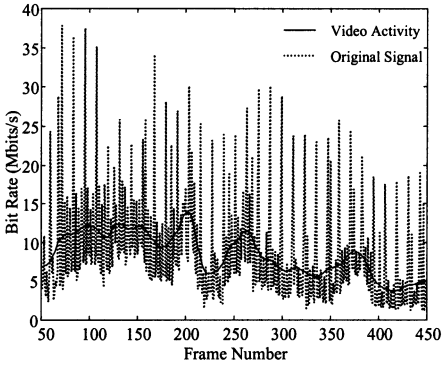

Fig. 1. Traffic rate of the Source1 sequence. The first 400 frames.

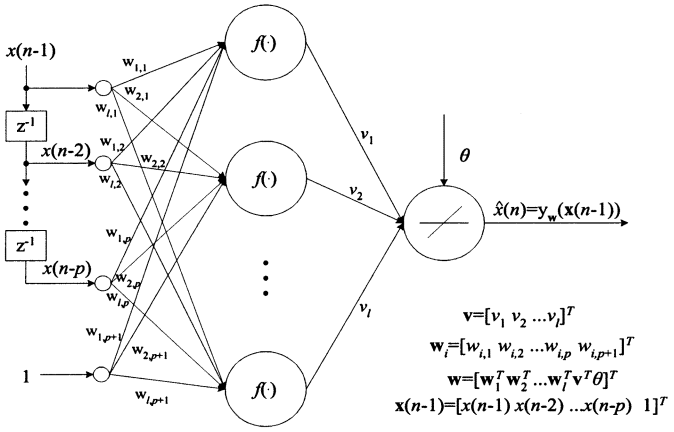

Fig. 2. Neural-network architecture.

useful for many applications, such as wireless video transmission, video streaming over IP networks or video on demand services. For example, in case that high video activity is expected, different scheduling algorithms can be applied to avoid network congestion. Since the adopted coding algorithm affects the statistical properties of video traffic, it is useful first to briefly describe the general structure of the MPEG standard.

A. Basic MPEG Source Characteristics

In the MPEG standard, three different coding modes are supported: (I), (P), and (B). In intraframe mode, only compression in spatial direction is performed, while in predictive mode (P frames), a motion compensation scheme is applied to reduce the temporal redundancy. B frames are coded similar to P frames apart from the fact that motion vectors are estimated with respect to the previous or the following (or an interpolation between them) I or P frame [8]. P and B frames are also called Inter frames. These three types of frames are deterministically merged, forming a group, group of picture (GOP), which is defined by the distance L between I frames and the distance M between P frames. In our case, $M = 3$ and $L = 12$ resulting in the following GOP pattern $\ldots IBBPBBPBBPBBI\ldots$. The dotted line of Fig. 1 depicts the frame rates of a video sequence (Jurassic Park) coded using the MPEG-2 standard over a time window of 400 samples. Since the three types of frames present different statistical properties, traffic modeling is separately performed for each type of frame [25], [28].

B. NARMs Based on Neural Networks

Let us denote as $x^c(n)$, the rate of $c \in \{I, P, B\}$-frame stream. It should be mentioned that variable n of $x^c(n)$ refers to the nth sample of c-stream and *not* to the nth sample of the aggregate sequence. Due to the MPEG coding algorithm, the frame rate of $x^c(n)$ depends on the previous p^c samples through a nonlinear relation. Intra and Inter frames are related to the previous frames due to the continuity of the video stream. Video content changes much slowly from frame to frame compared to the frame rate. In addition, Inter frames are related to each other due to motion estimation algorithm used for their encoding. Since, in real-life, MPEG-coded video sources many complicated effects are encountered, such as scene cuts, degradation of lighting conditions, camera zooming and panning and so on, the input–output relation is highly nonlinear [2], [9] and [20]. Therefore, the frame rates are modeled as a NARM of order p^c, denoted as NAR(p^c) similar to the notation used in the linear case. The input–output relation of an NAR(p^c) is given by the following equation:

$$x^c(n) = g^c(x^c(n-1), x^c(n-2), \ldots, x^c(n-p^c)) + e^c(n), \qquad c \in \{I, P, B\} \quad (1)$$

where $g^c(\cdot)$ is a nonlinear function and $e^c(n)$ an independent and identically distributed (i.i.d.) error with mean value of μ^c and standard deviation of b^c. In the following analysis, we omit superscript c for simplicity purposes since it is involved in all equations.

The main difficulty in implementing a NAR model is that function $g(\cdot)$ is actually unknown. However, in [27], it has been shown that a feedforward neural network, with a tapped delay line (TDL) filter as input, is able to implement a NAR model, within any acceptable accuracy. Fig. 2 illustrates the architecture of such a network, consisting of one hidden layer of l neurons, one output neuron and a TDL filter of p input elements, equal to the order of the model. Let us denote as $\mathbf{w}_i = [w_{i,1} \cdots w_{i,p+1}]^T$, $i = 1, 2, \ldots l$ the $(p+1) \times 1$ vectors containing all weights $w_{i,k}$, $k = 1, \ldots, p$ which connect the ith hidden neuron to the kth input element and $w_{i,p+1}$ the biases of the ith neuron. Let us also define as $\mathbf{v} = [v_1\ v_2\ \cdots\ v_l]^T$, an $l \times 1$ vector, which contains the network weights, say v_i, connecting the ith hidden neuron to the output neuron and as θ the respective bias. Then, vector $\mathbf{w} = [\mathbf{w}_1^T\ \mathbf{w}_2^T \cdots \mathbf{w}_l^T\ \mathbf{v}^T\ \theta]^T$ represents all network weights and biases. These weights and biases are also illustrated in Fig. 2 for clarity. In this case, the network output $y_{\mathbf{w}}(\cdot)$, which provides an estimate, say $\hat{x}(n)$ of $x(n)$, is given by

$$y_{\mathbf{w}}(\mathbf{x}(n-1)) \equiv \hat{x}(n) = \mathbf{v}^T \cdot \mathbf{u}(\mathbf{x}(n-1)) + \theta \quad (2a)$$

with

$$\mathbf{u}(\mathbf{x}(n-1)) = \begin{bmatrix} u_1(\mathbf{x}(n-1)) \\ \vdots \\ u_l(\mathbf{x}(n-1)) \end{bmatrix} = \begin{bmatrix} f(\mathbf{w}_1^T \cdot \mathbf{x}(n-1)) \\ \vdots \\ f(\mathbf{w}_l^T \cdot \mathbf{x}(n-1)) \end{bmatrix} = \mathbf{f}(\mathbf{W}^T \cdot \mathbf{x}(n-1)) \quad (2b)$$

where $\mathbf{W}$ is a $(p+1)\times l$ matrix, the columns of which correspond to the weight vector $\mathbf{w}_i$, that is $\mathbf{W}=[\mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_l]$ and $\mathbf{f}(\cdot)$ a vector-valued function, the elements of which correspond to the activation functions, say $f(\cdot)$, of hidden neurons. In our case, the sigmoid function is used as $f(\cdot)$. The

$$\mathbf{x}(n-1)=[x(n-1)\cdots x(n-p)\ 1]^T \tag{3}$$

is a $(p+1)\times 1$ input vector containing the p-previous samples $x(n-1),\ldots,x(n-p)$ plus a unity to accommodate the bias effect. In (2a) a linear activation function has been used for the output neuron, since the network output approximates a continuous valued signal, i.e., the frame rate of I, P, and B frames.

Initially, a training set of N samples is used to estimate the network weights $\mathbf{w}$. Without loss of generality, we can assume that the initial training set S_{init}, consists of, say, K pairs

$$S_{init}=\{(\mathbf{x}(p),\,x(p+1)),\ldots,(\mathbf{x}(K+p-1),\,x(K+p))\}. \tag{4}$$

The network is initially trained to minimize the mean squared value of the error for all samples in the training set S_{init}

$$\begin{aligned}C&=\tfrac{1}{2}\sum_{n=p+1}^{K+p}\{x(n)-\hat{x}(n)\}^2\\&=\tfrac{1}{2}\sum_{n=p+1}^{K+p}\{x(n)-y_{\mathbf{w}}(\mathbf{x}(n-1))\}^2.\end{aligned} \tag{5}$$

A second-order method has been used, in our case, for training the network based on the Marquardt–Levenberg algorithm. This method has been selected due to its efficiency and fast convergence, since it was designed to approach second-order training speed without having to compute the Hessian matrix. To further increase the generalization performance of the network, the cross validation method has also been applied [26].

III. Recursive Nonlinear Autoregressive Modeling

In the previous implementation, the model parameters, i.e., the network weights, are considered constant throughout video transmission. However, in dynamic environments, where the system characteristics change through time, this assumption deteriorates the prediction accuracy, since the model response cannot be adapted to current conditions [29]. This is the case of real-life MPEG coded video sources, where traffic statistics locally fluctuate according to video activity. To face the aforementioned difficulty, a novel recursive weight adaptation algorithm is proposed in this paper resulting in an adaptable neural-network architecture. In particular, the proposed scheme optimally updates network weights to current conditions as input–output data receive so that 1) the network response, after the adaptation, satisfies the current conditions as much as possible, while 2) a minimal degradation over the previous network knowledge is provided. The neural network of Fig. 2 enhanced by the optimal weight adaptation algorithm actually implements an RNAR model.

A. Weight Adaptation Algorithm

Let us denote by $\mathbf{w}_b$ the network weights *before* the adaptation. Let us assume that these weights have been estimated using a training set

$$S_b=\{(\mathbf{t}_1,\,d_1),\ldots,(\mathbf{t}_{N_b},\,d_{N_b})\} \tag{6}$$

of N_b pairs, which actually represent the previous network knowledge. Vectors $\mathbf{t}_i$ correspond to network inputs and have the form of (3), while d_i to the target outputs (i.e., is a specific frame rate). Similarly, let $\mathbf{w}_a$ denote the network weights *after* the adaptation. Without loss of generality, we can consider that the weight adaptation algorithm is activated at the kth sample $x(k)$. This means that the $(k+1)$th sample will be estimated using the new weights $\mathbf{w}_a$, while the $x(k)$ has been predicted based on the previous weights $\mathbf{w}_b$. Then, the new weights $\mathbf{w}_a$ are estimated by minimizing the following equation:

$$\mathbf{w}_a=\arg\min_{\mathbf{w}} D_b=\tfrac{1}{2}\sum_{i=1}^{N_b}(y_{\mathbf{w}}(\mathbf{t}_i)-d_i)^2=\tfrac{1}{2}\sum_{i=1}^{N_b}D_{i,b}. \tag{7a}$$

Subject to

$$y_{\mathbf{w}_a}(\mathbf{x}(k-1))\equiv\hat{x}(k)\approx x(k) \tag{7b}$$

where $y_{\mathbf{w}_a}(\cdot)$ is the network output using the new weights $\mathbf{w}_a$ and $D_{i,b}=(y_{\mathbf{w}}(\mathbf{t}_i)-d_i)^2$ the squared prediction error over the ith sample of S_b.

Equation (7a) and (7b) indicates that the new network weights $\mathbf{w}_a$ should be estimated so that the network output, after the adaptation, is approximately equal to the current traffic rate, i.e., $x(k)$ [(7b)], while simultaneously a minimal distortion over all samples of S_b is provided [see (7a)].

Assuming that a small weight perturbation is sufficient for satisfying (7a) and (7b), we have that

$$\mathbf{w}_a=\mathbf{w}_b+\Delta\mathbf{w} \tag{8}$$

where $\Delta\mathbf{w}$ represents small increments of network weights.

The effect of$\Delta\mathbf{w}$ in (7a) and (7b) can be expressed by following two theorems.

Theorem 1: The effect of the small weight perturbation $\Delta\mathbf{w}$ to the term of (7b) (current network knowledge) is given as a linear constraint of the form $b=\mathbf{a}^T\cdot\Delta\mathbf{w}$, where scalar b and vector $\mathbf{a}$ are expressed with respect to the previous weights $\mathbf{w}_b$.

The proof of this theorem is given in Appendix A.

Theorem 2: The effect of the small weight perturbation $\Delta\mathbf{w}$ to the term of (7a) (previous network knowledge) is provided by minimizing a squared convex function of the form $(1/2)\Delta\mathbf{w}^T\cdot\mathbf{J}^T\cdot\mathbf{J}\cdot\Delta\mathbf{w}$, where matrix $\mathbf{J}$ is expressed with respect to the previous weights $\mathbf{w}_b$.

The proof of Theorem 2 is given in Appendix B.

Based on the previous two theorems, we can conclude that (7a) and (7b)yields to the following constraint minimization:

minimize

$$E=\tfrac{1}{2}\Delta\mathbf{w}^T\cdot\mathbf{J}^T\cdot\mathbf{J}\cdot\Delta\mathbf{w} \tag{9a}$$

subject to

$$b=\mathbf{a}^T\cdot\Delta\mathbf{w}. \tag{9b}$$

The expression of Matrix $\mathbf{J}$, vector $\mathbf{a}$ and scalar b is found in Appendixes A and B.

TABLE I
MAIN STEPS OF THE PROPOSED WEIGHT ADAPTATION ALGORITHM

Summary of the Proposed Recursive Estimation of Network Weights

1. Estimate matrix **J** using the derivatives (B7-B9).
2. Estimate vector **a** using the equations (12) and (13).
3. Apply the reduced gradient method and find the optimal $\Delta\mathbf{w}$ which minimizes the cost function $E=\frac{1}{2}\Delta\mathbf{w}^T\cdot\mathbf{J}^T\cdot\mathbf{J}\cdot\Delta\mathbf{w}$ [equation (9a)] subject to the constraint $b=\mathbf{a}^T\cdot\Delta\mathbf{w}$ [equation (9b)].
4. Update the current network weights $\mathbf{w}_b$ as follows $\mathbf{w}_a=\mathbf{w}_b+\Delta\mathbf{w}$ [equation (8)].

Equation (9a) is a convex function since it is of square form. Furthermore, (9b) corresponds to a linear constraint. As a result, only one minimum exists, which is the global [30]. To minimize (9a) and (9b), the reduced gradient method has been adopted. Table I presents the main steps of the proposed weight adaptation algorithm.

B. Reduced Gradient Method

The reduced gradient method is an iterative process, which starts from a feasible point and moves in a direction, which decreases the error function of (9a), while simultaneously satisfies the constraint defined by the (9b). A point is called feasible if it satisfies the constraint of (9b). In our case, as initial feasible point, $\Delta\mathbf{w}(0)$, the minimal distance from the origin to the constraint hyper-surface $b-\mathbf{a}^T\cdot\Delta\underline{w}=0$ is used. Therefore, $\Delta\mathbf{w}(0)$ is given by the following equation:

$$\Delta\mathbf{w}(0)=\frac{b\cdot\mathbf{a}}{\mathbf{a}^T\cdot\mathbf{a}}. \tag{10}$$

This selection is a "good" feasible solution and permits the convergence of the algorithm within few iterations. This is very important for online traffic prediction applications where time is often crucial. It should be also noticed, calculation of $\Delta\mathbf{w}(0)$ requires low computational load since only an inner product is involved.

At the mth iteration of the algorithm, the feasible point $\Delta\mathbf{w}(m)$ is arbitrarily partitioned into groups; the first group contains the dependent (basic) variables, while the second the independent variables. Since, in our case, only one constraint is available, one element of $\Delta\mathbf{w}(m)$ is considered as dependent variable, while the remaining N_w-1 elements are considered as independent variables. The N_w indicates the number of all network weights. Without loss of generality, we select the first element of vector $\Delta\mathbf{w}(m)$ as dependent variable. Therefore

$$\Delta\mathbf{w}(m)=\left[\Delta w^D(m)\ \ \Delta\mathbf{w}^I(m)^T\right]^T \tag{11}$$

where $\Delta w^D(m)$ is a scalar, which corresponds to the dependent variable, while $\Delta\mathbf{w}^I(m)$ a $(N_w-1)\times 1$ vector which contains the independent variables.

Based on (11) and (9b), we can express the dependent variable $\Delta w^D(m)$ with respect to the independent variables $\Delta\mathbf{w}^I(m)$ as follows:

$$\Delta w^D(m+1)=\frac{b-(\mathbf{a}^I)^T\cdot\Delta\mathbf{w}^I(m+1)}{a^D}. \tag{12}$$

Scalar a^D is the first element of vector $\mathbf{a}$ of (9b), i.e., the element which corresponds to the dependent variable. On the other hand, vector $\mathbf{a}^I$ contains the remaining elements of $\mathbf{a}$, i.e., the elements of independent variables. Therefore, we have that

$$\mathbf{a}=\left[a^D\ \ \left(\mathbf{a}^I\right)^T\right]^T. \tag{13}$$

At next iterations, the independent variables are updated as follows:

$$\Delta\mathbf{w}^I(m+1)=\Delta\mathbf{w}^I(m)-\gamma(m)\mathbf{r}(m) \tag{14}$$

while the dependent variable $\Delta w^D(m)$ is provided by (12). Scalar $\gamma(m)$ regulates the convergence rate of the weight updating.

In (14), $\mathbf{r}(m)$ is the reduced gradient of cost function E of (9a), i.e., the gradient with respect to the independent variables $\Delta\mathbf{w}^I(m)$. The reduced gradient of cost function E is given as

$$\mathbf{r}(m)=\mathbf{d}-\frac{1}{a^D}c\cdot\mathbf{a}^I \tag{15}$$

where scalar $c$and vector $\mathbf{d}$ are provided by splitting the gradient of cost function E into the dependent and independent group

$$\nabla E=\mathbf{K}\cdot\Delta\mathbf{w}(m)=\left[c\ \ \mathbf{d}^T\right]^T. \tag{16}$$

The main steps of the reduced gradient method, which is used in our case for estimating the new network weights, are summarized in Table II.

C. Computational Complexity

The computational complexity of the proposed weight adaptation algorithm, in contradiction to the generally long training periods of neural networks, is very small. The recursive weight estimation algorithm includes two main phases; the initialization phase and the iteration phase. In the initialization phase, the main parameters of the algorithm are estimated. On the other hand, in the iteration phase, the weight updating is performed. Let us first examine the computational cost of the iteration phase. In this case, the main computational load is due to the multiplication of matrix $\mathbf{K}$ of size $N_w\times N_w$, by the vector $\Delta\mathbf{w}$, of size $N_w\times 1$ required for the estimation of the gradient of cost function E [see (16)]. We recall that N_w is the number of all network weights. This multiplication requires $O(N_w^2)$ operations. However, for a typical value of N_w (around 100 as explained in the Section VI), the product $\mathbf{K}\cdot\Delta\mathbf{w}(m)$ of (16) demands few msec to be executed (~5 ms on a PC 550 MHz). The computational costs of (12) and (15), which are also

TABLE II
ALGORITHMIC FORM OF THE REDUCED GRADIENT METHOD

Reduced Gradient Method

Initialization Phase

1. Find the initial feasible solution $\Delta\mathbf{w}(0) = \frac{b \cdot \mathbf{a}}{\mathbf{a}^T \cdot \mathbf{a}}$ using equation (14).
2. Set m=0

Iteration Phase

3. if $m > T_D / T_s$ then stop.
4. Calculate the gradient of (9a) using equation (20). Set the first element of the gradient as the scalar c and the remaining elements as vector $\mathbf{d}$.
5. Partition vector $\mathbf{a}$ into two groups as in equation (17)
6. Estimate the reduced gradient using equation (19)
7. Update the independent network weights as in equation (18)
8. Calculate the dependent weights using equation (16)
9. Set m=m+1
10. If the reduced gradient is zero $\mathbf{r}(m)$ =0, or $m > T_D / T_s$, then stop. Otherwise goes to step 4.

involved in the iteration phase, are negligible [of order $O(N_w)$] compared to the cost of product $\mathbf{K} \cdot \Delta\mathbf{w}(m)$.

In the initialization phase, scalar b, vectors $\mathbf{a}$ and $\Delta\mathbf{w}(0)$ and matrix $\mathbf{K}$ should be calculated. Matrix $\mathbf{K}$ is available before the activation of the weight adaptation, since its elements depend only on the previous network knowledge, which does not change between the previous and the current weight updating phase [see equations (B5)–(B9)]. Vector $\Delta\mathbf{w}(0)$ requires $O(N_w)$ operations [see (10)] to be executed, while vector $\mathbf{a}$ $O(l \cdot p)$ [see Appendixes A and B], where we recall that l is the number of hidden neurons and p the number of inputs elements. Since $O(l \cdot p) \approx O(N_w)$, the computational complexity for calculating vectors $\Delta\mathbf{w}(0)$ and $\mathbf{a}$ are very small compared to the cost of the iteration phase. Finally, scalar b, which expresses the prediction error at the sample where the weight adaptation algorithm is activated, is known before the weight updating process. As a result, the computational load for the initialization phase is much smaller than the cost required for the iteration phase.

The number of iterations m that the reduced gradient method requires to derive the optimal solution is in general small. The number is restricted by a maximum permitted time, say T_c. T_c is related to T_D, which expresses the time interval for two successive frames of the same type. For example, for one-step prediction of I frames, $T_D = 12 * 40 = 480$ ms for $L = 12$ and 40 ms interframe interval (PAL system), while for P and B frames is $T_D = 80$ ms and 40 ms, respectively. Time T_c should be smaller than T_D so that the prediction results can be used to control network parameters. Thus, optimization of (9a) and (9b) should be terminated within $T_c \leq T_D$ ms or equivalently the adaptation should be stopped earlier than T_c/T_s iterations, where T_s denotes the computational time for one iteration.

D. Activation of the Weight Adaptation Algorithm

While, in theory, the weight adaptation can be performed at every new incoming sample, in practice there is no reason to update network weights (model parameters) in case that future samples are predicted with high accuracy. A way for activating the weight adaptation process is based on the prediction error and is given by

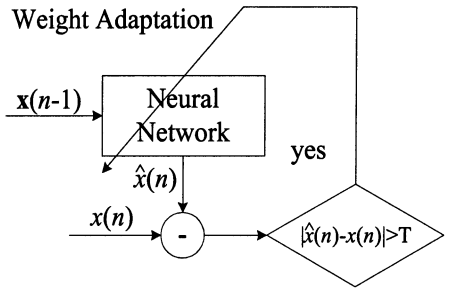

Fig. 3. Weight adaptation mechanism.

$$A = \begin{cases} 1, & \text{if } D = |\hat{x}(n) - x(n)| > T \\ 0, & \text{if } D = |\hat{x}(n) - x(n)| \leq T. \end{cases} \tag{17}$$

In case that difference D exceeds a certain threshold T, the network weights (model parameters) are updated using the aforementioned recursive algorithm $(A = 1)$. Otherwise, the same weights are used $(A = 0)$. The value of threshold T is calculated based on the average validation error, E_v, since this expresses the network performance during its operation phase. In particular, $T = \lambda * E_v$. A choice for scalar λ is around 1.05–1.1 meaning that the adaptation algorithm is activated when the current prediction error is 5%–10% higher than the average validation error E_v. A graphical representation of the activation mechanism is illustrated in Fig. 3.

IV. OFFLINE TRAFFIC MODELING

In this section, the problem of offline traffic modeling is investigated. Off-traffic modeling, is useful for simulating a communication network and thus selecting the appropriate resources and parameters to satisfy predetermined specifications. For example, we can estimate the frame loss probability in case of network congestion.

The neural-network structure which is described in Section II-B is used as statistical model for the offline traffic

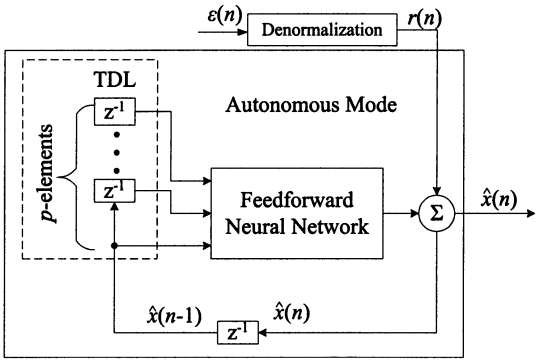

Fig. 4. Autonomous mode operation.

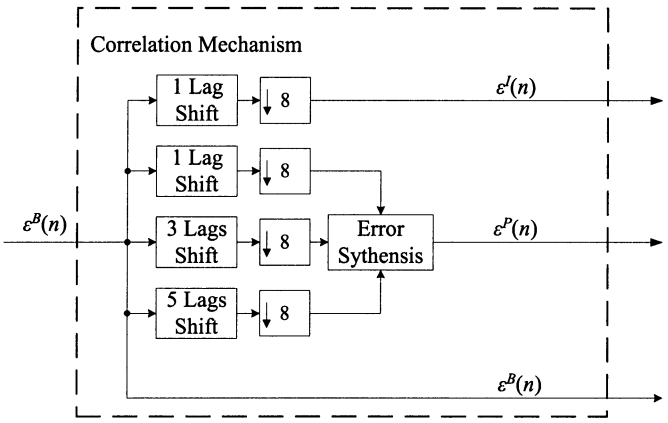

Fig. 5. Correlation mechanism for I, P, and B frame streams.

modeling. The only difference, in this case, is that the network inputs are the *estimated* instead of the actual data since the latter *are not* available. Thus, (1) is written as

$$\hat{x}(n) = y_{\mathbf{w}}\left(\hat{\mathbf{x}}(n-1)\right) + r(n) \tag{18}$$

where

$$\hat{\mathbf{x}}(n-1) = [\hat{x}(n-1)\cdots\hat{x}(n-p)\ 1]^T \tag{19}$$

is a vector containing the p-previous estimated rates plus a unity to accommodate the bias effect. The $\hat{\mathbf{x}}(n-1)$ is defined similarly to (3). In (18), we have also omitted the superscript c for simplicity purposes. The error $r(n)$ is an i.i.d. process and presents the same statistics as the error $e(n)$ of (1), i.e., it has the same mean value μ and standard deviation b as $e(n)$. The statistical model of (18) actually operates in a recursive autonomous mode (closed loop operation) [26], once the training procedure has been completed. Fig. 4 illustrates a graphical representation of the closed-loop operation. To start its recursive operation, only p initial samples are required. One common choice, for the p initials, is to be randomly selected as a sequence of p consecutive samples belonging to S_{init}. In order, however, the generated models to be accurate estimators of the network resources, two factors should be taken into account. 1) The error $r(n)$ should be appropriately modeled. 2) The rates of I, P, and B frames should be generated in correlation with each other.

A. Error Modeling

Since, by definition, the additive error of (18) is an i.i.d. process, the variables $r(n)$, for different n, represent statistically independent error samples that follow the same pdf. This error pdf can be estimated by the distribution of the difference between the actual data and the predicted ones using, for example, the estimated NAR model, over all samples of set S_{init}. Based on several experiments of VBR MPEG coded video sequences, it can be shown that a Gaussian distribution provides an accurate approximation of the error pdf. Similar conclusions have been also drawn for other VBR video streams, which have been coded using, however, different compression schemes [12]. Two parameters are required for determining the Gaussian pdf of $r(n)$; the mean value of μ and the variance of b^2. Since $r(n)$ presents the same statistics as $e(n)$ and using (1), it can be shown that

$$\mu = E\{x(n)\} - E\{\hat{x}(n)\} \tag{20a}$$

$$b^2 = E\left\{x(n)^2\right\} - 2\cdot E\left\{x(n)\hat{x}(n)\right\} + E\left\{\hat{x}(n)\hat{x}(n)\right\} - \mu^2 \tag{20b}$$

where $E\{\cdot\}$ is the expectation operator. Estimation of $E\{x(n)\}$, $E\{\hat{x}(n)\}$, $E\{x(n)^2\}$, $E\{x(n)\hat{x}(n)\}$, $E\{\hat{x}(n)\hat{x}(n)\}$ is provided using data from set S_{init}.

In the following analysis for convenience, we normalized the error $r(n)$ so that it has zero mean and variance equal to one. Thus, the normalized error, say $\varepsilon(n)$, is related to $r(n)$ by

$$\varepsilon(n) = (r(n) - \mu)/b. \tag{21}$$

B. MPEG Video Source Construction

The generated sequences for each c-stream by the neural-network model, $\{\hat{x}^c(n)\}$, are deterministically merged, according to the L and M values, to form the aggregate video sequence. However, if uncorrelated errors $r^c(n)$, or equivalently $\varepsilon^c(n)$, are used as filter inputs to signal $\hat{x}^c(n)$ [(18)], the aggregate MPEG sequence will contain uncorrelated I, P and B components, since $\hat{x}^c(n)$ are generated independently. However, there is correlation between the actual rates of I, P, and B frames mainly due to motion estimation algorithm and the continuity of the actual video traffic. As a result, independent generation of I, P, and B frames results in significant underestimate of the network resources, though the models follow the statistics of each c-stream, since they cannot capture the burstness of the actual video traffic [20], [21].

One way to correlate $\hat{x}^c(n)$, for different c, is to correlate their respective errors, due to the fact that they follow the same pdf. Particularly, a reference error is generated, following the Gauss distribution with zero mean and variance equal to one. Then, the errors of I, P, and B frames are generated with respect to this error. A simple approach is to consider as reference error, the $\varepsilon^B(n)$ due to the fact that B frames constitute the majority within a GOP. In this case, the errors of I and P frames are correlated with $\varepsilon^B(n)$ as illustrated in Fig. 5. Let us denote as N_B the number of B frames within a GOP period (in our case $N_B = 8$). Then, the normalized error of I frames, $\varepsilon^I(n)$ is related to $\varepsilon^B(n)$ through the following equation:

$$\varepsilon^I(n) = \varepsilon^B\left(N_B * n + 1\right) = \varepsilon^{B_2}(n) \tag{22}$$

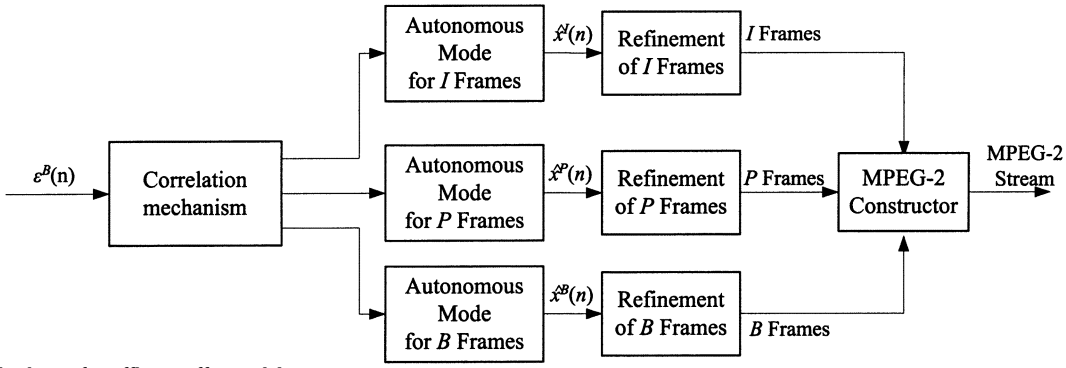

Fig. 6. Proposed scheme for offline traffic modeling.

where B_2 denotes the second B frame within a GOP and $\varepsilon^{B_2}(n)$ the respective error. Equation (22) indicates that I and B_2 frames are fed with the same errors. Then, the correlation of I frames with the other B frames is achieved through the equation (18). In Fig. 5, (22) is depicted using the decimator filter $\downarrow$ after shifting the reference error by one lag [the input–output is given by $y(n) = x(n) \downarrow M = x(n * M)$].

With a similar procedure, the error $\varepsilon^P(n)$ is created. In particular, let us denote as N_P the number of P frames within a GOP. In our case, where $L = 12$ and $M = 3$, $N_P = 3$. Then, $\varepsilon^P(n)$ is split into N_P error sequences, denoted by $\varepsilon^{P_i}(n)$, $i = 1, 2, \ldots, N_p$, each of which corresponds to the error of the ith P frame, say P_i, within a GOP. The error $\varepsilon^{P_i}(n)$ is related to the reference error $\varepsilon^B(n)$ as follows:

$$\varepsilon^{P_i}(n) = \varepsilon^B\left(N_B * n + \alpha\right)$$

with

$$\alpha = 1 \text{ for } i = 1, \quad \alpha = 3 \text{ for } i = 2 \quad \text{and} \quad \alpha = 5 \text{ for } i = 3 \tag{23}$$

where, without loss of generality, (23) has been expressed in the case of $N_P = 3$. Equation (23) indicates that correlation between P and B frames is achieved by generating P_1, P_2, and P_3 frames with the same errors as B_2, B_4, and B_6 frames. Correlation between P and I frames is indirectly achieved through (22) and (23). Equation (23) is also illustrated in Fig. 5 using the decimator filter $\downarrow$ and shifting the reference error $\varepsilon^B(n)$ by one (for P_1), three (for P_2) and five (for P_3) lags respectively. In this figure, the error synthesis module is responsible for merging the errors $\varepsilon^{P_i}(n)$ into a common link to generate the total error $\varepsilon^P(n)$.

V. Improvement of Offline Traffic Modeling Accuracy

The VBR MPEG video traffic presents highly fluctuated bit rates, especially for Inter frames in case of high motion periods, where the motion compensation algorithm usually fails [9]. However, the learning algorithm, used to train the neural network, is in fact based on a least squares fitting [minimization of (5)]. Consequently, if we imposed the network to track the abrupt changes of frame rates, belonging to the training set with high accuracy, this would give noisy results for data outside the training set (overfitting) by overemphasizing the abrupt rate changes [27]. Thus, underestimate of high frame rates appears which further leads to underestimate of network resources since high frame rates usually overload the network buffers. To increase the model accuracy, especially at high rates, the algorithm presented in the previous section is improved as follows.

Let us assume that the neural-network training has been completed, and thus the estimates $\hat{x}^c(n)$ of $x^c(n)$ are available. Let us also form the set I^c containing the time indexes of high traffic rates for the I, P, or B frames over data of set S_{init}, that is,

$$I^c = \{n \in \{p+1, p+2, \ldots, K+p\}\colon \hat{x}^c(n) > Q^c\} \quad \text{with } c \in \{I, P, B\} \tag{24}$$

where Q^c is an appropriate threshold expressed as a function of the mean value and standard deviation of signal $\hat{x}^c(n)$

$$Q^c = m^c + \eta^c \sigma^c \tag{25}$$

where m^c stands for the mean value of $\hat{x}^c(n)$ while σ^c for the standard deviation, estimating over all samples of the set S_{init}. Parameter η^c regulates how far from the mean value the threshold is. Small values of η^c indicate that almost all rates, which are greater than the respective average, are considered as high ones. On the contrary, large values of η^c imply that only few frames rates are regarded as high ones. Usually, the value η^c is chosen to be around 1.5 for all frame types.

Using the index set I^c, the sequences $\{\hat{x}^c(j)\}_{j \in I^c}$ and $\{x^c(j)\}_{j \in I^c}$ are created. Then, the pairs $(\hat{x}^c(j), x^c(j))$, for all $j \in I^c$ can be considered as points of the two-dimensional space, defining a nonlinear function $h(\cdot)$, which maps the estimated sample $\hat{x}^c(j)$ to the actual value $x^c(j)$. Estimation of the unknown function $h(\cdot)$ can be provided through a nonlinear least squares fitting based on a generalized regression neural network (GRNN) which is often used for function approximation [26].

Training a GRNN is straightforward. In particular, the neurons of the first layer are equal to the number of pairs $(\hat{x}^c(j), x^c(j))$, $j \in I^c$ with radial basis activation functions, while the respective weights are equal to the estimated data, $\hat{x}^c(j)$, $j \in I^c$. The second layer consists of one neuron, with linear activation function, whose respective weights are equal to the actual data $x^c(j) j \in I^c$. Then, the network output $\hat{x}_r^c(n)$, is the dot product of the first-layer neuron outputs and the weights of the second layer, after being normalized by the sum of first layer outputs. Fig. 6 illustrates the proposed scheme, incorporating all aforementioned stages for offline modeling as they are presented in Sections IV and V.

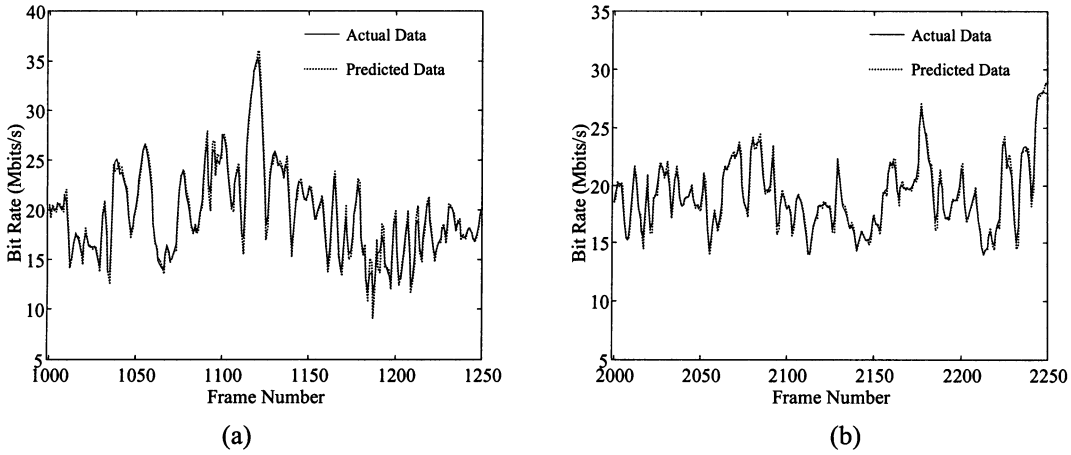

Fig. 7. Actual and predicted traffic rate using the proposed model over a time window of 250 frames for I frames. (a) Source3. (b) Source4.

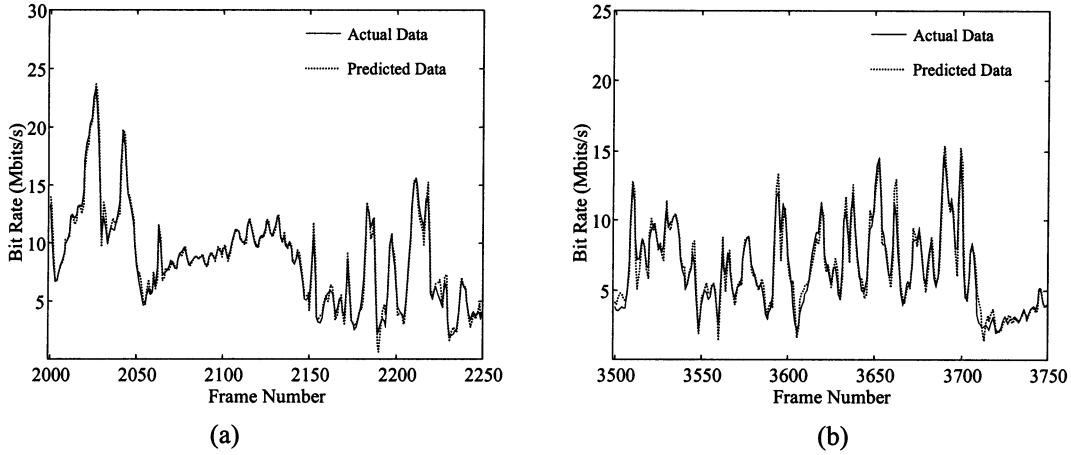

Fig. 8. Actual and predicted traffic rate using the proposed model over a time window of 250 frames for P frames. (a) Source3. (b) Source4.

VI. Experimental Results

In the following, the performance of the proposed model is evaluated both for online and offline traffic modeling. In our simulations, four long duration VBR MPEG-2 coded video sequences (each approximately of 45 min) have been used; two films and two TV series. For clarity of presentation, we call the first film and the first TV series as Source1 and Source2, while the second film and second TV series as Source3 and Source4 respectively, in the rest of this paper.

The 20% of data of Source1 and Source2 have been used to train the network; 75% of them are used as training data, while the rest 25% as validation data. The model accuracy is evaluated using data of Source3 and Source4, which are not included in the training set. The network inputs have been normalized so that they have zero mean and variance of one. The order of the model is selected to be 6, 9, 12 for the I, P, and B frame stream respectively. Furthermore, one hidden layer has been selected for the network with eight neurons.

A. Online Traffic Modeling

Fig. 7(a) and (b) illustrate the traffic rate of I frames (a time window of 250 frames) versus the frame number for Source3 and Source4, respectively. The solid line corresponds to the actual data, while the dotted line refers to the predicted data. In this case, the weight adaptation is performed each time a prediction error greater than 10% of the average validation error has been encountered. Similarly, Fig. 8 presents the traffic prediction for P frames of Source3 and Source4, while Fig. 9 for B frames of the same sequences. In all cases, the prediction accuracy is very high, even at time instances of highly fluctuated frame rates. An alternative way to indicate the good performance of the proposed model as traffic-rate predictor is to plot the predicted data versus the actual ones [31]. Fig. 11(a)–(c) present the results for the three types of frames (I, P and B) of Source3. In these figures, the solid line corresponds to perfect fit. It can be seen that the plotted data are close to the line of perfect fit, meaning that the proposed model is good predictor of traffic rates.

In some cases, however, it is useful to predict video activity, instead of the actual traffic. This is due to the fact that in a VBR transmission mode, periods of high-activity overload the network lines, while periods of low activity empty the network lines. The average frame rate over a GOP period, say $x^G(n)$, can be considered as an estimator of video activity for an MPEG stream and thus signal $x^G(n)$ is very useful for network management. The signal $x^G(n)$ results from the rate of the aggregate MPEG sequence $x^F(n)$ as

$$x^G(n) = \left(\sum_{i=0}^{L-1} x^F(n-i) \right) \Big/ L \tag{26}$$

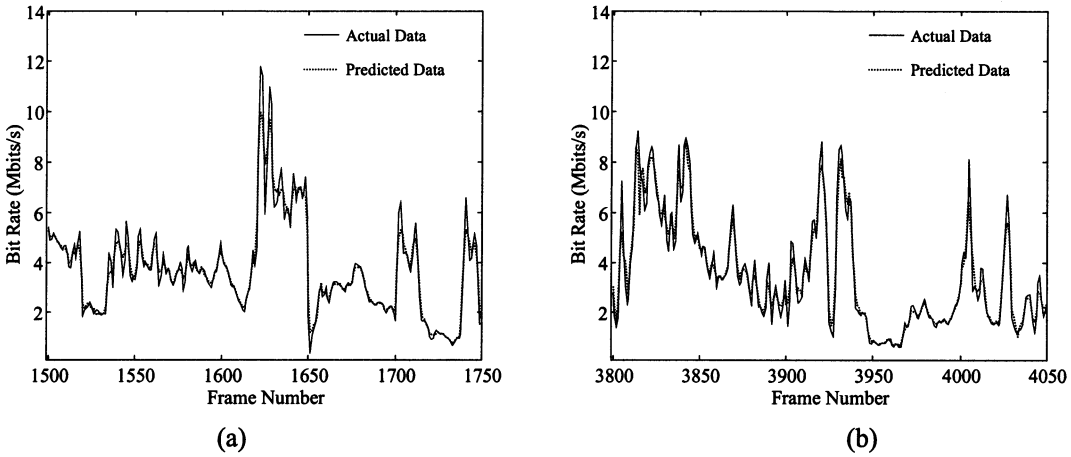

Fig. 9. Actual and predicted traffic rate using the proposed model over a time window of 250 frames for B frames. (a) Source3. (b) Source4.

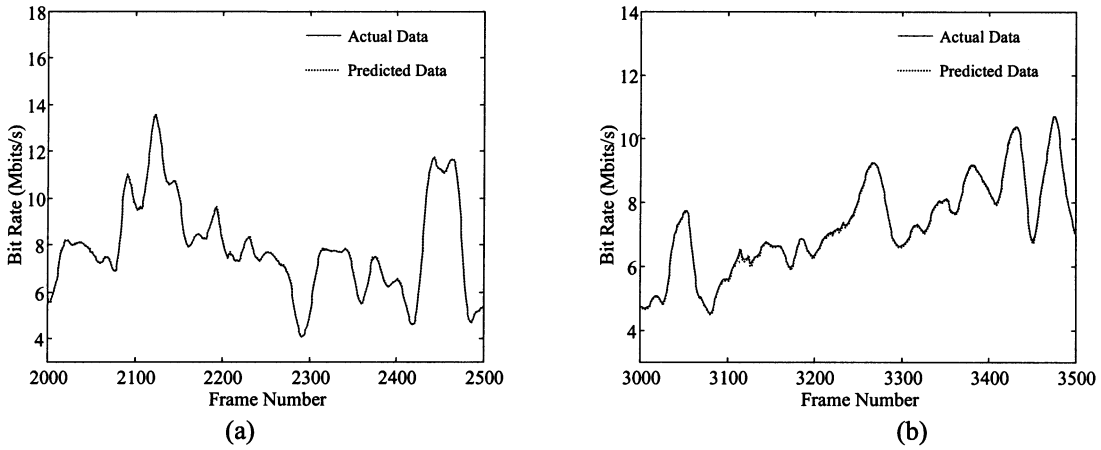

Fig. 10. Actual and predicted traffic rate using the proposed model over a time window of 250 frames for the video activity [$x^G(n)$ signal]. (a) Source3. (b) Source4.

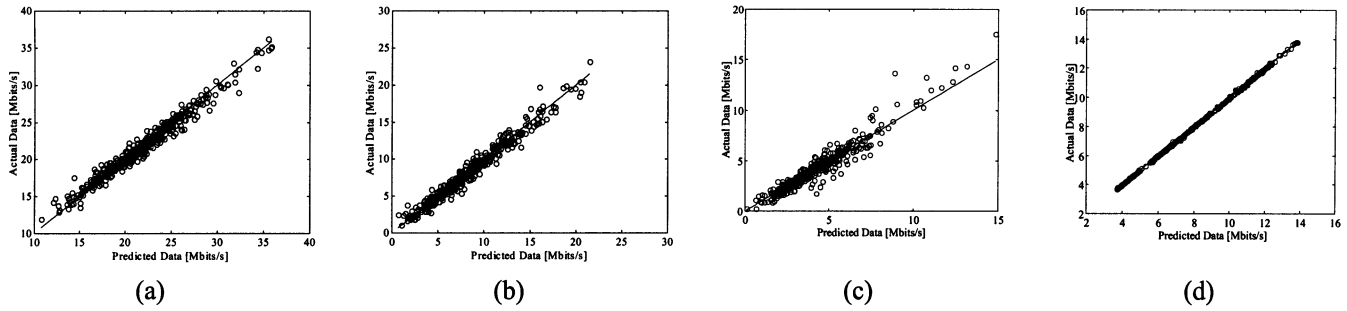

Fig. 11. Actual data versus the predicted ones for Source3. (a) I frames. (b) P frames. (c) B frames. (d) Video activity [$x^G(n)$ signal].

where signal $x^F(n)$ corresponds to the aggregate video traffic. The $x^F(n) = 0$, for $n < 0$. The solid line of Fig. 1 shows the respective samples of signal $x^G(n)$ for the sequence of Source1. As is observed, signal $x^G(n)$ is less bursty than the aggregate video traffic but follows video activity, in the sense that it increases (decreases) whenever the rates of Intra and Inter frames on average increase (decrease).

The prediction results of signal $x^G(n)$ are depicted in Fig. 10(a) and (b) for Source3 and Source4, respectively, over a time window of 500 frames, while in Fig. 11(d) the predicted traffic rates of $x^G(n)$ are plotted versus the actual rates. The highest prediction accuracy is observed for $x^G(n)$ since it is much smoother signal compared to I, P, and B frame streams.

An objective measure for evaluating the prediction accuracy is to compute the relative prediction error with respect to the actual data E^c,

$$E^c = \frac{1}{N^c} \sum_{n=1}^{N^c} \frac{|x^c(n) - \hat{x}^c(n)|}{x^c(n)} \times 100 \qquad (27)$$

where N^c is the total number of samples for the c-stream and $x^c(n)$, $\hat{x}^c(n)$ the nth sample of the actual and predicted signal.

TABLE III
RELATIVE PREDICTION ERROR

Sequences	Relative Prediction Error							
	Source3				Source4			
	I Frame (%)	P Frame (%)	B Frame (%)	$x^G(n)$ Signal (%)	I Frame (%)	P Frame (%)	B Frame (%)	$x^G(n)$ Signal (%)
The proposed Model	1.12	1.89	2.81	0.34	1.35	2.28	3.12	0.42
Recurrent NN	7.12	8.55	10.02	1.35	8.98	10.05	10.65	1.52
Feedforward NN	9.36	10.87	11.88	2.22	10.16	11.46	12.26	2.44
Linear Model	12.58	13.42	14.75	3.75	12.68	14.35	14.98	4.01

Table III presents the relative prediction error for the three frame streams over all samples of Source3 and Source4, using the proposed model. In this table, the relative prediction error for the signal $x^G(n)$ is also illustrated. As is observed the prediction performance is very satisfactory since in the worst case the relative error is less than 3.12%.

However, Intra frames are predicted more accurate than Inter frames, while signal $x^G(n)$ appears the highest prediction accuracy. This is due to the fact that Intra frames appears smaller fluctuation rate than Inter frames, while $x^G(n)$ is the smoothest signal, since it is generated as the average of all types of frames within a GOP period. The lower fluctuation rate of Intra frames than that of Inter ones is due to the coding algorithm. In particular, Intra frames are coded only in spatial direction while Inter frames both in spatial and the temporal one. For this reason, the average frame rate of Inter frames is smaller than of Intra ones. However, in cases of high video activity the motion estimation algorithm fails and Inter frames (P and B) are coded similarly to the Intra ones. Thus, the Inter frame's traffic rate is highly fluctuated compared to the Intra ones. Furthermore, B frames present the lowest average rate since their motion vectors are bidirectionally estimated with respect to the previous or the following (or an interpolation between them) I or P frame. This is verified in Figs. 7–9, where it is observed that the rate for I frames ranges from about 10 Mbits/s to 35 Mbits/s (3.5 times), while for P and B from 3 Mbits/s to 22 Mbits/s (7.33 times) and from 0.8 Mibts/s to 11 Mbits/s (13.75 times).

Due to the coding algorithm, I frames are generated without respect to any previous frame and thus, they present lower correlation compared to Inter frames. As a result, one would expect to present higher prediction error than that of Inter frames. However, the fluctuation rate of Inter frames is much higher than of intra ones, which mainly affect the prediction accuracy. Instead, within a simple scene with low video activity, Inter frames can be predicted more accurately verifying the previous observation.

As we have stated in Section III-C, the number of iterations for updating neural-network weights depends on a maximum permitted time T_c which should be smaller than time interval T_D between two successive frames of the same type so that the traffic management algorithms exploit the prediction results by regulating, for example, if necessary, network parameters. The effect of time T_c on the prediction accuracy E^c is illustrated in Fig. 12 for I, P, and B frames. As is observed, the prediction performance deteriorates as time T_c becomes shorter. B frames present the greatest variation and therefore present the

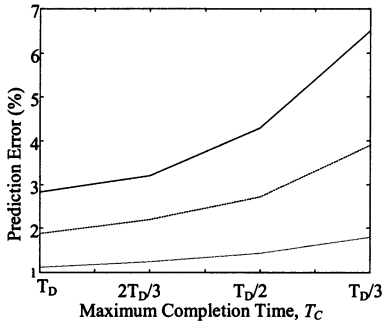

Fig. 12. Effect of the prediction performance with respect to the maximum completion time T_c.

main bottleneck for the online traffic prediction; a) they should be predicted in within a short time interval (less than 40 ms) and b) they are highly fluctuated and thus it is more difficult to be predicted. However, as presented in Fig. 12, even for B frames, the prediction performance remains satisfactory though a small number of iterations is allowed. This is due fact that the weight updating algorithm starts, according to equation (10), from an initial "good" feasible solution, which provides a minimal modification of the network weights, instead of a random one. Therefore, even within few iterations, a satisfactory solution is obtained.

1) Comparison With Other Linear and Nonlinear Techniques: In the following, the performance of the proposed model as traffic rate predictor is compared with three other methods. The first uses a recursive implementation of a linear AR model [23], the second a recurrent neural-network architecture as in [25] and finally the third a feedforward neural network without the weight updating mechanism [24]. Table III presents the results obtained for the I, P, B frame streams and signal $x^G(n)$, using the three aforementioned methods. As is observed, the proposed model provides the best prediction performance in all cases, while the recurrent neural-network approach [25] the second one. This is due to the fact that the first method uses a linear model (although its recursive implementation) to predict the MPEG video traffic, while the third does not update the network weights during prediction. To clearly illustrate the differences of the proposed method from the second best technique, i.e., the recurrent neural-network approach [25], a rate by rate comparison is depicted in Fig. 13(a)–(c) over a time window of the first 125 frames as in Figs. 7–9 in case of I, P and B frame stream of

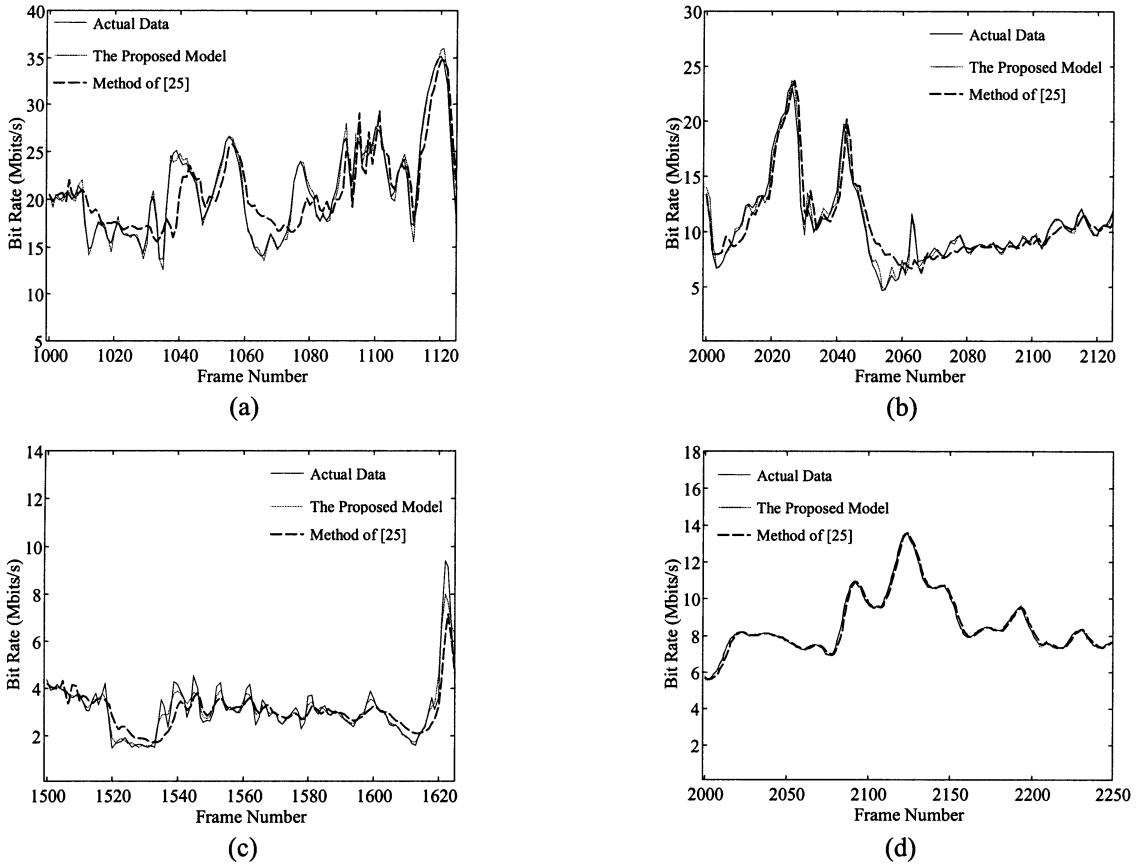

Fig. 13. Rate by rate comparison of the proposed model with the method of [25] over a time window of 125 frames of Source3 in case of (a) I frames, (b) P frames, (c) B frames, and (d) of signal $x^G(n)$.

Source3. Similarly, Fig. 13(d) presents the comparison between the proposed method and the recurrent one for signal $x^G(n)$ over the same time window as in Fig. 10.

A recurrent neural-network models an NARMA(p, q), where p is the order of the autoregressive term while q the order of the moving average component. Instead, the proposed model implements an adaptable feedforward neural-network architecture and thus simulates an NAR system in an recursive mode, denoted as RNAR. The MA component of a recurrent architecture cannot be removed by setting the order $q = 0$, since it stems from the feedback between the output and input [27]. The recurrent neural-network model cannot be efficiently track highly fluctuated rates as it happens in the examined VBR MPEG video sources. This is clearly noticed in Fig. 13. Particularly, we observed that the prediction accuracy of the $x^G(n)$ signal, which is the average over a GOP period, and thus presents the lowest fluctuation, is more accurate than that of Inter and Intra frames. Additionally, the highest prediction error is noticed for the B frames (see Table III) since they are highly fluctuated. Instead, the proposed adaptable neural-network model tracks well the traffic rates even at highly fluctuated rates, as shown in Table III. The adopted recursive algorithm trusts the current traffic data as much as possible meaning that the model is adapted to the current traffic statistics as indicated by the constraint [equation (9b)], while simultaneously provides a minimum degradation of the previous network knowledge. In contrast, the recurrent model adapts network weights at each sample toward the opposite direction of the current prediction error. Furthermore, the recurrent approach presents an unstable behavior especially when applied for long traffic periods.

B. Offline Traffic Modeling

In this section, the off line traffic modeling is evaluated using data of Source3 and Source4 which have been excluded from the training set.

Fig. 14 presents the frame loss probability, obtained from the proposed traffic model (dotted line), versus buffer size, along with those obtained by varying the rate of real data of Source3 $\pm 1\%$ for $N = 20$ multiplexed video sources. It should be mentioned that in this figure, the buffer size is expressed in time units (ms), which correspond to the maximum delay that the buffer causes. In Fig. 14, we observe that the traffic model provides a good approximation of frame loss probability at two different degrees of utilization, 75%, and 85%. In this figure, we also depict the frame loss probability obtained by applying two other methods. The first uses linear AR models for the I, P, and B frames. This is mentioned as "Linear Case" in the figure. In the second approach, frame loss probability is provided without implementing the algorithm, described in Section V (mentioned as "Model without GRNN" in the figure). As is observed, the linear case significantly underestimates frame loss probability, especially at large buffer size. Instead, the second method results in a slight underestimate of frame losses, since high traffic rates of I, P, and B frames cannot be estimated with high accuracy. Fig. 15 presents the performance of the proposed model for data of Source4 within a $\pm 1\%$ uncertainty, in case of $N = 20$ mul-

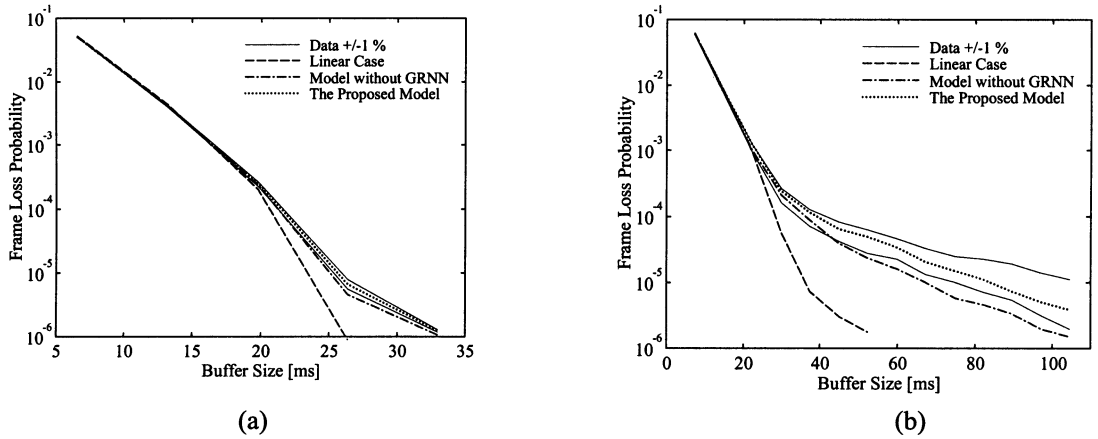

Fig. 14. Frame loss probability versus buffer size using data of Source3 in case of $N = 20$ sources. (a) $U = 75\%$. (b) $U = 85\%$.

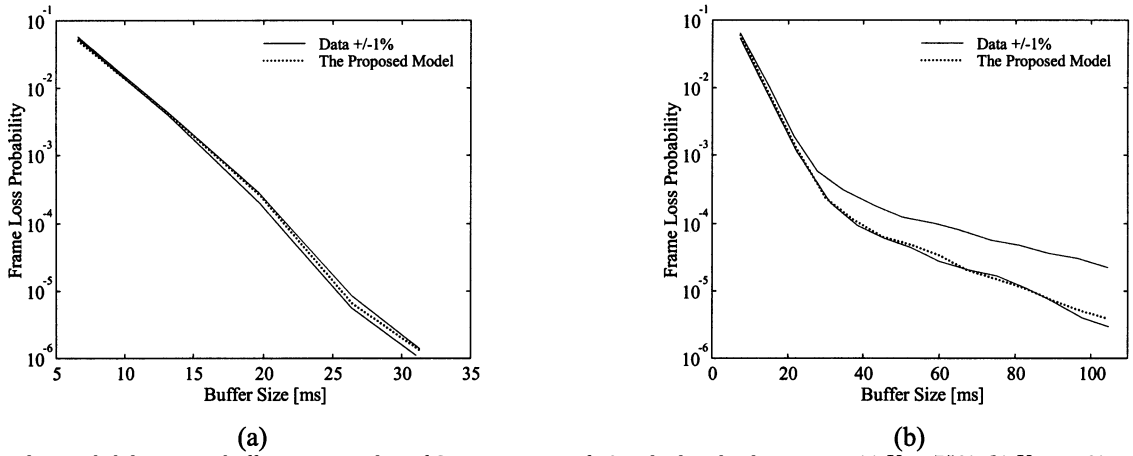

Fig. 15. Frame loss probability versus buffer size using data of Source4 in case of 20 multiplexed video sources. (a) $U = 75\%$. (b) $U = 85\%$.

TABLE IV
COMPARISON OF FRAME LOSSES FOR THE ACTUAL DATA OF SOURCE3 AND THE PROPOSED MODEL AT DIFFERENT BUFFER SIZES AND UTILIZATION DEGREES

Source	Starting Frame	Starting Time (ms)	Frame Loss Probabilities							
			44ms & U=85%		59ms & U=85%		29ms & U=80		35ms & U=80%	
			Data	Model	Data	Model	Data	Model	Data	Model
1	1	20.9286	-	-4.6	-	-	-	-	-	-
2	130	35.4815	-	-	-	-	-	-	-4.27	-4.13
3	10982	29.8838	-4.38	-4.06	-4.55	-4.3	-4.37	-4.55	-4.05	-3.90
4	1090	10.0323	-4.32	-4.58	-4.6	-	-4.56	-4.42	-	-
5	11656	12.7993	-3.62	-3.7	-3.92	-3.85	-3.90	-3.76	-2.93	-2.76
6	7117	33.0114	-4.14	-4.11	-4.39	-4.38	-4.07	-4.39	-3.27	-3.34
7	17595	36.0766	-4.57	-4.54	-	-	-4.6	-	-4.15	-4.17
8	6415	3.0171	-4.02	-4.28	-4.43	-4.6	-4.43	-4.56	-2.67	-2.87
9	22948	32.0337	-3.52	-3.95	-4.12	-4.38	-3.95	-3.65	-2.59	-2.68
10	31126	5.6047	-4.6	-4.52	-4.6	-4.53	-4.55	-4.6	-3.64	-3.56
11	9004	3.5876	-4.08	-3.99	-4.23	-4.48	-3.95	-4.15	-3.08	-3.43
12	14558	28.6451	-3.88	-4.11	-4.47	-4.42	-4.24	-4.27	-3.14	-3.2
13	3571	39.9001	-4.32	-4.16	-4.38	-4.38	-4.6	-	-4.05	-3.78
14	22879	9.6458	-	-4.6	-	-4.6	-	-	-4.07	-3.816
15	18492	28.3909	-4.14	-3.86	-4.06	-4.13	-4.43	-4.37	-3.06	-3.42

tiplexed sources. As can be seen, the loss rates provided by the model fall within the range of $\pm 1\%$ uncertainty for almost all delays.

Table IV depicts a source by source comparison of frame losses for the actual data of Source3 and the proposed model for delays, 44 ms and 59 ms, in case of $U = 85\%$ and 29 ms and 35 ms in case of $U = 80\%$. It is observed that the model approximates well not only the aggregate traffic intensity, but also the traffic characteristics of each of the first 15 individual multiplexed sources. In this table, the periodicity effect is also evident [13]. According to this phenomenon, although all multiplexed sources present the same statistical properties, they are

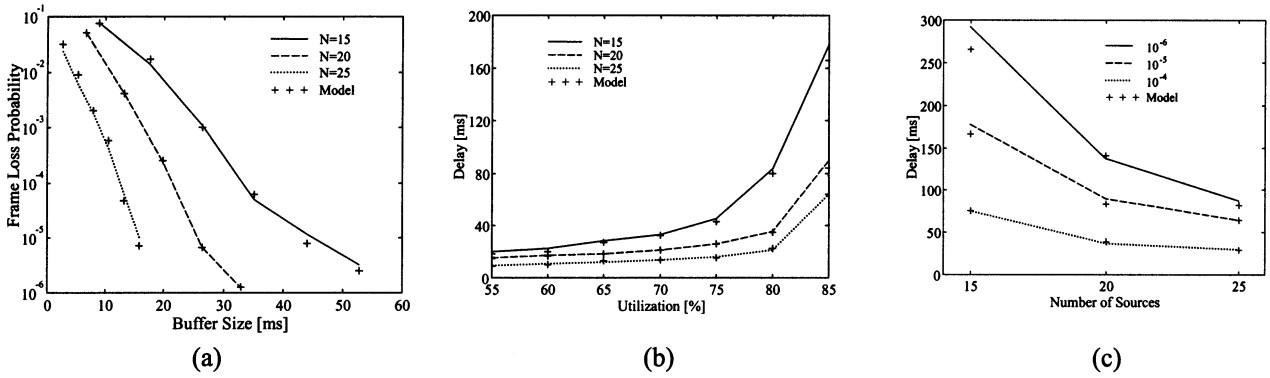

Fig. 16. (a) Frame loss probability versus buffer size of Source3 for $U = 75\%$. (b) Delay versus utilization for 10^{-5} frame losses. (c) Delay versus number of multiplexed for $U = 75\%$.

characterized by different loss rates. A video source that starts its transmission a short time after another source, will present much more losses since frames from this source are more likely to face larger queue lengths than frames arriving earlier from other sources.

The effect of the number of multiplexed sources on network resources is depicted in Fig. 16(a). In this figure, frame losses are plotted versus buffer size for three different number of multiplexed sources (15, 20, and 25) and utilization of 75% using data of Source3, along with those obtained by the proposed model. We observe that loss rates decay more rapidly as the number of multiplexed sources increase. Furthermore, the model provides a very good approximation of frame losses in all cases. The delay versus utilization for different number of multiplexed sources is depicted in Fig. 16(b) for the same video sequence, when the frame loss rate is equal to 10^{-5}. In this figure, we have depicted the delay provided by the model for the same loss rate. Fig. 16(c) shows the delay versus the number of multiplexed sources for three different loss rates (10^{-4}, 10^{-5} and 10^{-6}) in case of $U = 75\%$. In the same figure, data obtained from the proposed model are also depicted. From the previous results, we can see that delay and frame losses are regulated by utilization and number of multiplexed sources.

VII. Conclusion

The demands of multimedia services and especially of digital video rapidly increase in the recent years. Since video data demand large bandwidth requirements, even in compressed domain, traffic characterization and modeling of such kind of services is an important issue for efficient network operation. In this paper, an adaptive neural-network architecture is proposed for traffic prediction and modeling of MPEG coded video sources. The scheme is based on an efficient recursive weight estimation algorithm, which adapts the network response to current conditions. In particular, the weights are updated so that 1) the network output, after the adaptation, is approximately equal to the current data (traffic rates) and 2) a minimal degradation over the obtained network knowledge is provided. It has been shown that the proposed adaptive neural-network architecture simulates a RNAR model.

The proposed recursive weight adaptation is performed in an efficient and cost effective manner in contrast to other conventional training schemes (like the backpropagation), which usually induce high computational load. This is an important issue for online applications, such as traffic prediction of MPEG frame rates. The proposed adaptive neural-network scheme has been applied for traffic prediction (online modeling) and offline modeling of real-life MPEG coded video sources. In particular, for the problem of online traffic modeling, experimental results and comparative study with other linear and nonlinear traffic models have shown the superiority of the proposed scheme. For the problem of offline traffic modeling, the proposed algorithm have been tested in a wide range of utilizations, number of multiplexed sources and delay, and the results have shown that the scheme is very robust and provides better performance compared to traditional linear "offline" traffic models.

Other types of video coding schemes than the examined MPEG one can also be modeled by the proposed adaptive neural-network architecture. In this case, the model order and the learning set used to initially train the network should be changed. Instead, the proposed adaptive algorithm remains the same. It should be also mentioned that the process of applying three traffic models to each of three types of frames of the MPEG coding scheme (I, P, and B) is not valid for other coding schemes since in this case different frame types are presented.

Appendix A
Proof of Theorem 1

Let us define by $\mathbf{w}_{i,b}$, $\mathbf{w}_{i,a}$ the $(p+1)\times 1$ vectors containing the network weights and biases, which connect the ith hidden neuron to the input layer *before* and *after* the weight adaptation respectively. Then, matrices $\mathbf{W}_a$, $\mathbf{W}_b$ can be formed as follows:

$$\mathbf{W}_a = [\mathbf{w}_{1,a}\ \mathbf{w}_{2,a}\ \cdots\ \mathbf{w}_{l,a}]$$

and

$$\mathbf{W}_b = [\mathbf{w}_{1,b}\ \mathbf{w}_{2,b}\ \cdots\ \mathbf{w}_{l,b}] \tag{A1}$$

similarly to matrix $\mathbf{W}$ of Section II-B. Let us also define by $\mathbf{v}_b$, $\mathbf{v}_a$ the $l \times 1$ vectors which contain the network weights connecting the ith hidden neuron to the output neuron *before* and *after* the weight adaptation respectively. Similarly, θ_b, θ_b

correspond to the biases of the output neuron. Since (7a) and (7b) is valid for all network weights, it can be derived that

$$\mathbf{W}_a = \mathbf{W}_b + \Delta\mathbf{W}, \quad \mathbf{v}_a = \mathbf{v}_b + \Delta\mathbf{v}, \quad \theta_a = \theta_b + \Delta\theta \tag{A2}$$

where $\Delta\mathbf{W}$, $\Delta\mathbf{v}$, and $\Delta\theta$ are small increments of the respective network weights.

Thus, (3) can be written as

$$\mathbf{u}_a(\mathbf{x}(k-1)) = \mathbf{f}\left(\mathbf{W}_b^T \cdot \mathbf{x}(k-1) + \Delta\mathbf{W}^T \cdot \mathbf{x}(k-1)\right) \tag{A3}$$

where subscript b and a refer to *before* and *after* the weight adaptation, respectively.

Application of a first-order Taylor series expansion to (A3) yields to

$$\mathbf{u}_a(\mathbf{x}(k-1)) = \mathbf{f}\left(\mathbf{W}_b^T \cdot \mathbf{x}(k-1)\right) + \mathbf{Q}\cdot\Delta\mathbf{W}^T\cdot\mathbf{x}(k-1) \tag{A4}$$

where $\mathbf{Q}$ is the gradient of $\mathbf{f}(\cdot)$ and can be expressed by the following diagonal matrix

$$\mathbf{Q} = \operatorname{diag}\{\delta_{1,b}(\mathbf{x}(k-1)), \ldots, \delta_{l,b}(\mathbf{x}(k-1))\} \tag{A5}$$

where

$$\delta_{i,b}(\mathbf{x}(k-1)) = u_{i,b}(\mathbf{x}(k-1)) \cdot [1 - u_{i,b}(\mathbf{x}(k-1))] \tag{A6}$$

indicate the gradient of the hidden neuron outputs, assuming that the sigmoid are used as activation functions.

Since the network output is approximately equal to the current bit rate, i.e., $y_{\mathbf{w}_a}(\mathbf{x}(k-1)) \approx x(k)$, from (2a) and (A4) is expressed as follows:

$$x(k) = \mathbf{v}_b^T\cdot\mathbf{u}_a(\mathbf{x}(k-1)) + \Delta\mathbf{v}^T\cdot\mathbf{u}_a(\mathbf{x}(k-1)) + \theta_b + \Delta\theta. \tag{A7}$$

Combining, (A7) and (A4), and ignoring the second order terms, we can find that

$$x(k) - \mathbf{v}_b^T \cdot \mathbf{u}_b(\mathbf{x}(k-1)) - \theta_b = \mathbf{v}_b^T \cdot \mathbf{Q} \cdot \Delta\mathbf{W}^T \cdot\mathbf{x}(k-1) + \Delta\mathbf{v}^T \cdot \mathbf{u}_b(\mathbf{x}(k-1)) + \Delta\theta. \tag{A8}$$

Equation (A7) can be rewritten as

$$b = \mathbf{a}^T \cdot \Delta\mathbf{w} \tag{A9}$$

where

$$b \equiv x(k) - \mathbf{v}_b^T \cdot \mathbf{u}_b(\mathbf{x}(k-1)) - \theta_b = x(k) - \hat{x}(k) \tag{A10}$$

is the prediction error before the weight adaptation, while vector $\mathbf{a}$ is produced by reordering the right term of (A7) for all network weights

$$\mathbf{a}^T\cdot\Delta\mathbf{w} = \mathbf{v}_b^T\cdot\mathbf{Q}\cdot\Delta\mathbf{W}^T\cdot\mathbf{x}(k-1) + \Delta\mathbf{v}^T\cdot\mathbf{u}_b(\mathbf{x}(k-1)) + \Delta\theta. \tag{A11}$$

Equation (A11) is a linear equation with respect to weights increment $\Delta\mathbf{w}$ and vector $\mathbf{a}$ can be estimated by simply identifying the terms of the right and left hand of (A11). Particularly, we have that

$$\mathbf{a} = \left[\operatorname{vec}\left\{\mathbf{r}\cdot\mathbf{x}(k-1)^T\right\}\ \mathbf{u}_b(\mathbf{x}(k-1))\ 1\right]^T \tag{A12}$$

with $\mathbf{r} = \mathbf{Q}\cdot\mathbf{v}_b$ and $\operatorname{vec}\{\mathbf{r}\cdot\mathbf{x}(k-1)^T\}$ denoting a vector formed by stacking up all rows of matrix $\mathbf{r}\cdot\mathbf{x}(k-1)^T$.

Appendix B
Proof of Theorem 2

The effect of perturbation $\Delta\mathbf{w}$ in (6a) can be modeled by [32]

$$\sum_{i=1}^{N_b} \Delta D_{i,b}^2 = \mathbf{s}^T \cdot \mathbf{s} \tag{B1}$$

where $\Delta D_{i,b}$ is the sensitivity of the squared error of the ith element of S_b, $D_{i,b} = (d_i - y_{w_b}(\mathbf{t}_i))^2$ [(6a)] and $\mathbf{s} = [\Delta D_{1,b}\ \Delta D_{2,b} \cdots \Delta D_{N_b,b}]^T$ an $N_b \times 1$ vector containing the sensitivities for all elements of S_b. The sensitivity of errors $D_{i,b}$ can be expressed as

$$\Delta D_{i,b} = \sum_{j,k} \frac{\partial D_{i,b}}{\partial w_{j,k}} \Delta w_{j,k} + \sum_k \frac{\partial D_{i,b}}{\partial v_k} \Delta v_k + \frac{\partial D_{i,b}}{\partial \theta} \Delta\theta. \tag{B2}$$

Using (B2) for all elements in S_b, $i = 2, \ldots, N_b$, vector $\mathbf{s}$ can be written as

$$\mathbf{s} = \mathbf{J} \cdot \Delta\mathbf{w} \tag{B3}$$

where $\mathbf{J}$ is the Jacobian matrix of errors $D_{i,b}$ with respect to network weights

$$\mathbf{J} = \begin{bmatrix} \cdots & \dfrac{\partial D_{1,b}}{\partial w_{j,k}} & \cdots & \dfrac{\partial D_{1,b}}{\partial v_k} & \cdots & \dfrac{\partial D_{1,b}}{\partial \theta} \\ \cdots & \dfrac{\partial D_{2,b}}{\partial w_{j,k}} & \cdots & \dfrac{\partial D_{2,b}}{\partial v_k} & \cdots & \dfrac{\partial D_{2,b}}{\partial \theta} \\ \vdots & \vdots & \vdots & & \vdots & \\ \cdots & \dfrac{\partial D_{N_b,b}}{\partial w_{j,k}} & \cdots & \dfrac{\partial D_{N_b,b}}{\partial v_k} & \cdots & \dfrac{\partial D_{N_b,b}}{\partial \theta} \end{bmatrix}. \tag{B4}$$

The derivatives involved in (B4) are calculated as follows.

Let us first define as g_i the difference between the target output and the network output for the ith element of set S_b (old information) in case that the old weights are used

$$g_i = (d_i - y_{\mathbf{w}_b}(\mathbf{t}_i)). \tag{B5}$$

Let us also recall that

$$\delta_i(\mathbf{t}_j) = u_{i,b}(\mathbf{t}_j) \cdot (1 - u_{i,b}(\mathbf{t}_j)) \tag{B6}$$

is the derivative of the ith hidden neuron output when vector $\mathbf{t}_i$ is fed as input to the network using the old weights, $\mathbf{w}_b$. Dif-

ferentiating (6a) with respect to network weights $w_{j,k}$ we have that

$$\frac{\partial D_{i,b}}{\partial w_{j,k}} = -g_i \cdot v_j \cdot \delta_k(\mathbf{t}_i) \cdot t_{i,k} \tag{B7}$$

where $t_{i,k}$ is the kth element of vector $\mathbf{t}_i$. We recall that $w_{j,k}$ refers to the network weight that connects the jth hidden neuron with the kth input element. The equation (6a) with respect to network weights v_k and θ, we find that

$$\frac{\partial D_{i,b}}{\partial v_k} = -g_i \cdot u_k(\mathbf{t}_i) \tag{B8}$$

$$\frac{\partial D_{i,b}}{\partial \theta} = -g_i. \tag{B9}$$

Using (B1) and (B3), minimization of (6a) is equivalent to minimization of

$$\min \tfrac{1}{2}\, \Delta\mathbf{w}^T \cdot \mathbf{J}^T \cdot \mathbf{J} \cdot \Delta\mathbf{w}. \tag{B10}$$

REFERENCES

[1] M. de Prycker, *Asynchronous Transfer Mode. Solution for Broadband ISDN*. Antwerp, Belgium: Alcatel Bell, 1993.

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

[3] ISO/IEC 13 818-2, "Generic coding of moving pictures and associated audio,", H.262, Committee Draft, May 1994.

[4] N. Doulamis, A. Doulamis, D. Kalogeras, and S. Kollias, "Low bit rate coding of image sequences using adaptive regions of interest," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 928–934, Dec. 1998.

[5] S.-F. Chang, A. Eleftheriadis, and D. Anastassiou, "Development of Coloumbia's video on demand tested," *Signal Processing: Image Commun.*, vol. 8, pp. 191–208, 1994.

[6] Y. Avrithis, A. Doulamis, N. Doulamis, and S. Kollias, "A stochastic framework for optimal key frame extraction from MPEG video databases," *Comput. Vision Image Understand.*, vol. 75, no. 1/2, pp. 3–24, July/Aug. 1999.

[7] N. Doulamis, A. Doulamis, Y. Avrithis, K. Ntalianis, and S. Kollias, "Efficient summarization of stereoscopic video sequences," *IEEE Trans. Circuits Syst. Video Technol. (Special Issue on 3-D Video Processing)*, vol. 10, pp. 501–517, June 2000.

[8] M. Tekalp, *Digital Video Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[9] T. Sikora, *MPEG Digital Video Coding Standards*, E. R. Jurgens, Ed. New York: Digital Consumer Electronics Handbook, McGraw-Hill, 1997.

[10] B. G. Haskell, "Buffer and channel sharing by several interframe picturephone coders," *Bell Syst. Tech. J.*, vol. 51, pp. 261–289, 1972.

[11] J. O. Limb, "Buffering of data generated by the coding of moving images," *Bell Syst. Tech. J.*, pp. 239–255, 1972.

[12] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. D. Robbins, "Performance models of statistical multiplexing in packet video communication," *IEEE Trans. Commun.*, vol. 36, pp. 834–843, July 1988.

[13] D. 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.

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

[15] C. J. Hughes, M. Ghanbari, D. E. Pearson, V. Sefridis, and J. Xiong, "Modeling and subjective assessment of cell discard in ATM networks," *IEEE Trans. Image Processing*, vol. 2, pp. 212–222, Apr. 1993.

[16] M. R. Frater, J. F. Arnold, and P. Tan, "A new statistical model for traffic generated by VBR coders for television on the broadband ISDN," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 521–526, Dec. 1994.

[17] D. Heyman and T. V. Lakshman, "Sources models for VBR broadcast video traffic," *IEEE/ACM Trans. Networking*, vol. 4, pp. 40–48, 1996.

[18] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," in *Proc. 20th Conf Local Computer Networks*, Minneapolis, MN, Oct. 16–19, 1995, pp. 397–406.

[19] M. R. Grasse, J. F. Frater, and —PLEASE LIST FIRST INITIAL— Arnold, "Traffic characteristics of MPEG-2 variable bit rate video," in *Proc. Australian Telecommunication Networks Application Conf*, Dec. 1994, pp. 473–478.

[20] A. D. Doulamis, N. D. Doulamis, G. E. Konstantoulakis, and G. I. Stassinopoulos, "Traffic characterization and modeling of VBR coded MPEG sources," *IFIP ATM Networks*, vol. 3, pp. 60–80, 1997.

[21] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis of MPEG-2 coded VBR video traffic," in *Packet Video'94*, B2.1.

[22] N. Doulamis, A. Doulamis, G. Konstantoulakis, and G. Stasinopoulos, "Efficient modeling of VBR MPEG-1 video sources," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 93–112, Feb. 2000.

[23] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[24] S. 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. 12–33, Jan. 1995.

[25] P.-R. Chang and J.-T. Hu, "Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks," *IEEE J. Select. Areas Commun.*, vol. 15, Aug. 1997.

[26] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[27] J. Connor, D. Martin, and L. Altas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, pp. 240–254, Mar. 1994.

[28] W. Xu and A. G. Qureshi, "Adaptive linear prediction of MPEF video traffic," in *Proc. IEEE Int. Symp. Signal Processing and Its Applications (ISSPA)*, vol. 1, 1999, pp. 67–70.

[29] A. Doulamis, N. Doulamis, and S. Kollias, "On line retrainable neural networks: Improving the performance of neural networks in image analysis problems," *IEEE Trans. Neural Networks*, vol. 11, pp. 134–155, Jan. 2000.

[30] D. J. Luenberger, *Linear and Non Linear Programming*. Reading, MA: Addison-Wesley, 1984.

[31] H. Kobayashi, *Modeling and Analysis*. Reading, MA: Addison-Wesley, 1981.

[32] D. C. Park, M. A. EL-Sharkawi, and R. J. Marks, II, "An adaptively trained neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 334–345, May 1991.

Anastasios D. Doulamis (M'00) received the diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA) in 1995 with the highest honor and the Ph.D. degree in electrical and computer engineering from NTUA in 2000. His Ph.D. dissertation was supported by the Bodosakis Foundation Scholarship.

From 1996 to 2001, he was with the IVML (Image, Video, and Multimedia) Lab of the NTUA as a Research Assistant. From 2001 to 2002, he served his mandatory duty in the Greek army in the computer center department of the Hellenic Air Force. Since 2002, he is also a Senior Researcher in the Communication Laboratory of the NTUA. His research interest includes neural networks for multimedia services, semantic object extraction, interactive multimedia, and content-based image retrieval. He has published more than 100 papers, both in journals and international conferences.

Dr. Doulamis has received several awards and prizes, such as the NTUA medal, Best Greek engineer, and best engineer student both from the Greek government and the National Technical Chamber. He was Chairman of technical program committee of the VLBV'01 workshop in 2000. He has also served as program committee in several international conferences and workshops. He is reviewer of all major IEEE journals and conferences in his area, as well as and other leading international journals. He is a Member of the Greek Technical Chamber.

Nikolaos D. Doulamis (M'00) received the diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA) in 1995 with the highest honors and the Ph.D. degree in electrical and computer engineering from NTUA in 2000.

In 1996, he joined the image-processing lab of the NTUA. From 2001 to 2002, he served his mandatory duty in the Greek army in the central department of computer center of the Hellenic Air Force. From 2002, he is also a Senior Researcher in the Communication Laboratory of the NTUA. His Ph.D. dissertation was supported by the Bodosakis Foundation Scholarship. His research interest include video transmission, content-based image retrieval, summarization of video sequences and intelligent techniques for video processing.

Dr. Doulamis was awarded as the Best Greek Student in the field of engineering in national level by the Technical chamber of Greece in 1995. In 1996, he received the Best Graduate Thesis Award in the area of electrical engineering with A. Doulamis. During his studies, he has also received several prizes and awards from the National Technical University of Athens, the National Scholarship Foundation and the Technical Chamber of Greece. In 1997, he was given the NTUA Medal as Best Young Engineer. In 2000, he served as Chairman of technical program committee of the VLBV'01 workshop, while he has also served as program committee in several international conferences and workshops. In 2000, he was given the Thomaidion Foundation Best Journal Paper Award in conjunction with A. Doulamis. He is a Reviewer of IEEE journals and conferences as well as other leading international journals. He is an IEEE Member and member of the Greek Technical Chamber.

Stefanos D. Kollias (M'81) was born in Athens, Greece, in 1956. He received the diploma degree in electrical engineering from the National Technical University of Athens (NTUA) in 1979, the M.Sc. degree in communication engineering from the University of Manchester (UMIST), Manchester, U.K., in 1980, and the Ph.D. degree in signal processing from the Computer Science Division of NTUA in 1984.

In 1982, he received a ComSoc Scholarship from the IEEE Communications Society. Since 1986, he has been with the NTUA, where he is currently a Professor. From 1987 to 1988, he was a Visiting Research Scientist in the Department of Electrical Engineering and the Center for Telecommunications Research, Columbia University, New York. Current research interests include image processing and analysis, neural networks, image and video coding, multimedia systems. He is the author of more than 250 papers in the aforementioned areas.