

# Neural Network Modeling of a Class of ON-OFF Source Models with Self-Similar Characteristics

Homayoun Yousefi'zadeh  
 Center for Pervasive Communications  
 Electrical and Computer Engineering Department  
 University of California, Irvine  
 Irvine, CA 92697  
 hyousefi@uci.edu

**Abstract**— The significant characteristic of bursty traffic is self-similarity. Self-similarity is the main reason for observing the burst within burst patterns across a wide range of time scales. This is one of the unique characteristics of nonlinear systems with fractal nature. Perceptron neural networks, because of their nonlinear nature and simple method of learning, provide a powerful tool to model such kind of traffic patterns. In this paper, we introduce a novel scheme for the modeling task of source- and aggregated-level self-similar traffic patterns relying on the prediction power of perceptron neural networks. We also present and discuss some of the experimental findings. **Index Terms**— Perceptron Neural Networks, Back Propagation Algorithm, Packet Network, Bursty Traffic, ON-OFF Source, Self-Similarity, Traffic Modeling.

## I. INTRODUCTION

**T**ELETRAFFIC analysis of computer communication networks is one of the most important applications of mathematical modeling and queuing theory. This is mostly because of the widespread deployment of packet switching, specifically, services from Ethernet LANs, Variable Bit Rate (VBR) video, ATM, and ISDN. Modeling of bursty traffic patterns is among the most challenging problems in teletraffic analysis. Although, numerous models of packet arrival processes were proposed by Ramaswami et al. [8], Hellstern et al. [9], Sriram et al. [10], Heffes et al. [11], it seems that there is still a number of packet traffic features not being understood perfectly. This is partly due to uncertainties in the traffic characteristics and to the difficulties in characterizing the traffic arrival models. Adas [17] provided a survey of different teletraffic models in his paper. Analysis of traffic data from networks and services

such as Ethernet LANs [7], Variable Bit Rate (VBR) video [12], ISDN traffic [9], and Common Channel Signaling Network (CCNS) [14], have all convincingly demonstrated the presence of features such as self-similarity, long range dependence, slowly decaying variances, heavy-tailed distributions and fractal dimensions which are among the characteristics of fractal processes. Leland and Wilson from Bellcore research center presented a statistical analysis of Ethernet traffic, on the presence of "burstiness" across a wide range of time scales [7] in which traffic spikes ride on the longer term ripples, that in turn ride on longer term swells, so on and so forth. This phenomenon is explained in terms of self-similarity, i.e., self-similar phenomena show structural similarities across all or a very wide range of time scales. This burst within burst structure not only captures the fractal properties observed in actual traffic data but also explains why measurements show no actual burst length for the packet traffic pattern despite prediction of conventional models.

Chaos is a phenomenon observed in nonlinear dynamical systems and may be described as a situation in which a low order system is able to exhibit a very complicated behavior. For this reason, chaos used to be called deterministic noise. Since the trajectories of chaotic systems are mostly fractals, they may be used as very suitable generators of fractals. From the modeling point of view, the challenge is to capture the complexity of bursty traffic pattern with a small number of parameters of a chaotic map. Erramilli et al [3] used a number of simple nonlinear maps in order to capture some of the real traffic patterns characteristics. Giovanardi et al. [15] used self-similar

chaos-based traffic patterns in an analytical study of queuing systems. Alkhatib et al. [16] used chaos theory in modeling and forecasting VBR video patterns. Neural networks are a class of nonlinear systems capable of learning and performing tasks accomplished by other systems. Their broad range of applications includes speech and signal processing, pattern recognition, system modeling, and servo mechanism control. The various kinds of neural networks, generally, have energy functions. The learning procedure of neural networks is, indeed, nothing more than decreasing these energy functions until reaching local minimum levels. Neural networks acquire the required information from the examples supplied to them in their learning procedure. Systems with neural network building blocks are robust in the sense that the occurrence of small errors in the systems does not interfere with the proper operation of the system. This characteristic of the neural networks makes them quite suitable for our traffic modeling task.

The main idea of using neural networks as a powerful teletraffic modeling tool was originally extracted from our previous research topic in which we modeled a class of chaotic maps using perceptron neural networks and back propagation learning algorithm as described in [1], and [2]. Combining that idea with the bursty traffic chaotic modeling proposal of Ermilli et al [3] led us to introduce the current research topic. The objective of this paper is, hence, to introduce the application of neural networks in bursty teletraffic patterns modeling.

An outline of the paper follows. In section [2], we briefly review the characteristics of aggregated traffic patterns with self-similar nature. In section [3], we briefly review perceptron neural networks and the back propagation learning algorithm. In section [4], we explain how to model source- and aggregated-level bursty traffic patterns with neural networks. Finally, we summarize a number of numerical issues in section [5].

## II. AGGREGATED BURSTY TRAFFIC

The main objective of the current section is to provide an analytical framework for self-similarity as a statistical property of time series. Intuitively, self-similar phenomena display structural similarities across a significant number of time scales. The degree of self-similarity is sometimes specified by

measuring a single parameter called Hurst parameter. In the following section, we provide a brief discussion about the mathematical and statistical properties of the self-similar processes.

### A. Second-Order Self-Similarity

Suppose  $X = (X_t : t = 0, 1, 2, \dots)$  is a covariance stationary stochastic process with mean  $\mu$ , variance  $\sigma^2$ , and autocorrelation function  $R(n)$ ,  $n \geq 0$ . Particularly, assume the autocorrelation function of  $X$  has the form

$$R(n) \sim k_1 n^{-\beta}, \quad \text{as } n \rightarrow \infty \quad (1)$$

where  $0 < \beta < 1$  and constants  $k_1, k_2, \dots$  are finite positive integers. For each  $m = 1, 2, 3, \dots$  let  $X^{(m)} = (X_n^{(m)} : n = 1, 2, 3, \dots)$  be the covariance stationary time series with corresponding autocorrelation function  $R^{(m)}$  obtained from averaging the original series  $X$  over the non-overlapping time periods of size  $m$ , i.e., for each  $m = 1, 2, 3, \dots$ ,  $X^{(m)}$  is given by

$$X_n^{(m)} = \frac{1}{m}(X_{nm-m+1} + \dots + X_{nm}), \quad n \geq 1 \quad (2)$$

The process  $X$  is called exactly second-order self-similar with the self-similarity parameter  $H = 1 - \beta/2$  if the corresponding  $X^{(m)}$  has the same correlation function as  $X$ , i.e.,  $R^{(m)}(n) = R(n)$  for all  $m = 1, 2, 3, \dots$ , and  $n = 1, 2, 3, \dots$ .  $X$  is called asymptotically second-order self-similar with self-similarity parameter  $H = 1 - \beta/2$  if  $R^{(m)}(n)$  asymptotically approaches to  $R(n)$  given by 1, for large  $m$  and  $n$ . Hence, if the correlation functions of the aggregated processes  $X^{(m)}$  are the same as the correlation functions of  $X$  or approach asymptotically to the correlation functions of  $X$ , then  $X$  is called exactly or asymptotically second-order self-similar.

Fractal Gaussian Noise (FGN) is a good example of an exactly self-similar process with self-similarity parameter  $H$ ,  $1/2 < H < 1$ . Fractional Arima processes with the parameters  $(p, d, q)$  such that  $0 < d < 1/2$  are examples of asymptotically second-order self-similar processes with self-similarity parameter  $d + 1/2$ .

Mathematically, self-similarity manifests itself in a number of equivalent ways as follows.

- The variance of sample mean decreases more slowly than the reciprocal of the sample size.

This is called slowly decaying variance property which means  $var(X^{(m)}) \sim k_2 m^{(-\beta)}$  as  $m \rightarrow \infty$  with  $0 < \beta < 1$ .

- The autocorrelations decay hyperbolically rather than exponentially fast, implying a non-summable autocorrelation function  $\sum_n R(n) = \infty$ . This is called long range dependence property.
- The spectral density  $f(\cdot)$  obeys a power-law near the origin. This is the concept of  $1/f$  noise with the meaning  $f(\lambda) = k_3 \lambda^{-\gamma}$  as  $\lambda \rightarrow \infty$  with  $0 < \gamma < 1$  and  $\gamma = 1 - \beta$ .

It appears that the most important feature of the self-similar processes is that their aggregated process  $X^{(m)}$  possess a non-degenerate correlation function as  $m \rightarrow \infty$ . This is completely different from typical packet traffic models previously considered in literature, all of which have the property that their aggregated processes  $X^{(m)}$  tend to second order pure noise, i.e.,  $R^{(m)} \rightarrow 0$  as  $m \rightarrow \infty$ .

The concept of self-similar processes provides a very elegant explanation for the Hurst effect phenomenon. In order to describe the Hurst effect, we should first describe the rescaled adjusted range. For a given set of observations  $(X_n : n = 1, 2, \dots, N)$  with sample mean  $\bar{X}(N)$  and sample variance  $S^2(N)$ , the rescaled adjusted range denoted by the  $R/S$  statistic is given by

$$\frac{R(N)}{S(N)} = \frac{1}{S(N)} [max(W_i) - min(W_i)] \quad (3)$$

where  $i = 0, \dots, N$ ,  $W_0 = 0$  and

$$W_n = (X_1 + \dots + X_n) - n\bar{X}(N), \quad n \geq 1 \quad (4)$$

While many time series appear to be well represented by the relation  $E[R(N)/S(N)] \sim k_4 N^H$ , as  $N \rightarrow \infty$ , with Hurst parameter  $H$  typically about 0.73, observations  $X_n$  from a short-range dependent models are known to satisfy  $E[R(N)/S(N)] \sim k_5 N^{0.5}$ , as  $N \rightarrow \infty$ . This is usually referred to as the Hurst effect.

### B. An Evidence of Self-Similarity: Ethernet and VBR Video Traffic Measurements

In [4], Leland et al. from Bellcore research center observed the absence of natural length of a burst for the high quality, high time-resolution LAN traffic data collected between August 1989 and February 1992 on several Ethernet LANs. This behavior

is very different from conventional telephone traffic and from previously considered formal models of packet traffic. In [8], Beran et al. observed the same feature while studying VBR video traffic. With the available data sets, Leland et al. investigated the persistent feature of Ethernet traffic across the network and across the time irrespective of the medium utilization level. They graphically estimated a Hurst parameter  $H$  of about 0.80. In general, the degree of self-similarity depends on the utilization level of the medium. For the Ethernet it increases as the utilization increases. See [2] for further details.

## III. PERCEPTRON NEURAL NETWORKS AND BACK PROPAGATION ALGORITHM

Perceptron neural networks and their learning algorithm back propagation algorithm (BPA) have been studied extensively in the literature [18], [19],[20], [21], [22], [23], [24], [25]. In this section, we investigate perceptron neural networks and back propagation learning algorithm within the context of our modeling task.

### A. Perceptron Neural Networks

In an artificial neural network, the unit similar to the neuron is called processing element (PE). An artificial neural network contains a number of PEs. A PE has a large number of input paths and combines them by a simple weighted addition. Usually, the combined input is further processed by a transfer function. This transfer function has the form of a continuous function from the combined input signal.

In the network we are using, the output path of each PE is connected to the input paths of other PEs by a number of weighting functions. Each of the individual input signals to a PE is adjusted by these weighting functions before combining with other input signals in the transfer function. PEs are arranged in layers. Our network contains five layers with complete connections between consecutive layers. The input layer takes the data from the outside environment, the output layer returns the data to the outside environment, and the other layers known as hidden layers process the data inside the network. Figure 1 shows a simple perceptron network. There are two phases in the operation of a neural network. The first phase called learning procedure is, indeed, nothing but the adjustment of weighting functions such that

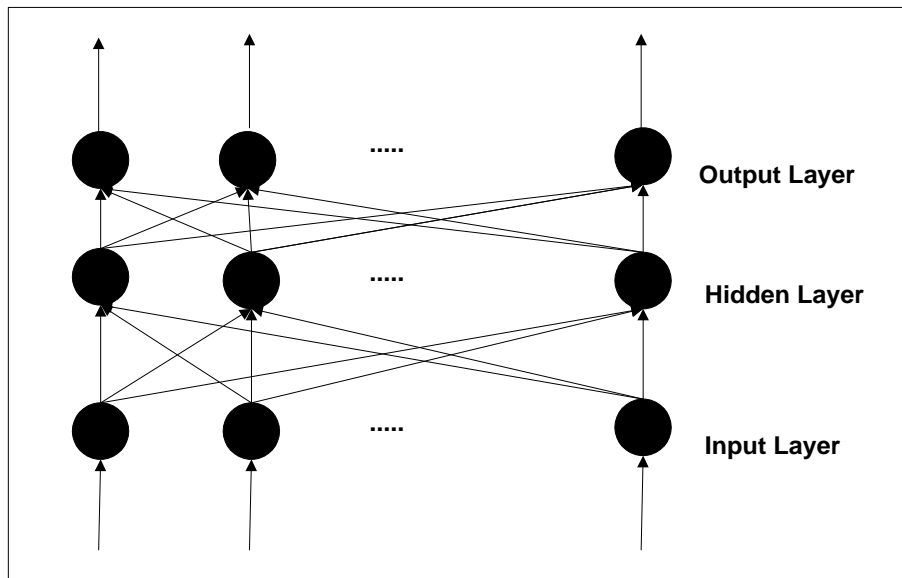


Fig. 1. A simple neural network with one hidden layer.

the network can respond suitably to the input stimulation. The number of samples required for training depends on the traffic pattern that the network is attempting to model. There are cases in which the network needs several thousands, or millions of samples to be trained. The second phase called recalling phase is a situation in which the network is able to create suitable response after the appearance of a specific stimulation. This phase may be part of the learning phase as it is the case when the network output must be compared to a desired output in order to create an error signal.

A simple network does not have any feedback connection between two different layers or a layer with itself. Such kind of network is called a feedforward network. In case of the fixed structure neural network that we are using, the input data from the input layer appears in the output layer via the interface of hidden layers, and hence it is called a feedforward network. Feedforward networks are generally considered because of their nonlinear properties. The data flow in feedback networks is more complicated. The network operates as a synchronous network as all of its PEs send their outputs at the same time. Each layer is considered to have a normalized operation because the output of the layer is adjusted at a fixed level. As the result, every PE of a layer has a criterion about the total output of the layer and can adjust its output correspondingly.

### B. Back Propagation Algorithm

Generally, a neural network must learn how to classify input patterns. It has been experimentally observed that, as the number of layers of a network increases, it can classify more and more complicated patterns. A significant problem is how a network can determine the error between its output and the desired output. The network, then, overcomes the mismatch between desired and actual output by adjusting the weightings of interconnections. This is called Credit Assignment (CA). The back propagation algorithm (BPA), originally introduced by Minsky and Papert [18], solves the CA problem by using all of the PEs and adjusting their total interconnections. It does so by propagating the output layer error to the preceding layer via the existing connections. This operation is then repeated until reaching the input layer. In other words, output error moves from from the output layer -just opposite the direction of the movement of original information- one layer at a time until reaching the input layer.

The classical form of a back propagation network consists of one input layer, one output layer, and one or two hidden layers. Although there is no limit on the number of hidden layers theoretically, it has been shown that it is possible to solve very complicated problems with four hidden layers. In a back propagation network, each layer is fully connected to the next layer. In the learning phase, information may come back through the network in order to update the weighting functions. The network may also be

hetero-associative (if the desired output(s) is(are) different from the input(s)) or auto-associative (if the desired output(s) is(are) the same as the input(s)). We use the following notation for the purpose of describing the back propagation algorithm. Please see [1] for further explanation.

- $x_j[s]$  : The present output state of the  $j$ -th neuron from the layer  $s$
- $w_{ji}[s]$  : Weighting function of the connection between  $i$ -th neuron from layer  $(s - 1)$  and the  $j$ -th neuron from layer  $s$
- $I_j[s]$  : The combined input of the  $j$ -th neuron of layer  $s$

Hence a PE in a back propagation network transfers its output as

$$x_j[s] = f\left\{\sum_i (w_{ji}[s].x_i[s - 1])\right\} = f\{I_j[s]\} \quad (5)$$

where  $f$  may be every continuous function. We use sigmoid function for our application as a continuous version of the step function. It is defined as

$$f(z) = \frac{1}{1 + e^{-z}} \quad (6)$$

Suppose that the network has an absolute error function differentiable with respect to all of the weighting functions. Then the critical parameter that is back propagated to the network is:

$$e_j[s] = -\frac{\partial E}{\partial I_j[s]} \quad (7)$$

where  $E$  is the absolute error function. The relationship between the relative error of a specified PE in layer  $s$  and the local errors in layer  $s + 1$  may be introduced as

$$e_j[s] = f'\{I_j[s]\} \cdot \sum_k \{e_k[s + 1].w_{kj}[s + 1]\} \quad (8)$$

Note that in relation 8 there must be a layer above the layer  $s$ , and hence it is not possible to use this relation for the output layer. As the derivative of the sigmoid function  $f$  can be expressed in terms of itself

$$f'\{I_j[s]\} = f(I_j[s]).\{1 - f(I_j[s])\} \quad (9)$$

relation 8 may be rewritten as:

$$e_j[s] = x_j[s].(1 - x_j[s]) \cdot \sum_k \{e_k[s + 1].w_{kj}[s + 1]\} \quad (10)$$

Hence the algorithm is described as follows. First, information propagates from the input layer to the output layer. Then, the error between the desired output and the network output propagates from the output layer to the input layer in the return path.

The objective of the learning procedure is to minimize the absolute error function. In this section, we introduce the error minimizing procedure based on the concept of local error. Assume that the weighting functions of the network are given in the form of  $w_{ji}[s]$ . In order to decrease the absolute error function, one may change the weighting functions in the opposite direction of the gradient vector as

$$\Delta w_{ji}[s] = -lc \frac{\partial E}{\partial w_{ji}[s]} \quad (11)$$

where  $lc$  denotes the learning coefficient. In relation 11 each of the weighting functions vary according to the magnitude and the opposite direction of gradient vector on the error surface. The variation of each weighting function is then calculated as:

$$\Delta w_{ji}[s] = lc.e_j[s].x_i[s - 1] \quad (12)$$

The performance of standard back propagation algorithm can be significantly improved by inserting a momentum term, using derivatives correction, and injecting the information obtained from the preceding layers to the current layer. Reference [1] includes further details for the enhanced version of the back propagation algorithm described below with the notation  $o$  indicating the present output of the network to the input  $i$ ,  $d$  corresponding desired output, index  $k$  denoting various elements of  $d$  and  $o$ , and  $E$  being the absolute error function defined as

$$E = \frac{1}{2} \sum_k (d_k - o_k)^2 \quad (13)$$

- Propagate the input  $i$  in the forward direction through the network until reaching to the output  $o$ . During propagating this information through the network all of the combined inputs  $I_j$  and output states  $x_j$  for each PE are set.
- For each PE in the output layer calculate the scaled local error and the variations of weighting functions from relations

$$\begin{aligned} e_o &= -\frac{\partial E}{\partial I_o} = -\frac{\partial E}{\partial y} \frac{\partial y}{\partial I_o} = (y_d - y).f'(I_o) \\ &= (y_d - y).y.(1 - y) \end{aligned} \quad (14)$$

$$\underbrace{\Delta w_{ji}[s]}_{(k+1)\text{-th step}} = lc.e_j[s].\{x_i[s-1] + k.e_i[s-1]\} + M(\underbrace{\Delta w_{ji}[s]}_{k\text{-th step}}) \quad (15)$$

where M stands for the momentum. The variations of  $k$  coefficient show the greatest impact on the adjustment algorithm of the weighting functions. The suitable value for this parameter again differs from case to case but in the most of the cases a value on the interval  $[0, 1]$  has led to have a pleasing result.

- For each PE in layer  $s$  located below the output layer and above the input layer obtain the scaled relative error and the variation in the weighting functions from relations

$$e_j[s] = x_j[s].(1-x_j[s]). \sum_k \{e_k[s+1].w_{kj}[s+1]\} \quad (16)$$

$$\underbrace{\Delta w_{ji}[s]}_{(k+1)\text{-th step}} = lc.e_j[s].\{x_i[s-1] + k.e_i[s-1]\} + M(\underbrace{\Delta w_{ji}[s]}_{k\text{-th step}}) \quad (17)$$

- Update all of the weighting functions by adding the variations to the old values.

#### IV. SOURCE- AND AGGREGATED-LEVEL NEURAL NETWORK MODELING OF BURSTY TRAFFIC

Earlier in this paper we mentioned that analysis of traffic data from networks and services have demonstrated that many data sources produce statistically self-similar data. We also mentioned that only a few number of the formal models of packet traffic considered in the literature are able to capture the self-similar nature of the measured traffic. The following lists a number of implications of the existence of self-similar packet traffic on high-speed networks.

- The Hurst parameter and fractal dimensions such as correlation dimension provide a more satisfactory measure of burstiness for self-similar traffic than the previously used measures such as the index of dispersion of counts.
- The nature of congestion produced by self-similar network traffic models differs from that

predicted by standard models. More specifically, the efficiency of the proposed congestion control schemes for high-speed networks greatly depend on how well those schemes perform under the influence of self-similar traffic scenarios.

In this section, a new approach capable of dealing with the fractal properties of the aggregated traffic is introduced. This approach provides a very elegant solution for self-similar traffic modeling and has the advantage of simplicity compare to the previously proposed approaches namely stochastic and deterministic chaotic map modeling. It is motivated by the desire for having a relatively simple model of the complex packet traffic generation process. As oppose to the above mentioned modeling approaches, it does not introduce a parameter that describes the fractal nature of traffic and hence need not cope with the complexity of estimating Hurst parameter and/or fractal dimensions. The approach simply takes advantage of using a fixed structure nonlinear system that is able to predict a bursty traffic pattern after getting trained by accessing to a number of the samples of the pattern. The number of traffic samples required for the training procedure depends on the network load and in general the complexity of the network dynamics. We believe the usage of neural networks provides a much simpler approach for the task of modeling because it works based on indirect learning of the network dynamics by using the information available in a number of samples.

The fixed structure neural network used for our modeling task consists of an input layer with up to eight neurons, three or four hidden layers with twenty neurons in each layer, and an output layer with one neuron. The inputs of the network are eight consecutive samples of the traffic pattern and the output of the network is the ninth sample which is supposed to be predicted. Based on the richness of the dynamic, it might be possible to reduce either the number of the neurons in the input layer or the number of hidden layers but as the standard structure, we use the above mentioned structure unless otherwise is stated. Figure 2 shows the structure of the network.

##### A. Modeling Individual Source Traffic Patterns

In the following, three different approaches based on the type of the input samples used for the training of the neural network are introduced. We use a fixed

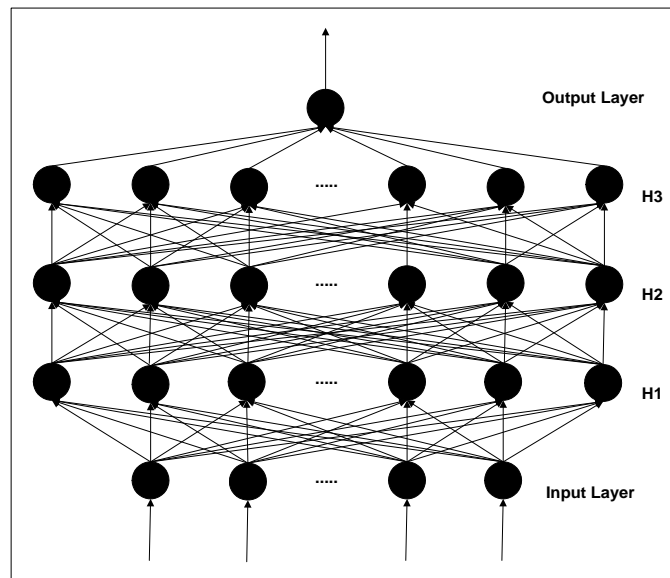


Fig. 2. Fixed structure neural network used for the modeling task.

structure neural network to model artificially generated traffic patterns by chaotic maps, namely the intermittency family of mappings. In [3], the authors provide a detailed description of the such packet generation schemes.

- The first method makes direct use of the available traffic samples. We work with normalized values for the traffic samples, i.e., a sample with value one is inserted when the source is active and a sample with value zero is inserted when the source is passive. This, in deed, is the normalized version of the peak packet generation rate divided by the peak rate. The method suffers from a major drawback, though. Since the samples provided for the network are discrete values equal to either zero or one, the network learning speed is very low. In fact, having a continuously distributed sample spectrum over the interval  $[0, 1]$  leads to have a quite faster learning procedure.
- The second method can be used in case of generating artificial traffic patterns by chaotic maps. The approach simply accomplishes the task of modeling by inserting a level of indirection, i.e., the neural network concentrates on predicting consecutive samples of the chaotic map instead. By modeling the chaotic map, and generating its samples, it would be very easy to generate the same artificial traffic pattern using the same threshold value for as long as the neural network is able to follow the corresponding chaotic map.

This approach is, hence, a combination of the approaches introduced in [1] as neural network modeling of chaotic maps, and [3] as chaotic modeling of bursty traffic.

- The third method provides a sophisticated and an elegant learning approach for well-behaved sources. We define a well-behaved source as a source that does not generate more than a specified number of packets in a time frame, i.e., there is an upper limit on the number of packets generated by the source. The most significant point about this approach is that it uses the real traffic samples where the samples have been arranged to create a continuous range of numbers distributed in  $[0, 1]$  interval. Suppose that the source generates no more than a specified number of traffic packets, say  $P$  in a period of time  $T$ . Then, considering an origin for the time, the cumulative distribution function of the traffic pattern for the period  $T$  is defined as the number of packets generated since the beginning of the time divided by the maximum number of packets, i.e.,  $p/P$ . Obviously, this is a monolithic increasing function starting at zero and ending at one.<sup>1</sup> The samples of this functions can be used to provide the desired sample set. At the end of the period, the desired output is compared with the network output and if

<sup>1</sup>In some cases, the source may generate a number of packets less than the maximum number. That leads to have a monolithic function ending at a value less than one.

the value of error has not entered the acceptable bound the training procedure is repeated.

Relying on the three mentioned training algorithms above, the fixed structure neural network is used to model a number of artificially generated traffic patterns by chaotic maps, namely single and double intermittency maps as discussed previously. The use of artificial traffic patterns provides the possibility of being able to compare the results obtained from all three approaches.

Figures 3, 4, 5, and 6 show the modeling results in case of single and double intermittency maps for initial conditions  $x_0 = 0.1$  and  $x_0 = 0.3$ . In the figures, the horizontal axis displays the discrete-time while the vertical axis displays the normalized version of packet generation at a peak rate. Comparing all three approaches together, it seems that the second approach provides the best results in terms of tracking. Comparing the first and third approaches, it is easy to observe the third approach provides more reliable results as it is able to follow the traffic pattern in a longer period of time. In general and as illustrated in the figures, the familiar ON-OFF learning/prediction pattern is observed. The neural network is able to follow the traffic pattern for the first time after approximately 700'000 and 1'000'000 million iterations in case of single and double intermittency maps respectively and can stay within the acceptable error bound for the next 60 and 40 samples. The network then goes out of sync and needs to be trained again in order to be able to follow the pattern properly.

### *B. Modeling Aggregated Bursty Traffic at the Gateway Level*

The following section illustrates the application of neural networks in the modeling of an aggregated level of bursty traffic. In order to be able to access such an aggregated traffic pattern, a system consisting of 100 individual sources and a gateway as indicated in figure 7 is considered. This is an example in which a number of end nodes send their packets to an intermediate gateway node. The traffic pattern might include a variety of different packets such as telnet, rlogin, ftp, mail, etc. The arrived packets are stored in a relatively large size buffer before getting forwarded to the corresponding destinations. In order to be able to simulate the real network, each individual source is replaced with an artificial traffic generator following an ON-OFF pattern. The ag-

gregated traffic, hence, can be considered to be self-similar [5]. It is important to note that the objective here is merely predicting the traffic pattern arrived at the gateway. Same as before, a fixed structure perceptron neural network is used for the task of modeling. The network consists of an input layer with eight neurons, three hidden layers with twenty neurons in each layer, and an output layer with one neuron. This is the same structure as indicated in figure 2. The inputs of the network are eight consecutive samples of the traffic pattern and the output of the network is the ninth sample that is supposed to be predicted.

The traffic pattern of each source is obtained from the double intermittency map and is distinguished from other sources by assigning a different threshold value to the corresponding map. Figure 8 shows the result of neural network modeling task. Again, the familiar tracking period followed by a divergent behavior is observed. The only difference is that self-similarity increases the speed of convergence at the aggregated level. In case of figure 8, the neural network learning algorithm converges after approximately 280'000 iterations and is able to follow the real pattern for the next 55 arrivals. Knowing that a statistically self-similar traffic pattern exhibits a fractal-like behavior in the sense that aggregating streams of such traffic pattern typically intensify burstiness instead of smoothing it and reminding its serious impacts on design, control, and analysis of high speed packet based networks, the observed result is a very interesting one as it shows self-similarity provides an extra source of information that can be interpreted as some kind of correlation among the generated traffic patterns.

The conclusion is that the simple nonlinear dynamic of neural networks is able to implicitly capture self-similarity and hence neural networks may be viewed as suitable generators of self-similar traffic.

## V. NUMERICAL ISSUES

In the following, we briefly mention some of the practical problems in the implementation of the modeling algorithm. It is important to note that the learning algorithm is time consuming because of the rich dynamic of the traffic pattern that the neural network is trying to learn. In deed, it takes thousands of samples for the network in order to get trained. In addition, there is a specific problem which can be explained by the fractal nature of traffic, i.e., the

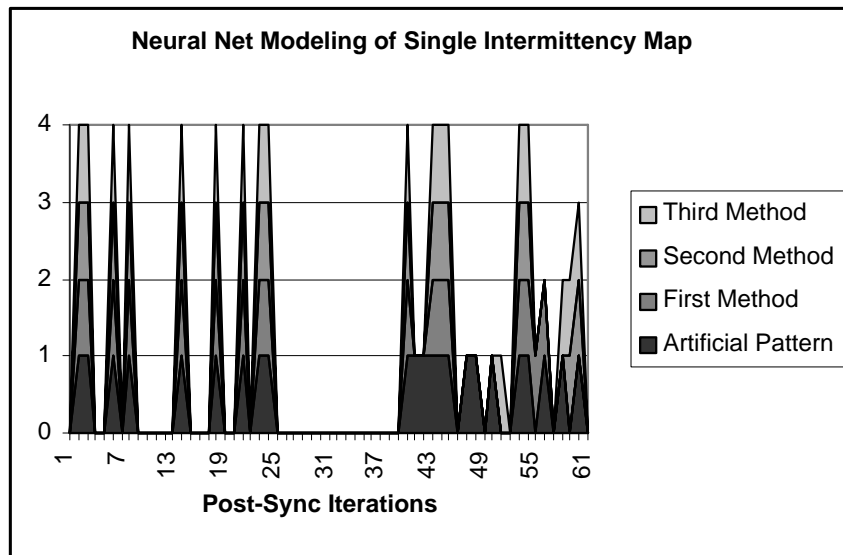


Fig. 3. Results of modeling the ON-OFF traffic pattern generated by single intermittency map for the initial condition  $x_0 = 0.1$ . Results are shown for the artificial traffic pattern, the first modeling method, the second modeling method, and the third modeling method.

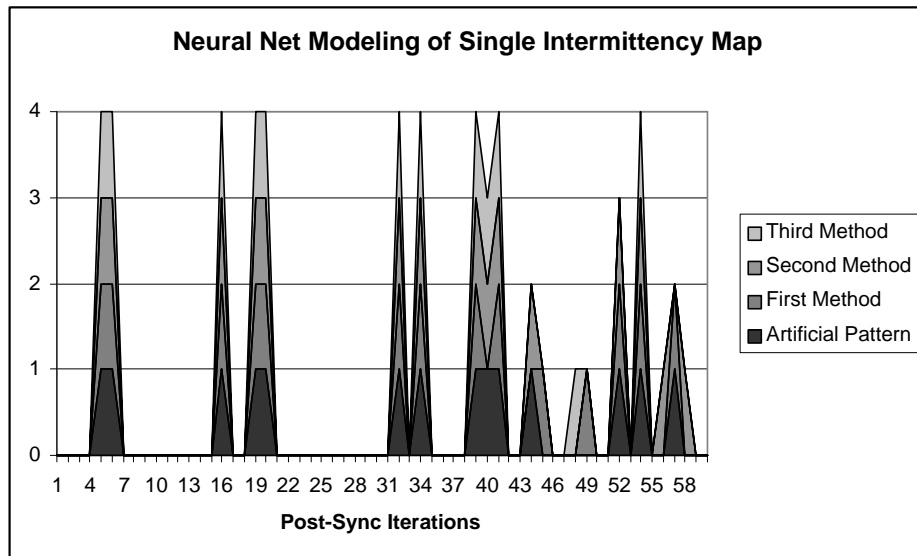


Fig. 4. Results of modeling the ON-OFF traffic pattern generated by single intermittency map for the initial condition  $x_0 = 0.3$ . Results are shown for the artificial traffic pattern, the first modeling method, the second modeling method, and the third modeling method.

traffic patterns with self-similar characteristics have been shown to exhibit chaotic behavior. See [3] for a detailed discussion. To explain the problem, it suffices to mention that although it is possible to reach a very small network error at some step during the learning phase, it is not possible to reach an absolute zero error. It is observed that the small network error begins to grow as time proceeds if it is studied for further samples. The reason relates to the chaotic nature of the system, i.e., since the nonlinear network wants to model a chaotic system, it becomes chaotic itself. In this situation the small error may be consid-

ered as a small difference between two close initial conditions for the desired output and the generated output and as a characteristic of a chaotic system, the difference begins to grow soon. This is nothing more than high sensitivity to the variations of initial conditions in terms of chaos theory. As a matter of fact, observing such a phenomenon can be interpreted as a sign to show that a network that has been trained to model a chaotic map has become chaotic itself. One way of relieving the effect of having an error component which grows with time is to periodically repeat the learning phase between recalling phases. In

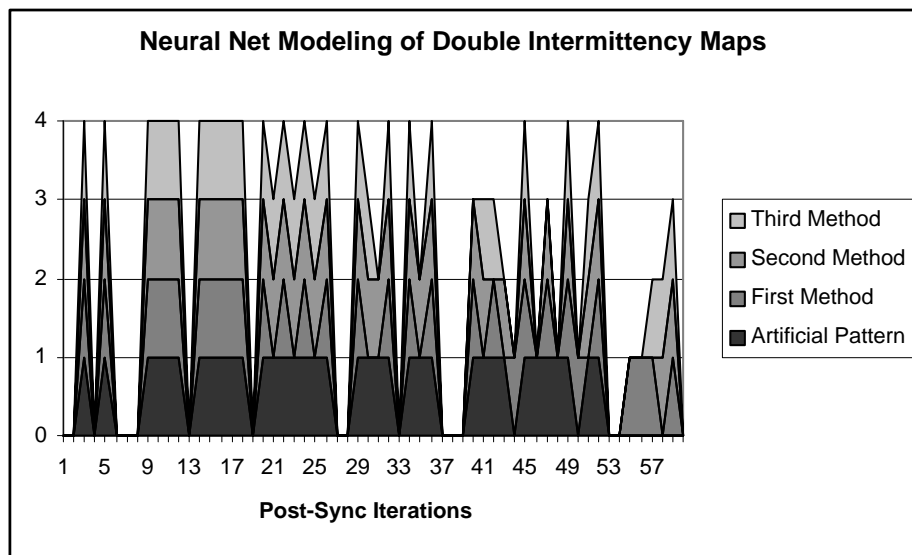


Fig. 5. Results of modeling the ON-OFF traffic pattern generated by double intermittency map for the initial condition  $x_0 = 0.1$ . Results are shown for the artificial traffic pattern, the first modeling method, the second modeling method, and the third modeling method.

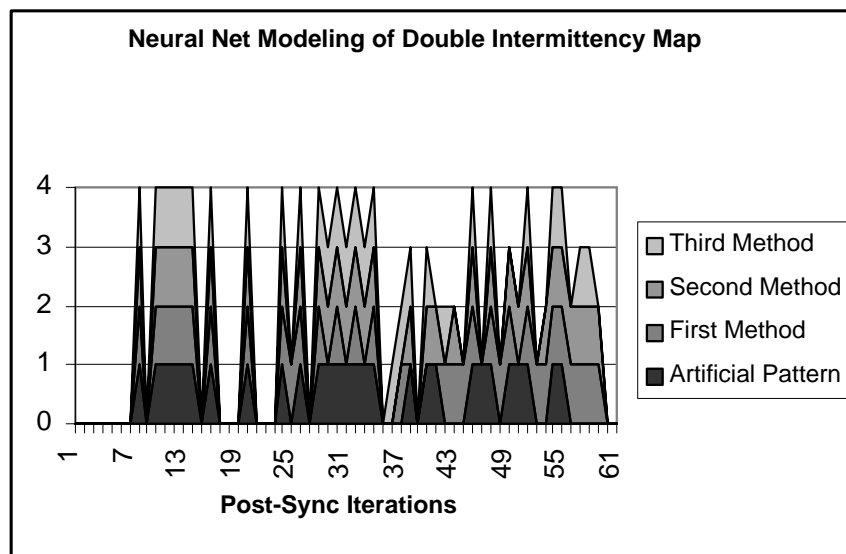


Fig. 6. Results of modeling the ON-OFF traffic pattern generated by double intermittency map for the initial condition  $x_0 = 0.3$ . Results are shown for (a) the artificial traffic pattern, the first modeling method, the second modeling method, and the third modeling method.

practice after the first learning phase with less than one million examples, the network would be able to predict less than one hundred samples and then the learning phase has to be repeated for a comparable number of examples as in the previous learning phase and so on to have reliable results.

Based on the same line of reasoning, all of the convergence results are affected strongly by the choice of initial conditions. It is important to note that the choice of initial values of the parameters play a crucial role in the convergence of the algorithm. Poor choice of initial conditions can and will lead

to observing a divergent behavior. As a practical result and the for the traffic patterns studied, selecting small initial values of the parameters, e.g.,  $w_{ji}(0) = 0.01 \quad \forall i, j$  typically speeds up the learning phase.

## VI. CONCLUSION

In this paper, we studied the modeling of self-similar traffic patterns with a multi-purpose fixed structure system, i.e, a neural network. We investigated the perceptron neural network and back propagation learning algorithm for the task of modeling.

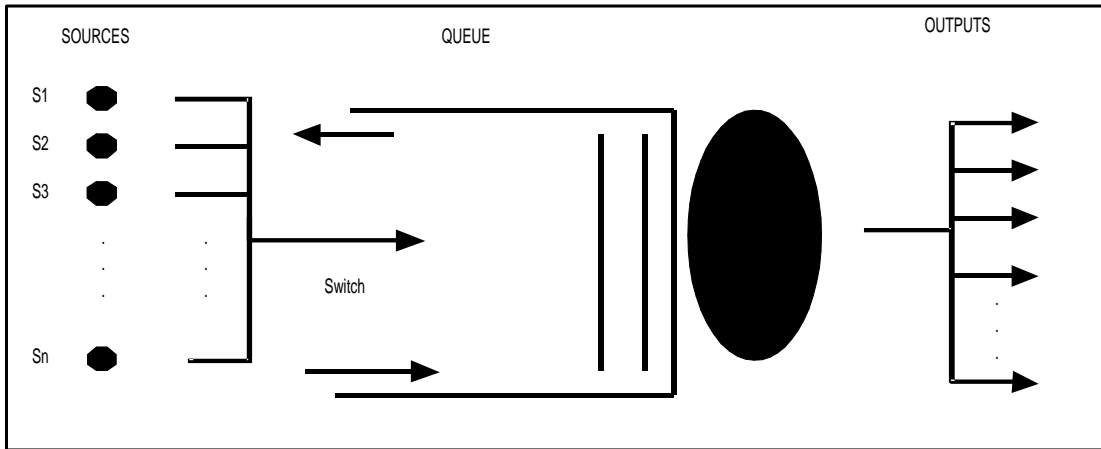


Fig. 7. A sample network used to demonstrate the modeling power of neural networks in modeling aggregated level bursty traffic.

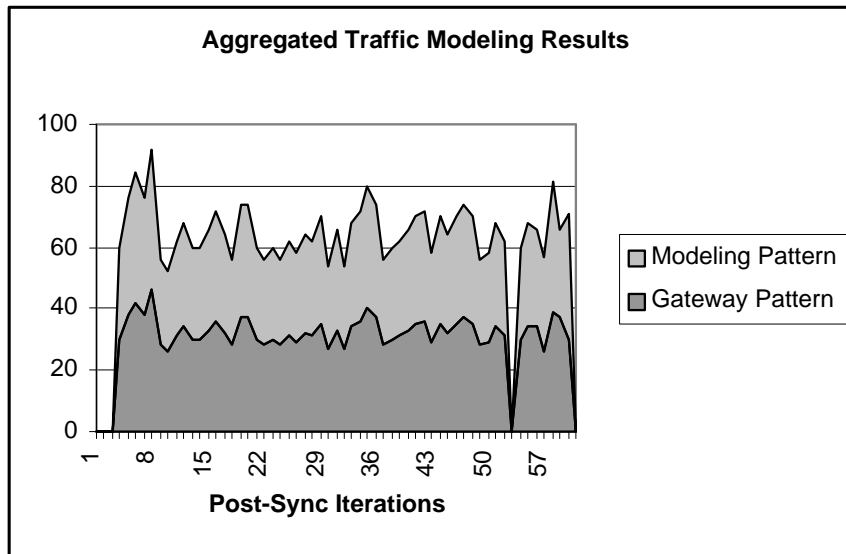


Fig. 8. Neural network modeling results of aggregated level traffic patterns.

We used a fixed structure neural network with three hidden layers in order to model source- and aggregated-level traffic patterns. The neural network had eight neurons in its input layer, and twenty neurons in each of its three hidden layers. The number of neurons in its output layer was one. We applied eight consecutive samples of the pattern as the input of the network and used the ninth sample as the desired output in each iteration of the learning algorithm. The number of samples required for the training of the fixed structure neural network depended on the steady state behavior of the traffic pattern and the network load, i.e., the number of required sam-

ples increased in the case of heavily loaded network which obviously indicated more complicated steady state behavior.

There were several issues in the learning algorithm such as the large number of required training samples and the divergent behavior of the trained neural network. We made use of enhanced back propagation learning algorithm to shorten the learning phase after which the network was able to predict the sequential samples of the corresponding traffic pattern with an error less than 0.01%. We followed each recalling phase with a training phase to be able to cope with the divergent behavior of the trained neural net-

work and provided some intuitive reasoning making use of sensitivity to the variations of initial conditions argument discussed in chaos theory to explain the observed divergent behavior.

## REFERENCES

- [1] H. Yousefi'zadeh, E. A. Jonckheere, "Neural Network Modeling of Discrete Time Chaotic Maps.", Submitted to IEEE Trans. on Neural Networks, Jan. 2002.
- [2] H. Yousefi'zadeh, M. Shafiee, A. Ziloochian, "Chaotic Arrays Modeling with Neural Networks.", Proc. of the Iranian Conference of Electrical Eng., Vol.4, PP. 667- 679, May 1993.
- [3] A. Erramilli, R. P. Singh, P. Pruthi, "Chaotic Maps as Models of Packet Traffic.", ITC Vol. 14, PP. 329-338, 1994.
- [4] W. E. Leland, W. Willinger, M. S. Taqqu, D. V. Willson, "Statistical Analysis and Stochastic Modeling of Self-Similar Datatrafic", ITC Vol. 14, PP. 319-328, 1994.
- [5] W. Willinger, M. Taqqu, "Self-Similarity through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", IEEE/ACM Trans. on Networking, Vol. 5, No. 1, Feb. 1997.
- [6] A. Erramilli, J. Gordon, W. Willinger, "Applications of Fractals in Engineering for Realistic Traffic Processes.", ITC Vol. 14, PP. 35-44, 1994.
- [7] W. E. Leland, W. Willinger, M. S. Taqqu, D. V. Willson, "On the Self-Similar Nature of Ethernet Traffic", IEEE/ACM Trans. on Networking, Vol. 2, NO. 1, PP. 1-15, Feb. 1994.
- [8] V. Ramaswami, "Traffic Performance Modeling for Packet Communication: Whence, Where, and Whither", Proc. of Australian Teletraffic Seminar, Nov. 1988.
- [9] K. M. Hellstern, P. Wirth, "Traffic Models for ISDN Data Users: Office Automation Application", Proc. ITC-13, Denmark, 1991.
- [10] K. Sriram, W. Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", IEEE JSAC, Vol. SAC-4, NO. 6, Sep. 1986.
- [11] H. Heffes, D. M. Lucantoni, "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance.", IEEE JSAC, Vol. 9, NO. 7, Sep. 1991.
- [12] J. Beran, R. Sherman, M. S. Taqqu, W. Willinger, "Variable Bit Rate Video Traffic and Long Range Dependence", IEEE/ACM Trans. on Networking, Vol. 2, NO. 3, Apr. 1994.
- [13] J. M. Pitts, L. G. Cuthbert, M. Bocci, E. M. Scharf, "An Accelerated Simulation Technique for Modeling Burst Scale Queuing Behavior in ATM", ITC Vol. 14, PP. 777-786, 1994.
- [14] D. E. Duffy, W. Willinger, "Statistical Analysis of CCSN/SS7 Traffic Data from Working CCS Subnetworks", IEEE JSAC, 1994.
- [15] A. Giovanardi, R. Rovatti, G. Mazzini, "Queue System Analytical Study with Self-Similar Chaos-Based Traffic", Electronics Letters, Volume: 37 Issue: 3, PP 169-170, Feb. 2001.
- [16] A. Alkhatib, M. Krunz, "Application of Chaos Theory to the Modeling of Compressed Video", Proceedings of the IEEE ICC 2000 Conference, Vol. 2, New Orleans, June 2000.
- [17] A. Adas, "Traffic Models in Broadband Networks", IEEE Communications Magazine, PP 82-89, July 1997.
- [18] M. Minsky, S. A. Papert, "Perceptrons: An Introduction to Computational Geometry.", MIT Press, Cambridge, MA, expanded edition, 1988/1969.
- [19] S. E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks", Technical Report CMU-CS-88-162, Carnegie Mellon University, June 1988
- [20] A. Van Ooyen, B. Neihuis, "Improving the Convergence of Back Propagation Algorithm", Neural Networks, Vol.5, No.3, 1992
- [21] H. Guo, S. Gelfland, "Analysis of Gradient Descent Learning Algorithms for Multilayer Feed Forward Neural Networks", IEEE Trans. on Circuit & Syst., Vol. 38, No.8, Aug. 1991

- [22] H. Yang, T. Dillon, "Convergence of Self-Organizing Neural Algorithms", Neural Networks, Vol.5, No.3, 1992
- [23] G. Mirchandani, W. Cao, "On Hidden Nodes for Neural Nets", IEEE Trans. on Circuit & Syst., Vol.36, No.5, 1989
- [24] N. J. Dimopoulos, "A Study of Asymptotic Behavior of Neural Networks", IEEE Trans. on Circuit & Syst., Vol. 36, No.5, 1989
- [25] V. Kurkova, "Kolmogrov's Theorem and Multilayer Neural Nets", Neural Networks, Vol.5, No.3, 1992

PLACE  
PHOTO  
HERE

**Homayoun Yousefi'zadeh** was born in Tehran, Iran. He received B.S., M.S., and Ph.D. degrees all in Electrical Engineering from Sharif University of Technology, Tehran Polytechnic Institute, and University of Southern California in 1989, 1993, and 1997 respectively. Dr. Homayoun Yousefi'zadeh research and industry experience spans over teletraffic modeling and analysis, network traffic control, real-time media systems, distributed database systems, storage networking, and enterprise client-server applications. He is currently with the Center for Pervasive Communications in Electrical and Computer Engineering Department of University of California, Irvine. He has been involved with a number of academic and industry initiatives in different capacities including chairperson of systems' management workgroup of Storage Networking Industry Association (SNIA), member of Scientific Advisory Board (SAB) of Integrated Media Services Center (IMSC) at the University of Southern of California, referee for IEEE Communication Letters, main panelist in domestic and international storage networking events such as IDC's European Storage Symposium and Search Storage Online, member of American Management Association (AMA), and member of American Society for Engineering Education (ASEE). His research interests include intelligent modeling of nonlinear dynamics, teletraffic analysis and control, heterogeneous real-time media systems, high-speed broadband networks, and distributed algorithms.