# Computer Technology and Computer Programming

## New Research and Strategies

James L. Antonakos
Editor

# COMPUTER TECHNOLOGY AND COMPUTER PROGRAMMING

New Research and Strategies

# COMPUTER TECHNOLOGY AND COMPUTER PROGRAMMING

## New Research and Strategies

James L. Antonakos

*Distinguished Professor of Computer Science,*
*Broome Community College, State University of New York,*
*Binghamton; Online Instructor and Faculty Advisor, Excelsior College,*
*Albany, New York and Sullivan University, Kentucky, U.S.A.*

Apple Academic Press

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Apple Academic Press, Inc
3333 Mistwell Crescent
Oakville, ON L6L 0A2
Canada


Version Date: 20140602

International Standard Book Number-13: 978-1-4665-6259-2 (eBook - PDF)



**Visit the Taylor & Francis Web site at**
**http://www.taylorandfrancis.com**

**and the CRC Press Web site at**
**http://www.crcpress.com**

**For information about Apple Academic Press product**
**http://www.appleacademicpress.com**

# CONTENTS


*Introduction* 7

1. Novel FTLRNN with Gamma Memory for Short-Term and Long-Term Predictions of Chaotic Time Series 9
   *Sanjay L. Badjate and Sanjay V. Dudu*
2. Flexible Interconnection Network for Dynamically and Partially Reconfigurable Architectures 50
   *Ludovic Devaux, Sana Ben Sassi, Sebastien Pillement, Daniel Chillet and Didier Demigny*
3. A Hardware Solution for an "On the Fly" Encryption 82
   *Daniel Filipas*
4. SQL Generation for Natural Language Interface 90
   *László Kovács*
5. Web 2.0 Technologies with jQuery and Ajax 99
   *Cornelia Györödi, Robert Györödi, George Pecherle, Tamas Lorand and Rosu Alin*
6. Reconfigurable Computing—A New Paradigm 111
   *Erica Mang, Ioan Mang and Popescu-Rotoiu Constantin*

7. In-Network Adaptation of Video Streams Using Network Processors 125
*Mohammad Shorfuzzaman, Rasit Eskicioglu and Peter Graham*
8. A Survey of Visual Sensor Networks 168
*Stanislava Soro and Wendi Heinzelman*
9. A Family of Tools for Supporting the Learning of Programming 213
*Guido Rößling*
10. InfoVis Interaction Techniques in Animation of Recursive Programs 231
*J. Ángel Velázquez-Iturbide and Antonio Pérez-Carrasco*
11. Towards a Serious Game to Help Students Learn Computer Programming 250
*Mathieu Muratet, Patrice Torguet, Jean-Pierre Jesse and Fabienne Viallet*
12. Distributed Network, Wireless and Cloud Computing Enabled 3-D Ultrasound; a New Medical Technology Paradigm 276
*Arie Meir and Boris Rubinsky*
13. miRMaid: A Unified Programming Interface for Microrna Data Resources 293
*Anders Jacobsen, Anders Krogh, Sakari Kauppinen and Morten Lindow*
14. Some Attributes of a Language for Property-Based Testing 306
*Matt Bishop and Vicentiu Neagoe*
*Index* 327

# INTRODUCTION

What I find most remarkable about the field of computer science is its vast scope. Practically any topic you might imagine falls in some way under the umbrella of computer science. Many topics may seem to naturally belong there, such as research into advanced computer architectures, distributed and cloud computing (and their associated high-speed networking components), computer forensics, operating systems, and the details of many different programming languages. But these areas are just a few of a wider array of topics and activities found in computer science. Computer scientists spend a great deal of time and energy studying compression algorithms for images, video, and data; encryption techniques; efficient hardware computation pipelines; computer gaming and its associated artificial intelligence; networking protocols that enable secure and reliable transmission of information; image processing; database technologies; and new ways of sharing information over the Internet.

Of course, many areas of computer science require a good foundation in mathematics. Here it is remarkable to know that we use mathematics to prove that some things are possible and that other things are impossible. Some concepts or problems have not yet been proved either way, even with a great many researchers looking into them. When and if these open problems are eventually solved, the solutions will usher in a new age in computer science and also new challenges. For example, if we gain a deeper insight and understanding of random numbers, what will be the effect on the security algorithms we use every day to encrypt our private communication over the Internet?

Let us also take note of the astounding visual reality now available, of computer graphics algorithms so complex they render stunning visual effects in video games in real time and produce "Is it real or computer generated?" effects in motion pictures. Again, here the computer scientist must have programming skills, knowledge of mathematics, physics, optics, and the hardware details of the processor or processors rendering the image. We can see for ourselves the fruits of many researchers' labors over the years.

The time spent by computer scientists examining arcane topics that appear to have little practical application is very misleading. Advances in medical imaging, understanding biological processes, recognizing human speech, mining data and distinguishing patterns, and exploring the nature of memory via neural networks have all been made possible by computer scientists toiling away in their labs.

Today the line between software and hardware is becoming blurred. A computer scientist crafting a new optimizing compiler must have detailed knowledge of the internal hardware workings of a processor in order to efficiently schedule instructions and generate code that utilizes the processor pipeline, registers, and cache memory to provide maximum performance. Even something as simple as extending the life of the battery in a laptop computer is a combined effort between the hardware designers and the software writers.

Perhaps the most important quality a computer scientist can possess is curiosity, a constant desire to understand how things work. When this curiosity is coupled with determination, the end result is often useful in ways that were not originally intended. Keep this curiosity in mind as you read the papers on *Computer Technology and Computer Programming* contained within this book.

**— James L. Antonakos**

# Novel FTLRNN with Gamma Memory for Short-Term and Long-Term Predictions of Chaotic Time Series

**Sanjay L. Badjate and Sanjay V. Dudu**

## ABSTRACT

*Multistep ahead prediction of a chaotic time series is a difficult task that has attracted increasing interest in the recent years. The interest in this work is the development of nonlinear neural network models for the purpose of building multistep chaotic time series prediction. In the literature there is a wide range of different approaches but their success depends on the predicting performance of the individual methods. Also the most popular neural models are based on the statistical and traditional feed forward neural networks. But it is seen that this kind of neural model may present some disadvantages when long-term prediction is required. In this paper focused time-lagged recurrent neural network (FTLRNN) model with gamma memory is developed for*

*different prediction horizons. It is observed that this predictor performs remarkably well for short-term predictions as well as medium-term predictions. For coupled partial differential equations generated chaotic time series such as Mackey Glass and Duffing, FTLRNN-based predictor performs consistently well for different depths of predictions ranging from short term to long term, with only slight deterioration after k is increased beyond 50. For real-world highly complex and nonstationary time series like Sunspots and Laser, though the proposed predictor does perform reasonably for short term and medium-term predictions, its prediction ability drops for long term ahead prediction. However, still this is the best possible prediction results considering the facts that these are nonstationary time series. As a matter of fact, no other NN configuration can match the performance of FTLRNN model. The authors experimented the performance of this FTLRNN model on predicting the dynamic behavior of typical Chaotic Mackey-Glass time series, Duffing time series, and two real-time chaotic time series such as monthly sunspots and laser. Static multi layer perceptron (MLP) model is also attempted and compared against the proposed model on the performance measures like mean squared error (MSE), Normalized mean squared error (NMSE), and Correlation Coefficient (r). The standard back-propagation algorithm with momentum term has been used for both the models.*

## Introduction

Predicting the future which has been the goal of many research activities in the last century is an important problem for human, arising from the fear of unknown phenomenon and calamities all around the infinitely large world with its many variables showing highly nonlinear and chaotic behavior. Chaotic time series have many applications in various fields of science, for example, astrophysics, fluid mechanics, medicine, stock market, weather, and are also useful in engineering such as speech coding [1], radar modeling of electromagnetic wave propagation and scattering [2]. The chaotic interconnected complex dynamical systems in nature are characterized by high sensitivity to initial conditions which results in long-term unpredictability. The dynamical reconstruction seems to be extremely difficult, even in developing era of super computers, not because of computational complexity, but due to inaccessibility of perfect inputs and state variables. Many different methods have been developed to deal with chaotic time series prediction. Among them neural networks occupy an important place being adequate model of the nonlinearity and nonstationarity.

Inspired from the structure of the human brain and the way it is supposed to operate, neural networks are parallel computational systems capable of solving

number of complex problems in such a diverse areas as pattern recognition, computer vision, robotics, control and medical diagnosis, to name just few [3]. Neural networks are an effective tool to perform any nonlinear input output mappings and prediction problem [4]. Predicting a chaotic time series using a neural network is of particular interest [5]. Not only it is an efficient method to reconstruct a dynamical system from an observed time series, but it also has many applications in engineering problems like radar noise cancellation [6], radar [7] demodulation of chaotic secure communication systems [8], and spread spectrum/code division multiple access (CDMA) systems [9, 10]. It is already established that, under appropriate conditions, they are able to uniformly approximate any complex continuous function to any desired degree of accuracy [11]. Later, similar results were published independently in [12]. It is these fundamental results that allow us to employ neural network in time series prediction. Since neural networks' models do not need any a priori assumption about the underlying statistical distribution of the series to be predicted, they are commonly classified as "data-driven" approach, to contrast them with the "model-driven" statistical methods. Neural networks that are the instruments in broad sense can learn the complex nonlinear mappings from the set of observations [13]. The static MLP network has gained an immense popularity from numerous practical application published over the past decade, there seems to be substantial evidence that multilayer perceptron indeed possesses an impressive ability [14]. There have been some theoretical results that try to explain the reasons for the success in [15, 16]. Most applications are based on feed-forward neural networks, such as the back-propagation (BP) network [17] and Radial basis function (RBF) network [18, 19]. It has also been shown that modeling capacity of feed-forward neural networks can be improved if the iteration of the network is incorporated into the learning process [20].

Several methods with different performance measures have been attempted in the literature to predict the chaotic time series. It is has been predicted for Mackey-Glass chaotic time series for short-term ahead prediction with a percentage error of 20% [21]. A new class of wavelet network was developed with a standard deviation of 0.0029 for short-term ahead prediction of Mackey-Glass chaotic time series and annual sunspots for 1 step ahead prediction [22]. By using recurrent predictor neural network for monthly sunspots chaotic time series for 6 months ahead prediction with $E_{PA}$ (Prediction accuracy) equals 0.992 and $E_{RMSE}$ (Root mean squared error) equals 4.419, for 10 months ahead prediction with $E_{PA}$ of 0.980 and $E_{RMSE}$ of 7.050, for 15 months ahead prediction of $E_{PA}$ equals 0.9222 and $E_{RMSE}$ equals 13.658, and 20 months ahead prediction with $E_{PA}$ of 0.866 and $E_{RMSE}$ of 16.79323 [23]. Also by using radial basis function with orthogonal least square Fuzzy model for monthly sunspots with prediction error +6 to -4 and for Mackey-Glass chaotic time series with $E_{RMSE}$ of 0.0015 [24]. It is also attempted with Hybrid network for Mackey-Glass time series with iterative prediction and

Normalized Mean Square Error NMSE of 0.053 [25]. By using Elman neural network for yearly sunspots for 1 year ahead prediction with $E_{RMSE}$ equals 30.2931 and prediction accuracy of 0.9732 [26].

From the scrupulous review of the related research work, it is noticed that no simple model is available for long-term prediction of chaotic time series so far. It is necessary to develop a simple model that is able to perform short-, medium- and long-term predictions of chaotic time series with reasonable accuracy. In view of the remarkable ability of neural network in learning from the instances, it can prove as a potential candidate with a view to design a versatile predictor (forecaster) for the chaotic time series. Hence in this paper a novel focused time-lagged recurrent neural network model with gamma memory filter is proposed as an intelligent tool for predicting the two non linear differential equation Mackey-Glass and Duffing time series and two real-time monthly sunspots and Laser chaotic time series not only for short-term but long-term prediction also because they acquire temporal processing ability through the realization of short-term memory and information about the preceding units, which is important when the long-term prediction is required. The Mackey-Glass chaotic time series was first proposed as a model for white blood cell production, the Duffing chaotic time series describes a specific nonlinear circuit or the hardening spring effect observed in many mechanical problems, monthly sunspots number is a good measure of solar activity which has a period of 11 years, so-called solar cycle. The solar activity has a measure effect on earth, climate, space weather, satellites, and space missions, and a highly nonlinear laser time series. These chaotic time series are the good benchmark for the proposed model. The various parameters like number of hidden layers, number of processing elements in the hidden layer, step size, the different learning rules, the various transfer functions like tanh, sigmoid, linear-tan-h, and linear sigmoid, different error norms $L_1$,$L_2$,$L_3$,$L_4$,$L_5$, and $L_\infty$, the different memories TDNN, Laguarre and gamma filter, and different combination of training and testing samples are exhaustively varied and experimented for obtaining the optimal values of performance measures as mentioned in the flow chart. The obtained results indicate the superior performance of estimated dynamic FTLRNN-based model with gamma memory over the MLPNN in various performance measures such as Mean Square Error (MSE), Normalized Mean Square Error (NMSE), and correlation coefficient (r) on testing as well as training data set. The proposed network is attempted for training up to 20 000 numbers of epochs for obtaining the improved values of performance measures. The experimentation process is demonstrated in flow chart of Figure 1. This paper is organized as follows in Section 2 the static MLP model is presented, and the learning procedure is explained. In Section 3 the proposed FTLRNN model is explained. In Section 4 the performance measures and their importance are discussed. Section 5 explains about the significance of the benchmark chaotic

time series. Section 6 explains the experimental procedure and analysis. Section 7 summarizes the evaluation results and analyses for the proposed model. Finally concluding remarks on empirical findings are provided in Section 8.

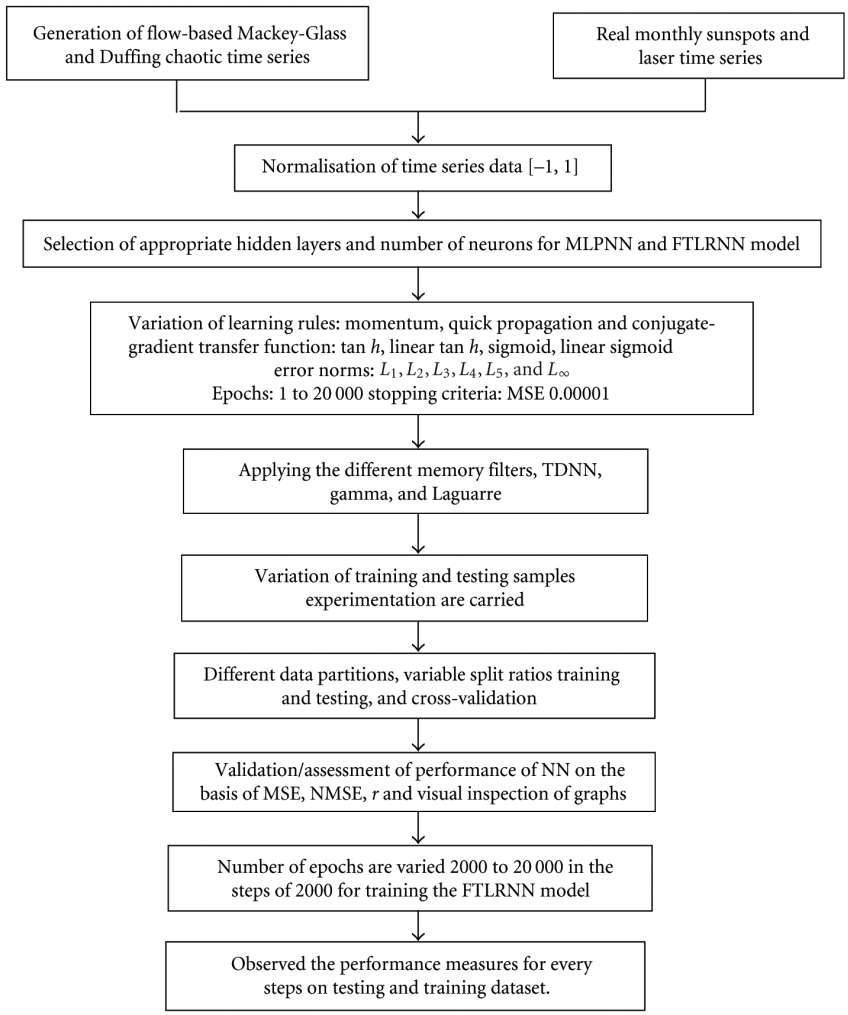


**Figure 1.** Flow Chart.

## Static NN-Based Model

Static Neural networks typically use Multilayer perceptron MLP as a backbone. They are layered feed-forward networks typically trained with static back propagation.

MLP solid-based model has a solid foundation [27, 28]. The main reason for this is its ability to model simple as well as complex functional relationships. This has been proven through a number of practical applications [29]. In [11] it is shown that all continuous functions can be approximated to any desired accuracy, in terms of the uniform norm, with a network of one hidden layer of sigmoid or (hyperbolic tangent) hidden units and a layer of linear or tanh output unit to include in the hidden layer. The paper does not explain how many units to include in the hidden layer. This is discussed in [30], and a significant result is derived approximation capabilities of two layer perception networks when the function to be approximated shows certain smoothness. The biggest advantage of using MLP NN for approximation of mapping from input to the output of the system resides in its simplicity and the fact that it is well suited for online implementation. The objective of training is then to determine a mapping from a set of training data to the set of possible weights so that the network will produce predictions y(t), which in some sense are close to the true outputs y(t). The prediction error approach is based on the introduction of measure of closeness in terms of mean square error (MSE) criteria:

$$\begin{aligned} V_N(\theta, Z^N) &= \frac{1}{2N}\sum_{t=1}^{N}\left[y(t) - y(t^{\wedge}\big|\right]^T \left[y(t) - y(t^{\wedge}\big|\theta)\right] \\ &= \frac{1}{2N}\sum_{t=1}^{N}\varepsilon^2(t,\theta). \end{aligned} \tag{1}$$

The weights are then found as

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta, Z^N), \tag{2}$$

by some kind of iterative minimization scheme:

$$\theta^{(i+1)} = \theta^{(i)} + \mu^{(i)} f^{(i)}, \tag{3}$$

where $\theta^{(i)}$ specifies the current iterate (number "i"), $f^{(i)}$ is the search direction, and $\mu^{(i)}$ is the step size.

When NN has been trained, the next step is to evaluate it. This is done by standard method in statistics called independent validation [31]. It is never a good idea to assess the generalization properties of an NN-based on training data alone. This method divides the available data sets into two sets, namely, training data set and testing data set. The training data set are next divided into two partitions: the first partition is used to update the weights in the network and the second partition is used to assess (or cross-validate) the training performance. The testing data set are then used to assess how the network has generalized. The learning and

generalization ability of the estimated NN-based model is assessed on the basis of certain performance measures such as MSE, NMSE, and the regression ability of the NN by visual inspection of the correlation coefficient characteristics for different outputs of system under study.

## FTLRNN Model

Time-lagged recurrent networks (TLRNs) are MLPs extended with short-term memory structures. Here, a "static" NN (e.g., MLP) is augmented with dynamic properties [14]. This, in turn, makes the network reactive to the temporal structure of information bearing signals. For an NN to be dynamic, it must be given memory. This memories may be classified into "short-term" and "long-term" memories. Long-term memory is built into an NN through supervised learning, whereby the information content of the training data set is stored (in part or in full) in the synaptic weights of the network [32]. However, if the task at hand has a temporal dimension, some form of "short-term" memory is needed to make the network dynamic. One simple way of building short-term memory into the structure of an NN is through the use of time delays, which can be applied at the input layer of the network (focused). A short-term memory structure transforms a sequence of samples into a point in the reconstruction space [33]. This memory structure is incorporated inside the learning machine. This means that instead of using a window over the input data, processing elements (PEs) created are dedicated to store either the history of the input signal or the PE activations.

The input PEs of an MLP are replaced with a tap delay line, which is followed by the MLPNN. This topology is called the focused time-delay NN (TDNN). The focused topology only includes the memory Kernels connected to the input layer. This way, only the past of the input is remembered. The delay line of the focused TDNN stores the past samples of the input. The combination of the tap delay line and the weights that connect the taps to the PEs of the first hidden layer is simply linear combiners followed by a static nonlinearity. Typically, a gamma short-term memory mechanism is combined with nonlinear PEs in restricted topologies called focused. Basically, the first layer of the focused TDNN is a filtering layer, with as many adaptive filters as PEs in the first hidden layer. The outputs of the linear combiners are passed through a nonlinearity (of the hidden-layer PE) and are then further processed by the subsequent layers of the MLP for system identification, where the goal is to find the weights that produce a network output that best matches the present output of the system by combining the information of the present and a predefined number of past samples (given by the size of the tap delay line) [32]. Size of the memory layer depends on the number of past samples that are needed to describe the input characteristics in time. This

number depends on the characteristics of the input and the task. This focused TDNN can still be trained with static back propagation, provided that a desired signal is available at each time step. This is because the tap delay line at the input layer does not have any free parameters. So the only adaptive parameters are in the static feed-forward path.

The memory PE receives in general many inputs $x_i(n)$ and produces multiple outputs $y = [y_0(n),\ldots, y_D(n)]^T$, which are delayed versions of $y_0(n)$ the combined input,

$$y_k(n) = g(y_{k-1}(n)), \qquad y_0(n) = \sum_{j=1}^{P} x_j(n), \tag{4}$$

where g(·) is a delay function.

These short-term memory structures can be studied by linear adaptive filter theory if g(·) is a linear operator. It is important to emphasize that the memory PE is a short-term memory mechanism, to make clear the distinction from the network weights, which represent the long-term memory of the network.

There are basically two types of memory mechanisms: memory by delay and memory by feedback. We seek to find the most general linear delay operator (special case of the Auto Regressive Moving Average model), where the memory traces yK(n) would be recursively computed from the previous memory trace yK-1(n). This memory PE is the generalized feed-forward memory PE. It can be shown that the defining relationship for the generalized feed-forward memory PE is mentioned

$$g_k(n) = g(n)*g_{k-1}(n), \qquad k \geq 1, \tag{5}$$

where, * is the convolution operation, g(n) is a causal time function, and K is the tap index. Since this is a recursive equation, $g_0(n)$ should be assigned a value independently. This relationship means that the next memory trace is constructed from the previous memory trace by convolution with the same function g(n), the memory Kernel yet unspecified. Different choices of g(n) will provide different choices for the projection space axes. When we apply the input x(n) to the generalized feed-forward memory PE, the tap signals $y_K(n)$ become

$$y_K(n) = g(n)* y_{K-1}(n), \tag{6}$$

the convolution of y_{k-1}(n) with the memory Kernel. For k=0, we have

$$y_0(n) = g_0(n)* x(n), \tag{7}$$

where g0(n) may be specified separately. The projection x(n) of the input signal is obtained by linearly weighting the tap signals according to

$$x(n) = \sum_{k=0}^{D} w_k y_k(n). \tag{8}$$

The most obvious choice for the basis is to use the past samples of the input signal x(n) directly, that is, the Kth tap signal becomes yK(n)=x(n-K). This choice corresponds to

$$g(n) = \delta(n-1). \tag{9}$$

In this case $g_0(n)$ is also a delta function δ(n) (delta function operator used in the tap delay line). The memory depth is strictly controlled by D, that is, the memory traces store the past D samples of the input. The time delay NN uses exactly this choice of basis.

The gamma memory PE attenuates the signals at each tap because it is a cascade of leaky integrators with the same time constant gamma model. The gamma memory PE is a special case of the generalized feed-forward memory PE, where

$$g(n) = \mu(1-\mu)^n, \qquad n \geq 1, \tag{10}$$

and g0(n)=δ(n). The gamma memory is basically a cascade of low-pass filters with the same time constant 1-μ. The over all impulse response of the gamma memory is

$$g_p(n) = \binom{n-1}{p-1} \mu^p (1-\mu)^{n-p}, \qquad n \geq p, \tag{11}$$

where (:) is a binomial coefficient defined by

$$\binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{p!}. \tag{12}$$

For integer values of n and p, the overall impulse response $g_p(n)$ for varying p represents a discrete version of the integrand of the gamma function, hence the name of the memory.

The gamma memory PE has a multiple pole that can be adaptively moved along the real Z-domain axis, that is, the gamma memory can implement only low-pass ($0 < \mu < 1$) or high-pass ($1 < \mu < 2$) transfer functions. The high-pass transfer function creates an extra ability to model fast-moving signals by alternating the signs of the samples in the gamma PE (the impulse response for $1 < \mu < 2$ has alternating signs). The depth in samples parameters (D) is used to compute the number of taps (T) contained within the memory structure (s) of the network.

## Performance Measures

Three different types of statistical performance evaluation criteria were employed to evaluate the performance of these models developed in this paper. These are as follows.

MSE: the mean square error is given by:

$$MSE = \frac{\sum_{j=0}^{P} \sum_{i=0}^{N} (dij - yij)^2}{N \times P}, \tag{13}$$

where P = number of output PEs, N = number of exemplars in the data set, $y_{ij}$ = network output for exemplar i at PEj, and $d_{ij}$ = desired output for exemplar i at PEj.

NMSE (normalized mean square error). The normalized mean square error is defined by the following formula, where P = Number of output PEs, N = Number of exemplars in data set,

$$NMSE = \frac{P \times N \times MSE}{\sum_{J=0}^{P} \left( \left( N \sum_{i=0}^{N} d_{ij}^2 - \left( \sum_{i=0}^{N} d_{ij} \right)^2 \right) / N \right)}. \tag{14}$$

MSE=Mean square error, $d_{ij}$ = desired output for exemplar i at PEj ( jth element of PEs) Correlation Coefficient (r). The mean square error (MSE) can be used to determine how well the network output fits the desired output but it does not necessarily reflect whether the two sets of data move in the same direction. For instance by simply scaling the network output, we can change the MSE without changing the directionality of the data. The correlation coefficient solves this problem. By definition, the correlation coefficient between a network output x and a desired output d is

$$r = \frac{\sum_{i} \left( (xi - \overline{x}) \left( di - \overline{d} \right) / N \right)}{\sqrt{\sum_{i} \left( \left( di - \overline{d} \right) / N \right)} \times \sqrt{\sum_{i} \left( \left( di - \overline{d} \right) / N \right)}}, \tag{15}$$

The correlation coefficient is confined to the range [-1,1].

## Benchmark Chaotic Time Series

In science, chaos is used as a synonym for irregular behavior, whose long-term prediction is essentially unpredictable. Chaotic differential equations exhibit not

only irregular behavior but they are also unstable with respect to small perturbations of their initial condition. Consequently it is difficult to forecast the future of time series based on chaotic differential equations; they are the good benchmark for a neural network design algorithm.

## Mackey-Glass Time Series

The Mackey-Glass equation is time delay differential equation, which was first proposed as model of white blood cells production [34]. It is often used in practice as a benchmark set because of its nonlinear chaotic characteristics. Chaotic time series do not converge or diverge in time, and their trajectories are highly sensitive to initial conditions. Data are generated by using fourth order Runge-Kutta method.The equation is given by:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^c} - bx(t), \tag{16}$$

where a, b, c are constant coefficients and t is time delay, the coefficient we are selected: a=0.2, b=0.1, c=10, τ=17. The Mackey-Glass time series is shown in Figure 2.

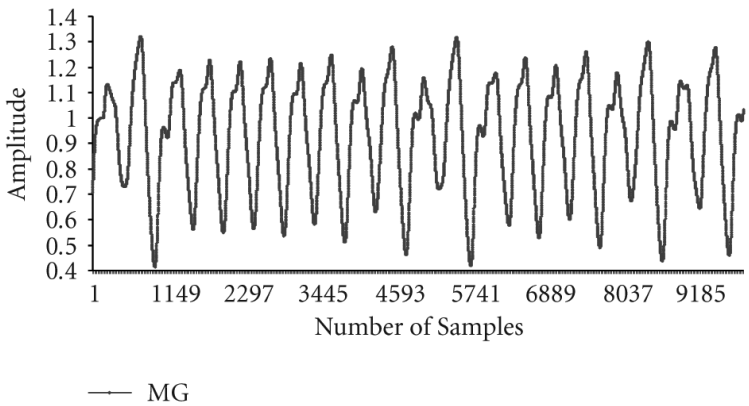


**Figure 2.** Mackey-Glass time series.

## Duffing Time Series

Duffing time series describes a specific nonlinear circuit or the hardening spring effect observed in many mechanical problems [35]. The Duffing equation is time delay differential equation which is given as

$$\begin{aligned}\frac{dy}{dt} &= y,\\ \frac{dy}{dt} &= \left\{F^x Cos(\tau) - x^3 - b^x y\right\},\\ \frac{d\tau}{dt} &= w,\end{aligned} \tag{17}$$

where driving force F = 7.5, $X_0$=initial position 1.0, damping constant (b = 0.05), frequency w = 1.0, Delay time = 0.001. The chaotic time series is as shown in Figure 3.

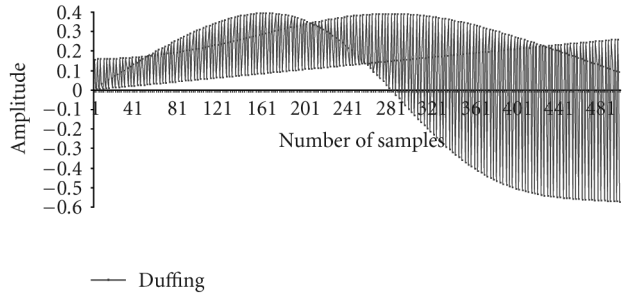


**Figure 3.** Duffing time series (first 500 samples).

## Sunspot Time Series

A sunspot number is a good measure of solar activity which has a period of 11 years, so-called solar cycle. The solar activity has a measure effect on earth, climate, space weather, satellites, and space missions, thus is an important value to be predicted. But due to intrinsic complexity of time behavior and the lack of a quantitative theoretical model, the prediction of solar cycle is very difficult. Many prediction techniques have been examined on the yearly sunspots number time series as an indicator of solar activity. However, in more recent studies the international monthly sunspot time series, which has a better time resolution and accuracy, has been used. In particular, a nonlinear dynamics approach has been developed in [36], and prediction results are compared between several prediction techniques from both statistical and physical classes. There has been a lot of work on controversial issue of nonlinear characteristics of the solar activity [36–39]; several recent analyses have provided evidence for low-dimensional deterministic nonlinear chaotic behavior of the monthly smoothed sunspot time series [36–38] which has intense. The data considered the monthly variations from January 1749

to December 2006. The total samples are 3096 considered and demonstrated in Figure 4. The series is normalized in the range of -1 to +1. The monthly smoothed sunspot number time series is downloaded from the SIDC (World Data Center for the Sunspot Index) [40].

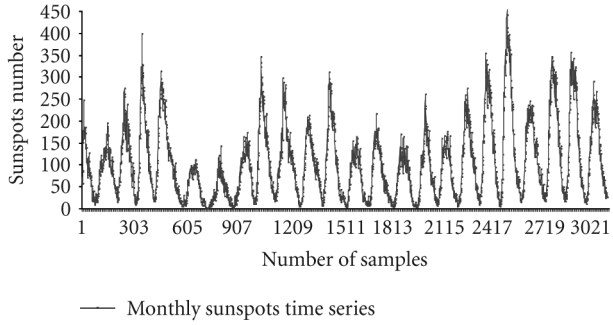


**Figure 4.** Monthly Sunspot time series.

## Laser Time Series

The laser data were recorded from a Far Infrared-laser in a chaotic state. The measurements were made on an 81.5-micron 14 $NH_3$ cw (FIR) laser, pumped optically by the P (16) line of an $N_2O$ laser, via the vibrational aQ(8, 7) $NH_3$ transition. The basic laser setup can be found in [41]. The intensity data was recorded by an Le Croy oscilloscope. It was made available worldwide during a time series prediction competition organized by Santa Fe Institute and a highly nonlinear data set, since then, it has been used in benchmark studies. The time series has 1000 samples points which has been rescaled to the range of [-1,1]. The time series is shown in Figure 5.

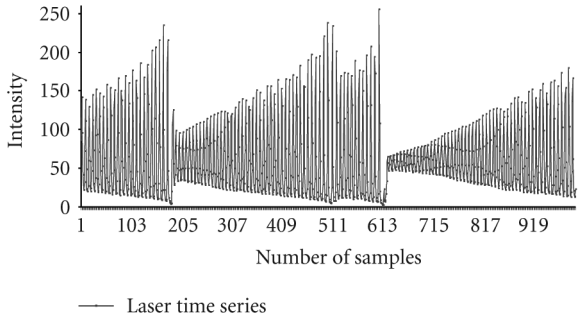


**Figure 5.** Laser time series.

## Experimental Results

The choice of the number of hidden layers and the number of hidden units in each hidden layers is critical [42]. It has been established that an MLPNN that has only one hidden layer, with sufficient number of neurons, acts as a universal approximators of nonlinear mappings [43]. The tradeoff between accuracy and complexity of the model should be resolved accurately [44]. In practice, it is very difficult to determine a sufficient number of neurons necessary to achieve the desired degree of approximation accuracy. Frequently the number of units in the hidden layer is determined by trial and error. To determine the weight values, one must have a set of examples of how the output should relate to the inputs. The task of determining the weights from these examples is called training or learning and is basically a conventional estimation problem. That is, the weights are estimated from the examples in such away that the network, according to metric, models the true relationship as accurately as possible. Since learning is a stochastic process, the learning curve may be drastically different from run to run. In order to compare the performance of a particular search methodology or the effects of different parameters have on a system, it is needed to obtain the average learning curve over the number of runs so that the randomness can be averaged out. An exhaustive and careful experimentation has been carried to determine the configuration of the static MLP model and the optimal proposed FTLRNN model with gamma memory for short-term and long-term ahead predictions for the benchmark chaotic time series for 60% training, 15% cross-validation and 25% testing samples for the considered benchmark chaotic time series. It is found that the performance of the selected model is optimal for 38, 21, 15, and 43 neurons in the hidden layer with regard to the MSE, NMSE, and the correlation coefficient r performance for the testing data sets for Mackey-Glass, Duffing, Monthly Sunspots, and Laser time series, respectively, and the different parameters like transfer function, Learning rule, step size, and momentum values are mentioned in Table 1 for Mackey-Glass and Duffing chaotic time series and in Table 2 for monthly sunspots and Laser time series.

**Table 1.** Optimal Parameters of FTLRNN for both time series.

| Sr. no. | Parameters | Hidden layer for MG | Hidden layer for Duffing | Output layer for MG | Output layer for Duffing |
|---|---|---|---|---|---|
| 1 | Number of PEs | 38 | 21 | 1 | 1 |
| 2 | Transfer function | Tanh | Tanh | Tanh | Lin tanh |
| 3 | Learning rule | Momentum | Momentum | Momentum | Momentum |
| 4 | Step size | 1 | 1 | 0.1 | 0.1 |
| 5 | Momentum | 0.8 | 0.8 | 0.8 | 0.8 |

**Table 2.** Optimal Parameters of FTLRNN for both real-time series.

| Sr. no. | Parameters | Hidden layer for sun spot time series | Hidden layer for laser time series | Output layer for sunspot time series | Output layer for laser time series |
|---|---|---|---|---|---|
| 1 | Number of PEs | 15 | 43 | 1 | 1 |
| 2 | Transfer function | Tanh | Tanh | Tanh | Tanh |
| 3 | Learning rule | Momentum | Momentum | Momentum | Momentum |
| 4 | Step size | 1 | 1 | 0.1 | 0.1 |
| 5 | Momentum | 0.8 | 0.8 | 0.8 | 0.8 |

When we attempted to increase the number of hidden layer and the number of processing element in the hidden layer, the performance of the model is not to seen to improve significantly. On the contrary it takes too long time for training because of complexity of the model. As there is single input and single output for the given system, the number of input and output Processing Elements is chosen as one. Now the NN models are trained three times with different weight initialization with 1000 iterations of the static backpropagation algorithm with momentum term for these two models. All the possible variations for the model such as number of hidden layers, number of processing elements in each hidden layer, different transfer functions like tanh, linear tanh, sigmoid, linear sigmoid in output layer, the different supervised learning rules like momentum, conjugant gradient, and quick propagation are attempted for 10-step ahead prediction for Mackey-Glass, Duffing, and Laser time series, and 6 months ahead prediction for the monthly sunspots time series. The results are placed in Table 3 for Mackey-Glass and Duffing time series and in Table 4 for real-time monthly sunspots, and Laser time series for different learning rules on testing data set.

**Table 3.** Learning rules variations for the Mackey-Glass and Duffing time series.

| Learning rule | Mackey-Glass time series ($K = 10$ step ahead prediction) | | | Duffing time series ($K = 10$ step ahead prediction) | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| Conjugate gradient | 0.00545 | 0.09321 | 0.94965 | 0.12332 | 0.80365 | 0.77838 |
| Momentum | 0.004475 | 0.07940 | 0.9596 | 0.00335 | 0.02185 | 0.98910 |
| Quick propagation | 0.00534 | 0.09486 | 0.9481 | 0.00660 | 0.04305 | 0.97821 |

**Table 4.** Learning rules variations for the real sunspots and Laser time series.

| Learning rule | Real sunspots time series ($K = 6$ months ahead prediction) | | | Real Laser time series ($K = 10$ step ahead prediction) | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| Conjugate gradient | 0.00624 | 0.1135 | 0.9450 | 0.1450 | 0.511140 | 0.79772 |
| Momentum | 0.00554 | 0.1008 | 0.95528 | 0.009563 | 0.337 | 0.8302 |
| Quick Propagation | 0.00866 | 0.15757 | 0.92256 | 0.01787 | 0.62995 | 0.64371 |

Also the various error norms $L_1$, $L_2$, $L_3$, $L_4$, $L_5$, and $L_\infty$, are varied, and FTL-RNN model is trained and tested for the optimum transfer function. The results are obtained and placed in Table 5 for artificial Mackey-Glass and Duffing time series and in Table 6 for real-time monthly sunspots and Laser time series. It is clear from Table 5 that for L2 error norm and tanh transfer function the value of MSE is minimum and correlation coefficient r is maximum for the Mackey-Glass chaotic time series. For Duffing chaotic time series the optimal values of MSE, NMSE, and correlation coefficient r is obtained for linear tanh transfer function and L1 error norm can be seen from Table 5.

**Table 5.** Error norms variations for the Mackey-Glass and Duffing time series for FTLRNN model on testing data set.

| Error norms | Mackey-Glass time series ($K$ = 10 step ahead prediction) Transfer function tanh | | | Duffing time series ($K$ = 10 step ahead prediction) Transfer function linear tanh | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| $L_1$ | 0.00455 | 0.08077 | 0.94576 | 0.00335 | 0.02185 | 0.98910 |
| $L_2$ | 0.00447 | 0.07940 | 0.9596 | 0.00389 | 0.02747 | 0.9831 |
| $L_3$ | 0.00484 | 0.08595 | 0.9345 | 0.0488 | 0.02756 | 0.9821 |
| $L_4$ | 0.00577 | 0.1024 | 0.93823 | 0.01624 | 0.02756 | 0.96108 |
| $L_5$ | 0.00629 | 0.1117 | 0.9332 | 0.01656 | 0.09354 | 0.95725 |
| $L_\infty$ | 0.00544 | 0.0887 | 0.94990 | 0.00341 | 0.01901 | 0.9810 |

**Table 6.** Error norms variations for the real Sunspots time series and laser time series for FTLRNN model on testing data set.

| Error norms | Monthly sunspots time series ($K$ = 6 months ahead prediction) Transfer function tanh | | | Laser time series ($K$ = 10 step ahead prediction) Transfer function tanh | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| $L_1$ (tanh) | 0.02016 | 0.36673 | 0.85915 | 0.006171 | 0.2174 | 0.9014 |
| $L_2$ (tanh) | 0.00554 | 0.1008 | 0.95528 | 0.009563 | 0.337 | 0.8302 |
| $L_3$ (tanh) | 0.00571 | 0.10387 | 0.94882 | 0.0279 | 0.98417 | 0.3038 |
| $L_4$ (tanh) | 0.00956 | 0.17388 | 0.91375 | 0.2430 | 1.18040 | 0.2430 |
| $L_5$ (tanh) | 0.01618 | 0.29439 | 0.85462 | 0.02168 | 0.7641 | 0.5541 |
| $L_\infty$ (tanh) | 0.00653 | 0.11889 | 0.92482 | 0.01536 | 0.5393 | 0.7887 |

Similarly, for real-time monthly sunspots time series, the minimum value of MSE and maximum value correlation relation coefficient r are obtained for $L_2$ error norm and tanh transfer function. For Laser time series the minimum value of MSE and maximum value of correlation coefficient r resulted for L1 error norm and tanh transfer function.

Then on these resulted optimal parameters the FTLRNN model is trained and tested for short-term (1, 5, 10) and long-term (20, 50, and 100) step ahead prediction. The FTLRNN structure is the MLP extended with the short-term memory structures. So these optimal parameters obtained for FTLRNN model

are used for training and testing the MLPNN. Then on the same optimal parameters the static MLPNN model was attempted, and the performance measures like MSE, NMSE, and correlation coefficient r for the short-term (1, 5, and 10) and long-term (20, 50, and 100) step ahead prediction were obtained as stated in Table 7 for Mackey-Glass chaotic time series, in Table 8 for Duffing chaotic time series, in Table 9 for laser time series, and in Table 10 for monthly sunspots time series. It is obvious from Tables 7, 8, 9, and 10 that for all the time series considered for short-term and long-term ahead predictions, the performance of this FTLRNN model is optimal on the test dataset for the following number of taps = 6, Tap Delay = 1, Trajectory Length = 50 with regards to the value of correlation coefficient r, MSE, and NMSE.

**Table 7.** Performance of MLPNN and FTLRNN on testing data for Mackey-Glass time series.

| | MLPNN on testing data | | | FTLRNN on testing data | | |
|---|---|---|---|---|---|---|
| $K$ (step) | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 1 | 0.007 | 0.01 | 0.9933 | 0.000491 | 0.00919 | 0.99542 |
| 5 | 0.0028 | 0.051 | 0.9745 | 0.00227 | 0.04145 | 0.97911 |
| 10 | 0.0056 | 0.1008 | 0.9485 | 0.004475 | 0.07940 | 0.9596 |
| 20 | 0.0128 | 0.2159 | 0.8860 | 0.00944 | 0.15861 | 0.891822 |
| 50 | 0.0411 | 0.599 | 0.6406 | 0.024382 | 0.35655 | 0.80849 |
| 100 | 0.08180 | 0.9988 | 0.1971 | 0.05345 | 0.65267 | 0.62265 |

**Table 8.** Performance of MLPNN and FTLRNN on testing data for Duffing time series.

| | MLPNN on testing data | | | FTLRNN on testing data | | |
|---|---|---|---|---|---|---|
| $K$ (step) | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 1 | 0.08750 | 0.5666 | 0.66035 | 0.00113 | 0.00957 | 0.99545 |
| 5 | 0.07634 | 0.55568 | 0.68434 | 0.00202 | 0.01315 | 0.99356 |
| 10 | 0.08331 | 0.54275 | 0.67991 | 0.00335 | 0.02185 | 0.98910 |
| 20 | 0.08580 | 0.62975 | 0.63454 | 0.00632 | 0.04150 | 0.97982 |
| 50 | 0.10079 | 0.74545 | 0.54593 | 0.00927 | 0.06562 | 0.96663 |
| 100 | 0.11397 | 0.85011 | 0.40191 | 0.00117 | 0.07870 | 0.96000 |

**Table 9.** Performance of MLPNN and FTLRNN for testing data set for Laser time series.

| $K$ (step) | MLP neural network | | | FTLRNN | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 1 | 0.01815 | 0.681 | 0.59818 | 0.00282 | 0.105 | 0.94592 |
| 5 | 0.02522 | 0.918 | 0.35641 | 0.00372 | 0.135 | 0.93962 |
| 10 | 0.01957 | 0.689 | 0.5587 | 0.00956 | 0.337 | 0.9014 |
| 20 | 0.03040 | 1.009 | 0.11287 | 0.01635 | 0.543 | 0.71872 |
| 50 | 0.03499 | 1.039 | 0.28146 | 0.03466 | 1.02971 | 0.41330 |

Then training and testing samples are varied from 10% to 80% as training with the increments of 10% samples and 75% to 5% as testing on the proposed optimal FTLRNN model, the number of training and testing samples was as testing samples with the decrements of 10% keeping exemplars for Cross-Validation

(CV) 15% constant. The performance measures were obtained and compared on testing and training datasets and to gauge the performance and robustness of the FTLRNN. The results are obtained and are placed in Table 11 for 1-step ahead predictions, Table 12 for 5-step ahead prediction, Table 13 for 10-step ahead prediction, Table 14 for 20-step ahead prediction, Table 15 for 50-step ahead prediction and Table 16 for 100-step ahead prediction for Mackey-Glass and Duffing chaotic time series.

**Table 10.** Performance of MLPNN and FTLRNN for testing data set for monthly sunspot time series.

| *K* (month) | MLP neural network | | | FTLRNN | | |
|---|---|---|---|---|---|---|
| | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 1 | 0.00247 | 0.04533 | 0.97569 | 0.002227 | 0.04162 | 0.98163 |
| 6 | 0.01229 | 0.2234 | 0.8917 | 0.00554 | 0.1008 | 0.9528 |
| 12 | 0.02387 | 0.43132 | 0.77184 | 0.01693 | 0.30599 | 0.8550 |
| 18 | 0.03989 | 0.7174 | 0.5741 | 0.02266 | 0.4075 | 0.7977 |
| 24 | 0.0538 | 0.9658 | 0.3584 | 0.03059 | 0.54899 | 0.71497 |

**Table 11.** For K = 1, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.000315 | 0.00627 | 0.99707 | 0.00230 | 0.03972 | 0.98015 |
| 20% | 65% | 0.000247 | 0.004854 | 0.99757 | 0.00101 | 0.00774 | 0.99618 |
| 30% | 55% | 0.000405 | 0.007732 | 0.99628 | 0.00104 | 0.00707 | 0.99671 |
| 40% | 45% | 0.000371 | 0.00691 | 0.99663 | 0.00103 | 0.00655 | 0.99692 |
| 50% | 35% | 0.000371 | 0.007283 | 0.9964 | 0.00087 | 0.00555 | 0.99733 |
| 60% | 25% | 0.000491 | 0.00919 | 0.99542 | 0.00113 | 0.00957 | 0.99545 |
| 70% | 15% | 0.00078 | 0.01353 | 0.9933 | 0.00091 | 0.00609 | 0.99712 |
| 80% | 05% | 0.002217 | 0.03312 | 0.98337 | 0.00484 | 0.03271 | 0.98827 |

**Table 12.** For K = 5, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.000501 | 0.01779 | 0.99122 | 0.00465 | 0.08523 | 0.95975 |
| 20% | 65% | 0.001006 | 0.019508 | 0.9902 | 0.00162 | 0.01236 | 0.99388 |
| 30% | 55% | 0.001196 | 0.02256 | 0.98875 | 0.00150 | 0.00944 | 0.99552 |
| 40% | 45% | 0.00144 | 0.02653 | 0.98675 | 0.00139 | 0.00885 | 0.99578 |
| 50% | 35% | 0.00167 | 0.0322 | 0.9838 | 0.00170 | 0.01154 | 0.99435 |
| 60% | 25% | 0.00227 | 0.04145 | 0.97911 | 0.00202 | 0.01315 | 0.99356 |
| 70% | 15% | 0.00371 | 0.06227 | 0.9684 | 0.00149 | 0.00999 | 0.99524 |
| 80% | 05% | 0.01089 | 0.1502 | 0.9226 | 0.00435 | 0.02941 | 0.98926 |

**Table 13.** For K = 10, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.00191 | 0.03733 | 0.9812 | 0.00804 | 0.15432 | 0.92405 |
| 20% | 65% | 0.00208 | 0.03994 | 0.97985 | 0.00314 | 0.02395 | 0.98796 |
| 30% | 55% | 0.02261 | 0.04205 | 0.97888 | 0.00202 | 0.01315 | 0.99356 |
| 40% | 45% | 0.00266 | 0.04815 | 0.97581 | 0.00260 | 0.01653 | 0.99185 |
| 50% | 35% | 0.03448 | 0.06483 | 0.967219 | 0.00288 | 0.01936 | 0.99014 |
| 60% | 25% | 0.004475 | 0.07940 | 0.9596 | 0.00335 | 0.02185 | 0.98910 |
| 70% | 15% | 0.00757 | 0.12238 | 0.9373 | 0.00306 | 0.02046 | 0.99001 |
| 80% | 05% | 0.02204 | 0.27796 | 0.8537 | 0.00655 | 0.04429 | 0.97835 |

**Table 14.** For K = 20, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.00407 | 0.07799 | 0.96028 | 0.01342 | 0.30048 | 0.86374 |
| 20% | 65% | 0.00436 | 0.08154 | 0.95852 | 0.00503 | 0.03857 | 0.98132 |
| 30% | 55% | 0.00469 | 0.08499 | 0.95721 | 0.00732 | 0.04650 | 0.96982 |
| 40% | 45% | 0.00578 | 0.10149 | 0.948802 | 0.00427 | 0.02725 | 0.98192 |
| 50% | 35% | 0.00707 | 0.127302 | 0.93491 | 0.00516 | 0.03518 | 0.98270 |
| 60% | 25% | 0.00944 | 0.15861 | 0.91822 | 0.00632 | 0.04150 | 0.97982 |
| 70% | 15% | 0.01535 | 0.23154 | 0.87938 | 0.00492 | 0.03154 | 0.98420 |
| 80% | 05% | 0.04379 | 0.47688 | 0.74082 | 0.00662 | 0.04481 | 0.97861 |

**Table 15.** For K = 50, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.01082 | 0.19456 | 0.89823 | 0.01290 | 0.79033 | 0.71021 |
| 20% | 65% | 0.01169 | 0.20374 | 0.89309 | 0.01027 | 0.08318 | 0.95984 |
| 30% | 55% | 0.01316 | 0.22115 | 0.88438 | 0.00658 | 0.04203 | 0.97902 |
| 40% | 45% | 0.01529 | 0.24709 | 0.87032 | 0.00763 | 0.05034 | 0.97470 |
| 50% | 35% | 0.01816 | 0.29049 | 0.84538 | 0.00781 | 0.05426 | 0.97271 |
| 60% | 25% | 0.024382 | 0.35655 | 0.80849 | 0.00927 | 0.06562 | 0.96663 |
| 70% | 15% | 0.03829 | 0.48826 | 0.73543 | 0.00743 | 0.05020 | 0.97567 |
| 80% | 05% | 0.1056 | 0.87428 | 0.52529 | 0.00953 | 0.06451 | 0.96811 |

**Table 16.** For K = 100, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Mackey-Glass time series | | | Duffing time series | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | *r* | MSE | NMSE | *r* |
| 10% | 75% | 0.029161 | 0.47623 | 0.7294 | 0.01359 | 0.54149 | 0.80092 |
| 20% | 65% | 0.02897 | 0.45601 | 0.74595 | 0.00927 | 0.07219 | 0.96339 |
| 30% | 55% | 0.03182 | 0.48156 | 0.72938 | 0.00116 | 0.07810 | 0.962300 |
| 40% | 45% | 0.03439 | 0.50398 | 0.71524 | 0.00875 | 0.05663 | 0.97196 |
| 50% | 35% | 0.04027 | 0.54749 | 0.68781 | 0.00883 | 0.06062 | 0.96947 |
| 60% | 25% | 0.05345 | 0.65267 | 0.62265 | 0.00117 | 0.07870 | 0.96000 |
| 70% | 15% | 0.08017 | 0.81243 | 0.53108 | 0.01033 | 0.06935 | 0.96570 |
| 80% | 05% | 0.21006 | 1.47654 | 0.24862 | 0.01141 | 0.07715 | 0.96069 |

In a similar way, for real-time monthly sunspots and Laser time series, the number of training and testing samples was varied from 10% to 80% as training with the increments of 10% samples and 75% to 5% as testing samples with the decrements of 10% keeping exemplars for Cross-Validation (CV) 15% constant. The performance measures were obtained and compared on testing and training data sets and to gauge the performance and robustness of the FTLRNN. The obtained results are placed in Tables 17, 18, 19, 20, and 21 for monthly sunspots and Laser time series for the value K in K-step ahead prediction.

**Table 17.** For K = 1, training and testing samples variation for FTLRNN on testing data set.

| Time series | | Sunspot time series | | | Laser time series $K = 1$ | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 10% | 75% | 0.00320 | 0.08705 | 0.96121 | 0.00777 | 0.24076 | 0.87475 |
| 20% | 65% | 0.00258 | 0.6819 | 0.96605 | 0.01042 | 0.30799 | 0.84821 |
| 30% | 55% | 0.00188 | 0.04608 | 0.97791 | 0.01054 | 0.32361 | 0.84657 |
| 40% | 45% | 0.00218 | 0.05092 | 0.97928 | 0.00991 | 0.39119 | 0.81057 |
| 50% | 35% | 0.00273 | 0.05795 | 0.97837 | 0.00185 | 0.09233 | 0.98836 |
| 60% | 25% | 0.00227 | 0.04162 | 0.98163 | 0.00282 | 0.10597 | 0.94592 |
| 70% | 15% | 0.00336 | 0.07416 | 0.96799 | 0.00080 | 0.02293 | 0.98873 |
| 80% | 05% | 0.00153 | 0.05069 | 0.97512 | 0.00184 | 0.04229 | 0.97872 |

**Table 18.** For K = 6, training and testing samples variation for FTLRNN on testing data set for monthly sunspot time series and K = 5 for laser time series.

| Time series | | Sunspot time series $K = 6$ Months | | | Laser time series $K = 5$ | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 10% | 75% | 0.00707 | 0.19197 | 0.91703 | 0.00782 | 0.24068 | 0.87439 |
| 20% | 65% | 0.00451 | 0.11888 | 0.94153 | 0.00811 | 0.23969 | 0.88027 |
| 30% | 55% | 0.00423 | 0.10308 | 0.94712 | 0.00808 | 0.24814 | 0.87504 |
| 40% | 45% | 0.00438 | 0.10197 | 0.95081 | 0.00635 | 0.25307 | 0.87748 |
| 50% | 35% | 0.00654 | 0.13744 | 0.94252 | 0.00267 | 0.12898 | 0.94513 |
| 60% | 25% | 0.00554 | 0.1008 | 0.95528 | 0.00372 | 0.13558 | 0.93962 |
| 70% | 15% | 0.00473 | 0.10259 | 0.95359 | 0.00534 | 0.14838 | 0.92717 |
| 80% | 05% | 0.00499 | 0.15725 | 0.92195 | 0.01579 | 0.34297 | 0.83272 |

**Table 19.** For K = 12 , training and testing samples variation for FTLRNN on testing data set for monthly sunspot time series and K = 10 for laser time series.

| Time series | | Sunspot time series $K = 12$ | | | Laser time series $K = 10$ | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 10% | 75% | 0.01187 | 0.32145 | 0.87483 | 0.01506 | 0.45940 | 0.76210 |
| 20% | 65% | 0.01116 | 0.29307 | 0.84917 | 0.01310 | 0.38332 | 0.80583 |
| 30% | 55% | 0.00967 | 0.23485 | 0.88221 | 0.00140 | 0.43507 | 0.77113 |
| 40% | 45% | 0.00749 | 0.17299 | 0.91236 | 0.01221 | 0.48119 | 0.75879 |
| 50% | 35% | 0.00909 | 0.19003 | 0.91050 | 0.00444 | 0.20709 | 0.89826 |
| 60% | 25% | 0.01693 | 0.30599 | 0.8550 | 0.00956 | 0.337 | 0.9014 |
| 70% | 15% | 0.00841 | 0.17893 | 0.92008 | 0.01047 | 0.28229 | 0.87301 |
| 80% | 05% | 0.00871 | 0.26119 | 0.86836 | 0.02631 | 0.59165 | 0.71077 |

**Table 20.** For K = 18 , training and testing samples variation for FTLRNN on testing data set for monthly sunspot time series and K = 20 for laser time series.

| Time series | | Sunspot time series $K = 18$ | | | Laser time series $K = 20$ | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing exemplars | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 10% | 75% | 0.01804 | 0.48780 | 0.74535 | 0.02676 | 0.80529 | 0.58788 |
| 20% | 65% | 0.01853 | 0.48428 | 0.75209 | 0.02361 | 0.68739 | 0.67208 |
| 30% | 55% | 0.01287 | 0.31169 | 0.82991 | 0.02569 | 0.79376 | 0.56198 |
| 40% | 45% | 0.01403 | 0.32187 | 0.83052 | 0.01659 | 0.66263 | 0.65482 |
| 50% | 35% | 0.02792 | 0.57914 | 0.73685 | 0.00703 | 0.30705 | 0.85539 |
| 60% | 25% | 0.02266 | 0.04075 | 0.7977 | 0.01635 | 0.543 | 0.71872 |
| 70% | 15% | 0.01885 | 0.39372 | 0.78804 | 0.03573 | 0.37992 | 0.37992 |
| 80% | 05% | 0.00745 | 0.21530 | 0.89769 | 0.00323 | 0.59234 | 0.59234 |

**Table 21.** For K = 24, training and testing samples variation for FTLRNN on testing data set for monthly sunspot time series and for laser time series (K = 50).

| Time series | | Sunspot time series $K = 24$ | | | Laser time series $K = 50$ | | |
|---|---|---|---|---|---|---|---|
| Training exemplars | Testing Exemplars | MSE | NMSE | $r$ | MSE | NMSE | $r$ |
| 10% | 75% | 0.03246 | 0.37734 | 0.49224 | 0.04073 | 1.18215 | 0.13264 |
| 20% | 65% | 0.03657 | 0.58297 | 0.43616 | 0.04310 | 1.23643 | -0.00980 |
| 30% | 55% | 0.02280 | 0.52882 | 0.69373 | 0.03018 | 0.98884 | 0.27393 |
| 40% | 45% | 0.01839 | 0.83622 | 0.77659 | 0.04425 | 1.87215 | -0.22273 |
| 50% | 35% | 0.04066 | 0.41935 | 0.53973 | 0.02534 | 0.94845 | 0.26601 |
| 60% | 25% | 0.03019 | 0.54899 | 0.71497 | 0.03466 | 1.02971 | 0.41330 |
| 70% | 15% | 0.02834 | 0.95150 | 0.75528 | 0.03447 | 0.89215 | 0.44330 |
| 80% | 05% | 0.02339 | 0.57734 | 0.8012 | 0.08839 | 0.96548 | 0.28132 |

Next for the optimum values of performance measures obtained from the combinations data partition as training and testing samples for the chaotic time series as mentioned in Tables 11 to 16 for the Mackey-Glass chaotic time series for all the cases of multistep ahead prediction and Duffing time series all the step ahead prediction. Similarly, for the real-time monthly sunspots and Laser time series for all the multistep ahead prediction.

Then for the optimal data partition of training and testing samples combination resulted for the K-step ahead prediction for the cases of K-value in K-step ahead prediction for all the considered time series. For those combinations, the number of epochs is varied from 2000 to 20,000 in a step of 2000, and again the proposed FTLRNN model is trained to observe the more prominent values of performance measures for all the chaotic time series and for all the steps ahead prediction.

## Discussion

It can be clearly observed that dynamic FTLRNN model with gamma memory clearly outperforms the static MLP not only for short-term prediction but also for long-term prediction for testing data as well as training data set. From the results of Table 7, for Mackey-Glass chaotic time series, it is noticed that up to the 20-step ahead prediction the Performance measures values of MLP and dynamic FTLRNN are slightly deviating but for long 50- and 100-step ahead prediction the Performance measures values of MSE, NMSE, and correlation coefficient (r) for FTLRNN model are significantly improved as compared to the static MLP. For the Duffing time series from Table 8 it is observed that for short- and long-step ahead predictions the proposed dynamic FTLRNN with gamma memory filter clearly outperforms well as compared to the static MLP with regards to the performance metrics like MSE, NMSE, and correlation coefficient r.

Also for the real monthly sunspots time series, it is observed from Table 9 that for the 1, 6, 12, months ahead predictions the performance metrics values of MLP and dynamic FTLRNN are slightly deviating but for 18 and 24 months ahead prediction, the performance metrics values for FTLRNN are significantly improved as compared to the static MLP. For the Laser time series from Table 10 it is observed that for short- and long-term ahead (K=1, 5, 10, 20, 50) prediction the proposed dynamic FTLRNN with gamma memory filter clearly outperforms well as compared to the static MLP for the values of MSE, NMSE, and correlation coefficient r.

Next for the resulted optimal parameters the FTLRNN model is trained for different training and testing samples combinations varying from 10% to 80% as a training sample and 75% to 5% as a testing sample and Keeping cross-validation samples constant equal to 15% for short-term and long-term step ahead predictions for finding robustness of model and obtaining the significant results of performance measures for training and testing samples combinations. Tables 11 to 16 show the performance metrics values for one-step ahead to 100-step ahead predictions for different combination of training and testing samples for the equation generated Mackey-Glass and Duffing time series.

Similarly for real-time series the training and testing samples are varied in combination as mentioned in Tables 17 to 21 for one month ahead to twenty four months ahead predictions for monthly sunspot time series and for one-step to fifty-step ahead predictions for Laser time series.

For which training and testing samples the values of MSE NMSE are minimum, and correlation coefficient is closed to unity for short-term and long-term predictions for the chaotic time series for that training and testing samples the FTLRNN with gamma filter is trained for 2000 to 20,000 epochs in the steps of 2000 for obtaining more significant values and observing the network performance with regards to the performance measures. The results are plotted in Figure 6 for 1- and 5-step ahead predictions, Figure 7 for 10- and 20-step ahead predictions and Figure 8 for 50- and 100-step ahead predictions for performance on MSE and NMSE due to epochs variation, and Figure 9 for performance on regression due to epochs variation for 1-, 5-, 10-, 20-, 50- and 100-step ahead predictions for Mackey-Glass chaotic time series. It is observed that up to 50 steps ahead prediction the performance metrics are slightly deviating but for long-term 100-step ahead predictions above the values of 12 000 epochs training the performance measure like MSE and NMSE values are decreased, and the correlation coefficient r is substantially increased.

Also it can be visually inspected closely from Figures 10, 11, 12, and 13 that the output of the proposed FTLRNN model closely follows the desired output for 1-, 5-, 10-, 20-step ahead predictions.

From Figure 14 and Figure 15, it is clear that, the output of the network is slightly deviating from the desired output for long 50-step and 100-step ahead predictions.

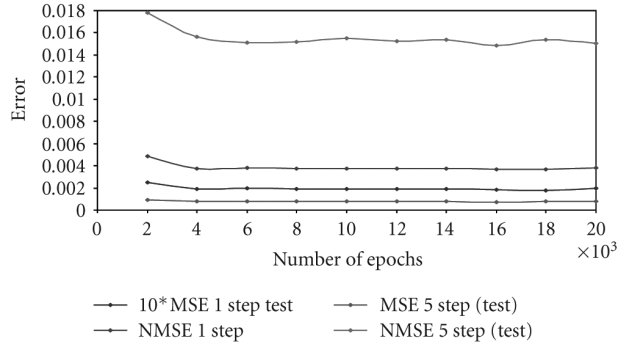


**Figure 6.** Performance of epoch's variation on Errors for 1 and 5 step for Mackey-Glass time series.

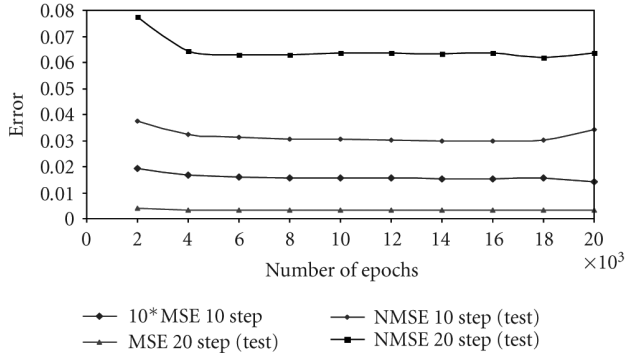


**Figure 7.** Performance of epoch's variation on Errors for 10 and 20 step for Mackey-Glass time series.

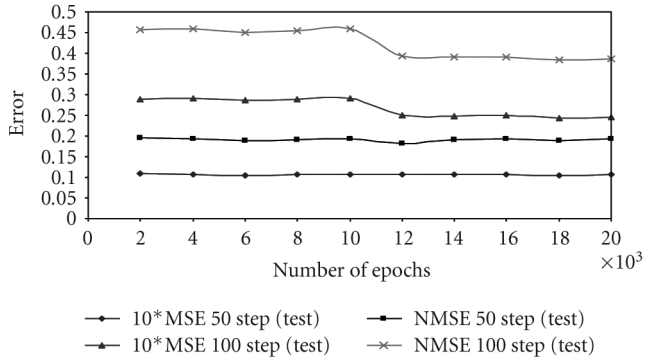


**Figure 8.** Performance of epoch's variation on Errors for 50 and 100 step for Mackey-Glass time series.

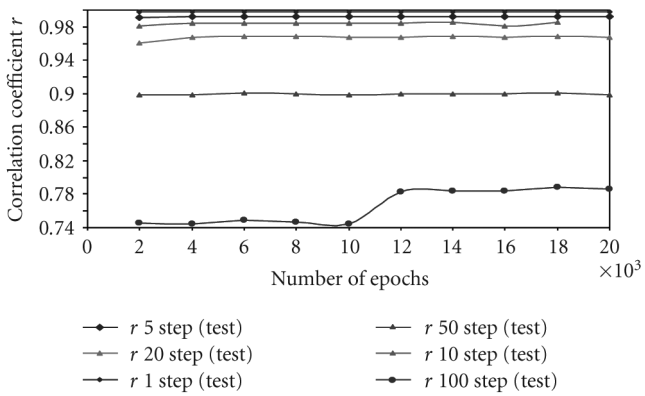


**Figure 9.** Performance of epochs variation on Correlation Coefficient r for 1, 5, 10, 20, 50, and 100 step for Mackey-Glass time series.

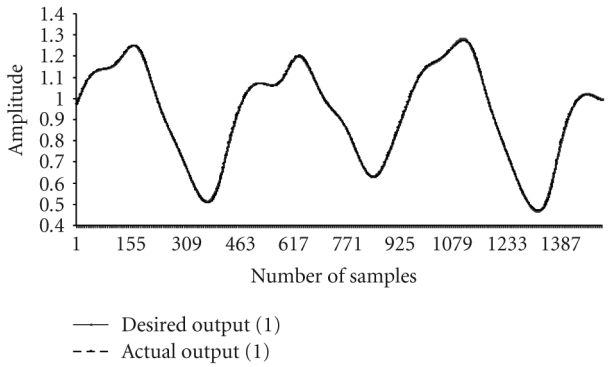


**Figure 10.** Desired output and network output for 1-step ahead Prediction for Mackey-Glass time series.

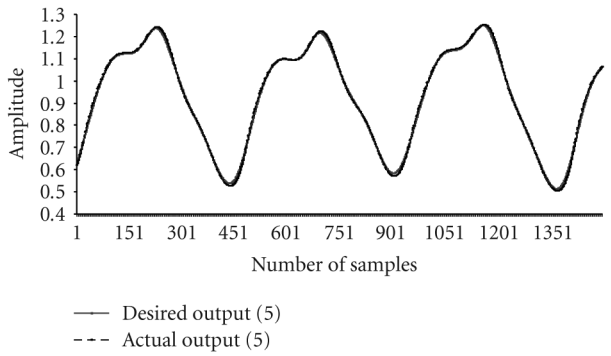


**Figure 11.** Desired output and FTLRNN output for 5-step ahead prediction for Mackey-Glass time series.