SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

ADDIS ABABA INSTITUTE OF TECHNOLOGY

ADDIS ABABA UNIVERSITY

NARX-Based Locally Distributed Web-Servers Load Prediction

The case of ethio telecom

By

Berta Etefa

Advisor

Dr. Mesfin Kifle

A thesis Submitted in partial fulfillment of the requirements of the Degree of Masters of Science in Telecommunication Engineering

February, 2020

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

TELECOMMUNICATION ENGINEERING GRADUATE PROGRAM

NARX-Based Locally Distributed Web-Servers Load Prediction

The case of ethio telecom

By: Berta Etefa

Approved by:

_____	_____	_____
Chairman, School Graduate Committee	**Signature**	**Date**
Dr. Mesfin Kifle	_____	_____
Advisor	**Signature**	**Date**
Dr. Yalemzewd Negash ___	_____	_____
Evaluator	**Signature**	**Date**
Dr. Surafel Lemma ___	_____	_____
Evaluator	**Signature**	**Date**

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Berta Etefa _____

Name Signature

Place: Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Mesfin Kifle _____

Advisor's Name Signature

Abstract

With the continuous development of network technology, the need to have a page view of users and the load on servers has grown exponentially, resulting in temporary loss of services. Distributed computing systems are becoming a widely used paradigm to provide high performance and uninterrupted services to users. In such system, it is crucial to use effective resource management techniques to handle a large number of requests and provide dependable services with high quality constantly. Indeed, both service interruptions and resource waste can be reduced with the implementation of an effective prediction system. One promising approach is realizing artificial neural network algorithm to predict resource usage series of server. Several research has been conducted using different combination of load descriptor to predict resource usage series of server. Yet, queue time, is believed to be a good load descriptor of a server, because it gives a good estimate of job response time. It strives to produce a global improvement in system performance. However, this load descriptor, still not be seen in the study of resource usage prediction of server. Thus, in this research, we have investigated nonlinear autoregressive network with exogenous inputs (NARX) neural network multi-step ahead predictability of web server load. Besides it, three different training algorithms: Lavenberg-Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG) forecasting accuracy were evaluated. We have collected resource usage series of a week data from locally distributed web servers of ethio telecom business support system. Two Cases (Case-1 and Case-2) of experiments were conducted to evaluate the performance of the algorithms; using as input in the first Case CPU and memory, and in the second Case CPU, memory and queue time. MATLAB was employed to verify the prediction accuracy of the algorithms. The results of the simulation show that for 12-step ahead web server load prediction, LM learning algorithm, Case-2 approach has registered the best prediction accuracy with MAPE of 4.459%, followed by the BR learning algorithm with MAPE of 4.649%. SCG was the lowest performer with MAPE of 5.610%. Thus, accuracy in prediction is necessary since the more efficient resources can be managed in data centers. Therefore, having such a model, enhances the process of working towards reliable services.

Keywords NARX neural network, training algorithm, web server load, multi-step ahead prediction.

Acknowledgment

First and foremost, I would like to thank God (α and Ω), his holy mother (Kidist Mariam), for the good health and well-being that were necessary to complete this thesis.

I would like to express my deep gratitude and respect to my advisor Dr. Mesfin Kifle for his patience, motivation, immense knowledge and friendly advice, which helped me a lot during the period of my work. I would also like to thank AAU instructors specially to Dr. Yalemzewd Negash, and Dr. Surafel Lemma, for their questions, insightful comments, and encouragements in all stages of this thesis.

I am also indebted to my friends and colleagues, especially Tade, and Mr. Werku as they were very cooperative in providing the necessary data for this work and as they were very encouraging throughout the whole process.

Finally, I must express my very profound gratitude to my beloved family: my wife Beza, my lovely kids 'Liyaye' and 'Mamushe'; for their continuous love, endless support, and motivation. They were by my side by giving me extreme support throughout my years of study. This accomplishment would not have been possible without you. Thank you all.

Table of Contents

List of Tables

List of Figures

Acronyms

ANN	Artificial Neural Networks
ARIMA	Auto Regressive Integrated Moving Average
ARX	Auto Regressive with exogenous input
BP	Back Propagation
BP NN	Back Propagation Neural Network
BR	Bayesian Regularization
BSS	Business Support System
CPU	Central Processing Unit.
FBNN	Feed-backward Neural Network
FFNN	Feed-forward Neural Network
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
LM	Lavenberg-Marquardt
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
NARX NN	Nonlinear Autoregressive network with exogenous inputs Neural Network
NN	Neural Network
OS	Operating System
R	Correlation coefficient
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SAN	Storage Area Network
SCG	Scaled Conjugate Gradient
SVM	Support Vector Machine
T	Response time
TDNN	Time Delay Neural Network
VM	Virtual Machine
WAN	Wide Area Network

1. Introduction

"The growth of web-based applications in business and e-commerce is building up demands for high performance web servers for better throughputs and lower user perceived latency. These demands are leading to a widespread substitution of powerful single servers by robust newcomers, cluster web servers, in many enterprise companies" [6]. Distributed computing systems are becoming a widely used paradigm for achieving high availability, scalability and performance. Such systems are usually formed as a group of nodes providing the same set of services. The cluster of servers operates transparently to clients and incoming requests are distributed among the nodes using a load balancing algorithm.

In a distributed computing environment, to handle a large number of requests and provide dependable services with constantly, it requires an effective load balancing mechanisms to distribute the client workload equitably among the back end servers to improve overall responses. Due to uneven job arrival patterns and unequal computing capacities and capabilities of computing nodes load balancing is required to distribute the dynamic workload evenly across all the nodes to achieve a high user satisfaction and resource utilization by ensuring an efficient and fair allocation of every computing resource [11].

The goal of load balancing is assigning a number of tasks proportionally among nodes of the distributed system to improve both resource utilization and job response time while also avoiding a situation where some nodes are heavily loaded while others are idle or doing little work [10].

According to Wang et al [25], load balancing mechanisms can be provided in network based, middleware based and OS based approach. In network based load balancing approach, load balancing is performed at the network layer (Layer 3) or transport layer (Layer 4). This approach is often used in many distributed system applications such as web sites. Load balancing performed in Middleware's, often on a per-session or a per-request basis, to balance the workload more efficiently. In this type of load balancing (middleware based), many of the existing approaches uses the control theory method or damping technology to predict the server load [4]. Which depends on a fixed factor in computing the load. And this damping factor should be adjusted dynamically according to the load condition to avoid abnormal state of the server. As a result, a new prediction method is required that take into account the load condition dynamically to achieve adaptive load balancing.

Recently, a number of researchers offers prediction based dynamic load balancing techniques to improve the overall utilization of resources and in turn to improve the system performance. Effective workload characterization and prediction hold the answers to the conundrum of efficient resource allocation in distributed and scaled out systems. Being able to accurately predict the upcoming workload within the next time frame allows the system to make proactive decisions rather than reactive ones [3]. Forecasting involves the process of making predictions of the future based on the past and the present analysis of data trends.

Over the last few years, forecasting has become a dynamic study area which has attracted attentions of research communities. The main aim of time series forecasting is to gather the past observations (data) to develop the best model that accurately fits the data and then generate the future data. Time series forecasting thus can be termed as the act of predicting the future by understanding the past time series data [12, 19].

Resource consumption of the server in the form of CPU, memory, network bandwidth, and disk space, is a useful piece of information when available before execution of an application. It can be used by the scheduler to accommodate the most number of applications without resource contention, it can help to estimate the waiting time on queued systems, it can help to identify the best resource to run an application and analyze what-if scenarios [26]. Thus, this desire has led researchers to explore the predictability of resource usage patterns to support improvements in load balancing, managing disk IO, scheduling processes, and quality of service [7]. The use of Machine Learning (ML) algorithms to predict these series of resource consumption is becoming an appealing approach.

These days' neural networks, are gaining renewed interest among researchers and they are replacing many practical implementation of the forecasting systems, previously based on static methods [17]. In this thesis work, nonlinear autoregressive neural network with exogenous inputs (NARX) based time series forecasting model was examined to predict usage series of the web server load such as CPU, memory and queue time. Besides it, three different neural network learning algorithms (Lavenberg-Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG)) in forecasting future server load accuracy were evaluated.

Resource prediction is not a new area of study, but continues to be active because of the promise it holds to improve utilization in so many areas of computing [6]. Currently, there are different methods of machine learning approach proposed by scholars to predict the

server load in future time. Some research activities have been made in this area based on Support Vector Machine (SVM) method and others use dynamic Neural Networks (NNs). The reason for selecting dynamic NNs for server load prediction is the ability of this approach to accurately forecast nonlinear time series data. In addition, dynamic NNs can be more compact and hence faster to evaluate than SVM approach [4]. Jina et al [2] proposed composite model by combining the back propagation (BP) neural network with autoregressive integrated moving average (ARIMA) model to improve the predictive result. According to the authors, the new combined model has great advantages in processing the feature undefined character of linear and nonlinear time sequence part of the data.

1.1 Motivation

Computing systems enable the Telecom operators to provide their customers with a vast variety of services which are aimed to meet their demands and desires. Telecommunication applications are often based on a multi-tier architecture, with web servers, application servers and database servers. An operator usually uses a network of several such computing systems to facilitate providing the end users with an ever growing variety of services [22]. Most companies heavily rely on technologies that are of different type and on enterprise systems which are running in different business units and technologies (i.e. platform, infrastructure, language, etc.).

The company, ethio telecom, currently has a number of data centers in Addis Ababa and in regions, and one disaster recovery site in Mekele. The company has installed different enterprise systems on top of these data centers to provide service to its customers. The systems are categorized as business support system, operation support system, corporate applications and office tools. Moreover, these applications shall also be grouped into a form of some similar behavior or use class. One possible grouping of applications might be business critical, mission critical, and others.

The IT infrastructure and application systems in ethio telecom, which have been deployed in these data centers, are large, heterogeneous and complex. Besides, the business-critical and mission-critical services require high level of service resilience. To achieve this, the company uses some of the technologies that can alleviate system performance and availability of these systems. For this purpose, the company has been deploying high availability techniques based on active/standby model as well as locally distributed model on multiple machines in the

same data center [33]. In addition, the company uses I2000 to check the healthiness of the servers and system resources. I2000 is a monitoring tool adopted by the company in order to monitor resources including application, operating system, servers, processes, network devices, services, hosts, and databases [32].

In the company, ethio telecom, IT infrastructure resources are provisioned to application service based on resource requirement and sizing during system design phase [33]. However, inefficient resource utilization is one of the major challenges due to uneven distribution between resources of different systems and the random nature of application and user requirements. As a result, the implemented system will be either over-utilized or under-utilized in resource usage which deviates from the sizing needs. Moreover, getting computing resources whenever required as to shorten the projects lifecycle and to make the Time-To-Market (TTM) more predictive, is another challenging task for the company.

These days, service providers have been applying different server load prediction techniques to improve the performance, availability, and utilization of resource in order to provide services for a large number of online users. The more accurate resource prediction, the more efficient resources can be managed in data centers [3, 4, 17].

1.2 Statement of the Problem

Application resource demand comes with different nature that create uncertainty situations for efficient resource matching and workload on resources. Inefficient resource utilization has not only a direct negative effect on performance but also has an indirect effect on system functionality due to poor performance [5]. To deal with such unpredictable load patterns it is desirable to use an effective predicting method to determine the available system resources on each computing nodes. Hence, knowing the resource utilization in advance help us to control proactively the distribution of the incoming task towards the desired resources.

A number of scholars used different load descriptor to predict future loads of the servers. The authors in [2], used CPU utilization to forecast the future time CPU usage of the server. Aljabari et al [4], preferred CPU, memory, and total number of processes to predict the future values of CPU utilization of the server. In similar fashion, Xue et al [3], used CPU, memory, disk, and network bandwidth workload trace to predict future loads of the server. However, Alakeel [10] argues, queue time is a good load descriptor of a server, because it gives a good estimate of job response time. A request to run a job is not serviced immediately but instead

placed in a queue and serviced only when resources are released by preceding jobs. Despite that, the author claims, it is better to predict this queue time interval in advance when there is job resource requirement since, the accuracy of server load prediction varies depending on the used prediction methods and the selected parameters of the server load descriptors [1, 3].

Still, resource utilization prediction remains an active area of research due to the difficulty in accurate forecasting because an accurate understanding of resource utilization can improve performance optimization, load balancing, and recognition of overload conditions [6]. Thus, the particular interest of this research is to realize a multi-step ahead prediction model for ethio telecom web server load of business support systems. Since such a challenge is the major concern of the company to provide reliable services.

1.3 General Objective

The general objective of this thesis work is to come up with a model for multi-step ahead prediction of web server load based on NARX neural network.

1.4 Specific Objectives

The following specific objectives are formulated to realize the general objective.

- ✓ Study and understand the basics of NARX neural network and the selected learning algorithms (LM, BR and SCG).
- ✓ Collect the web server resource usage series data.
- ✓ Simulate the data with NARX neural network using LM, BR and SCG learning algorithms to make a multi-step ahead prediction.
- ✓ Evaluate and compare the prediction accuracy using performance parameters of mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), response (validation) time and regression value.

1.5 Methodology

Initially, review on previous related studies is used for the indication of the importance and significance of prediction models for usage series of server load. The first phase of the research includes obtaining of data from web servers of ethio telecom business support system. The collected sample data was checked whether it was a time series or not, to make the prediction in the next step. Then, the data was preprocessed and analyzed using the

NARX neural network forecasting model through evaluation of three different learning algorithms simulated in MATLAB. Finally, by using the best performed learning algorithms, the future usage series of the web server resource was predicted (see detailed proposed methodological approach in Chapter 4).

1.6 Scope and Limitation

1.6.1 Scope of the Thesis

This thesis addresses the predictability of resource usage series of web server taking ethio telecom as a case study. The scope of this thesis is to predict multi-step ahead resource usage of the web server; using NARX neural network time series prediction method through comparing three different training algorithms, LM, BR and SCG, and select the best one with minimum prediction error.

1.6.2 Limitation of the Thesis

This thesis has some constraint mainly due to the limitation of the company's, ethio telecom, monitoring tool (I2000), the prediction is done only in five-minute time of intervals. In addition to this, the prediction model is built only around the web server tier.

1.7 Significance of the Thesis

The thesis has two sided importance. On the first, the NARX NNs based web server load prediction aids the administrator to dimension the expected resource loads. As a result, services could be more properly sized to IT infrastructures and provide growth projections for a system retirement or upgrade, in addition to, IT resource planning. On the other side, as the first of its kind, it creates motivation on applying NARX NNs based IT resource usage prediction in the company, ethio telecom.

1.8 Thesis Organization

The remainder of the thesis are organized as follows. Chapter 2, presents the theoretical background of time series, forecasting models, artificial neural network, taxonomy of artificial neural network architecture and neural network training algorithms. In Chapter 3, related works regarding different machine learning based server load prediction algorithms are discussed. Following that, the proposed forecasting method, its phase details are provided in Chapter 4. Afterwards, in Chapter 5, the results of the simulation were analyzed and

discussed. Lastly, in Chapter 6, conclusive remarks are drawn and potential future work is presented.

2. Theoretical Background

This chapter presents the theoretical notion of time series, forecasting models, artificial neural network, taxonomy of artificial neural network architecture, nonlinear autoregressive neural network with exogenous inputs and neural network training algorithms. This helps to understand the key aspects, technical definitions and their significances.

2.1 Time Series

Time series is a sequential set of data points, measured typically over successive times. The term “time series” itself, denotes a data storing format, which consists of the two mandatory components; time units and the corresponding value assigned for the given time unit. Values of the series need to denote the same meaning and correlate among the nearby values [12].

A time series containing records of a single variable is termed as univariate. But if records of more than one variable are considered, it is called multivariate. The record of a time series is a discrete set of numbers that may arise from instantaneous sampling of continuous process or when the original source is discrete in nature. Despite the fact that, many scientific processes produce continuous time series data, numerical approach of computer systems allows to store data only as the discrete values. Thus all further forecasting methods performed on computer, assume test data in the discrete values form.

In the study of time series, a suitable model is fitted to the known data values (time series) and the corresponding parameters are estimated from these data values. In time series prediction, past observations are collected and analyzed to develop a model which captures the underlying data generating process for the series [15]. The future events are then predicted using the model. In general, time series prediction is the use of models that forecast future values of time series variables by extrapolating trends and patterns in the past values of the series or by extrapolating the effect of other variables on the series. To date, application of artificial neural networks (ANN) to time-series forecasting has been growing rapidly due to several unique features of ANN models. More details about this model can be found in [13].

2.2 Forecasting Models

Forecasting is the process of making predictions of the future based on the past and the present and analysis of data trends. Time series forecasting belongs to most important analysis methods, performed over the time series data. “The general idea is based on the fact, that information about the past events can be eff ectively exploited to create predictions about the future events” [12].

Scholars have proposed different forecasting models to resolve various problems such as a multi-step ahead response time prediction for database server using nonlinear autoregressive neural network with exogenous inputs (NARX) [22], network traffic prediction for adaptive load balancing strategy using radial basis function (RBF) neural network [28], CPU utilization prediction for virtual machine (VM) migration in cloud computing using composite model combining autoregressive integrated moving average (ARIMA) model with back propagation (BP) neural network [2], etc.

In this research, we were used NARX neural network to predict CPU, memory and queue time series of the web server load with three different training algorithms, levenberg-Marquardt (LM), bayesian regularization (BR) and scaled conjugate gradient (SCG). This was done with a view to see which algorithm produces better results and has faster training for the application under consideration.

2.3 Artificial Neural Network

Neural networks are nonlinear parallel structure, inspired by human brain system. Originally modeled from the biological central nervous system of human beings. ANN is a large-scale parallel distributed information processing system that is composed of many inter-connected nonlinear computational units, i.e., neurons [18]. These neurons have adaptive weights, tuned by a learning algorithm, which enables neural networks to approximate non-linear functions of their inputs. The adaptive weights describe the connection strengths between neurons, activated during training and prediction.

An ANN based approach yields some valuable features over traditional methods, such as adaptive learning, distributed association, non-linear mapping, as well as the ability to handle imprecise data [14]. The authors in [18] states, ANN can be comparable machine produced to function the same way the human brain performs a given task or function of interest. A good

advantage of that, it can make models easy to use and more accurate from complex natural systems with large inputs. The network maps the input vector into corresponding output vector and it is only imperative and other values need not be known. This makes ANNs very useful to mimic non-linear relationships without the need of any already existing models.

In recent times artificial neural networks (ANN) has become popular and helpful model for classification, clustering, pattern recognition and prediction in many disciplines. ANNs are one type of model for machine learning (ML) and has become relatively competitive to conventional regression and statistical models regarding usefulness. Taking advantage of the capability of modeling non-linear functions of the input, researchers has been using artificial neural networks to capture the hidden and complex characteristics lying within the time series itself, and then to predict the upcoming series [18].

Generally, ANN is structured as three layers: input layer, hidden layer(s), and output layer. To design a good ANN model, three steps must be performed. Firstly, the input data accompanied with the required output are presented to the network together. Then, the network is trained using the provided data to estimate the output based on some specific configuration. Finally, the developed model must be tested to estimate the output based on input data, which are not used in the training step.

2.4 Taxonomy of Artificial Neural Network Architecture

The authors in [18, 20] presents, artificial neural networks can be classified into two categories such as feedforward neural network, (FFNN) and recurrent or feed-backward neural network (FBNN). In FFNN, information is transmitted only in one direction, that is from the input nodes, to the hidden nodes, if any, and then to output nodes. Each unit in the FFNN layer relates to all the other units in the layers. These layers' connections with units are not all equal because each connection can have a different weight or strength. The weights of the network connections measure the potential amount of the knowledge of the network. The information processing in the network involves data entry from the input units and passes through the network, flowing from one layer to another layer until it gets to the output units. That means FFNN can logically handle task according to first come first serve bases of inputs. The architecture of FFNN is shown in Figure 2.1.

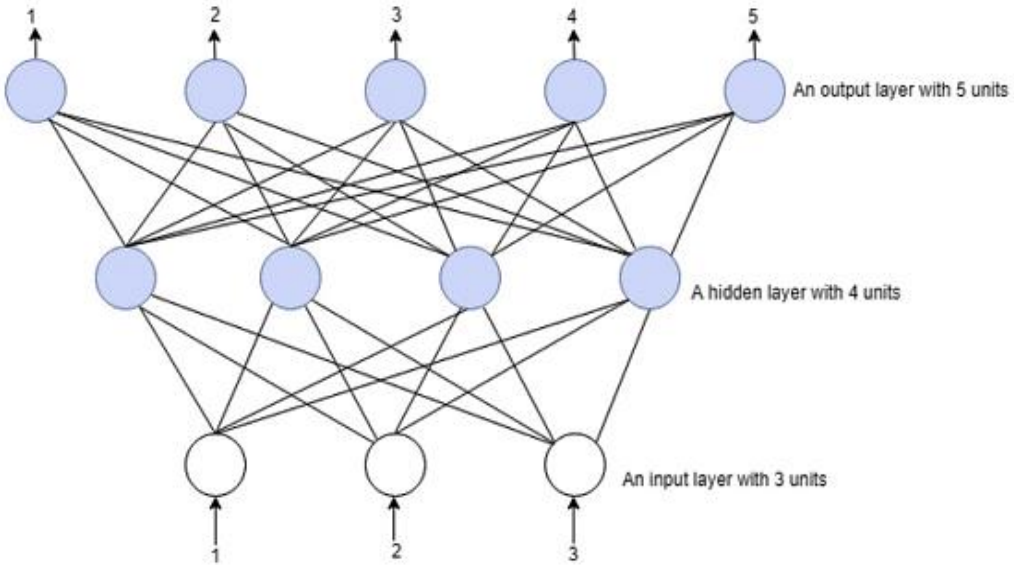


Figure 2. 1 : Feed-forward neural network [18].

On the other hand, the feed backward neural network uses internal state "memory" to store information and process sequence of data inputs. Figure 2.2 depicts the architecture of feed backward neural network. In FBNN the connections between nodes produced a coordinated graph in sequence that allows the feedback neural networks to demonstrate dynamic terrestrial behavior for a time sequence. The network extends over time, with edges that feed into the next time step rather than feeding into the next layer concurrent time of step.

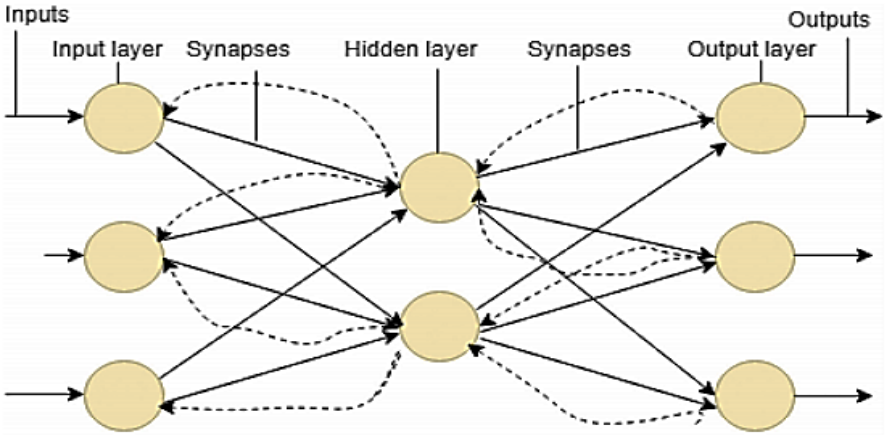


Figure 2. 2: Feed-backward neural network [18].

2.5 Nonlinear Autoregressive Network with Exogenous Inputs

Recurrent neural networks (RNNs) are capable of representing arbitrary nonlinear dynamical systems [21]. Among various types of the recurrent neural networks such as distributed time delay neural networks (TDNN), layer recurrent networks and nonlinear autoregressive network with exogenous inputs (NARX), the latest is of great interest in input output modeling of nonlinear dynamical systems and time series prediction [17, 22].

NARX is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modelling. Figure 2.3 illustrates the standard NARX network. The standard NARX network used here is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. This network also uses tapped delay lines (d) to store previous values of the input, x(t) and output, y(t) sequences. A mathematical description of the NARX model is summarized in equation (1) in which f is a nonlinear function.

$$y(t)=f\,(y\,(t-1),\ldots,y\,(t-d),\,x\,(t-1),\ldots,\,x(t-d)) \qquad (1)$$

Where, y(t) is the output of the NARX network and also feedback to the input of the network and tapped delay lines (d) that store the previous values of x(t) and y(t) sequences. These make the model dependent on predicted values and old values of the input [16, 17, 21]. The experimental results showed in [15], NARX networks are better in discovering the characteristics and behavior of long time series than conventional recurrent neural networks based on using the back propagation through time (BPTT) algorithm.

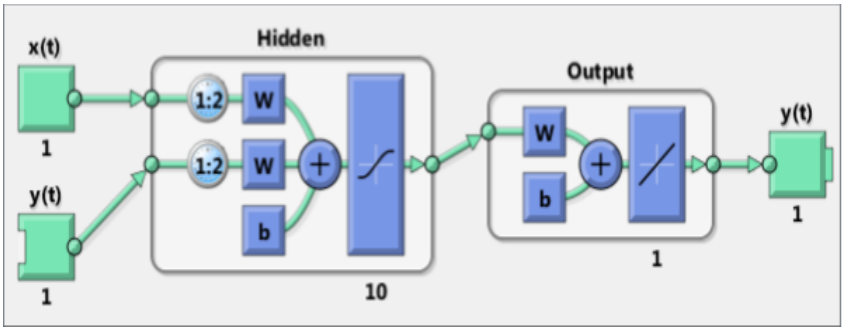
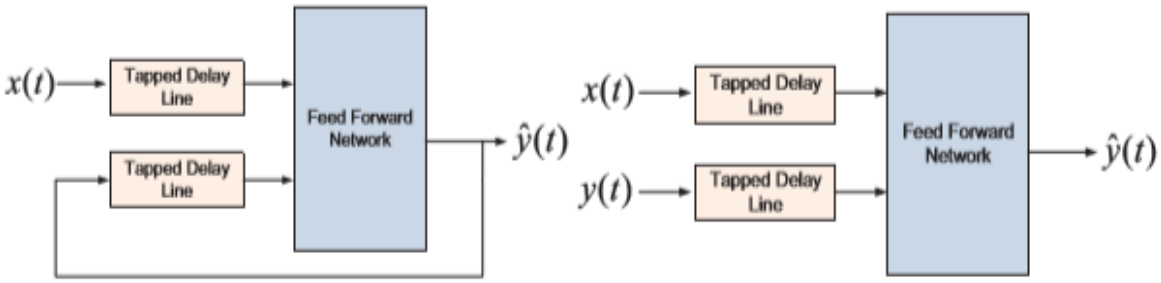


Figure 2. 3 : NARX network configuration

"A NARX neural network can be implemented in two setups namely parallel and series-parallel architectures" [22]. These are shown in Figure 2.4. In parallel architecture the output is feedback to the input of the feedforward neural network as part of the standard NARX architecture, while in series-parallel architecture the true output is used instead of feeding back the estimated output. NARX networks are computationally powerful in theory and have several advantages in practice. It also has been reported that gradient descent learning can be more effective in NARX networks than in other recurrent architecture [15, 16, 19]. In this research we have used both architecture in order to make future prediction values of the server load.



(A) Parallel (B) Series-Parallel

Figure 2. 4 : NARX (a): Parallel and (b): Series-Parallel Architectures [22]

2.6 Neural Network Training Algorithms

Training is the process of determining the optimal weights and bias points of the artificial neural networks. This is done by defining the total error function between the network's output and the desired target. Recently, there are different variations of training algorithms being innovated where each has certain advantages and disadvantages depending on the network architecture [15, 16].

Beale et al [16], argue that, it is very difficult to know which training algorithm will be the fastest for a given problem due to several factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and whether the network is being used for pattern recognition or function approximation.

The authors also compared different network architectures and training algorithms and concluded that, to provide the best forecasting accuracy, the various settings of the artificial neural network (the number of neurons in each layer, the number of hidden layers, the network's weights and biases) should be carried out with caution. Lowering the number too much could have the effect of reducing the neural network's computational power and restricting its capability of generalization, while using a higher number than needed might increase the complexity of the system. In this research we were employed three learning algorithms (LM, BR and SCG) and their performances pertaining to accuracy of multistep ahead web server load prediction are compared.

2.6.1 Levenberg-Marquardt Algorithm

Levenberg-Marquardt (LM) algorithm was designed to approach second-order training speed without having to compute the Hessian matrix [16]. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as equation (2).

$$H = J^T J \tag{2}$$

And the gradient can be computed as equation (3).

$$g = J^T e \tag{3}$$

Where 'J' is the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights and biases, and 'e' is a vector of network errors. The Jacobian matrix can be computed through a standard back-propagation technique that is much less complex than computing the Hessian matrix.

The LM algorithm uses this approximation to the Hessian matrix in the following Newton like update as shown in equation (4), where 'x' represents connection weights:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \tag{4}$$

When the scalar 'μ' is zero, this is just Newton's method, using the approximate Hessian matrix. When 'μ' is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, 'μ' is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would

increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

2.6.2 Bayesian Regularization Algorithm

Bayesian regularization (BR) is a network training algorithm that updates the weight and bias values according to LM optimization [23]. It minimizes a combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalizes well. BR introduces network weights into the training objective function which is denoted as $F(\omega)$ in equation (5).

$$F(\omega) = \alpha E_\omega + \beta E_D \quad (5)$$

Where 'E_ω' is the sum of the squared network weights and 'E_D' is the sum of network errors. Both 'α' and 'β' are the objective function parameters. In the BR framework, the weights of the network are viewed as random variables, and then the distribution of the network weights and training set are considered as Gaussian distribution.

2.6.3 Scaled Conjugate Gradient Algorithm

The algorithm was proposed by Leonard and Kramer to combine the conjugate gradient and line search strategy and get a much better convergence speed [23, 24]. The general training algorithms use the learning rate to decide the updating step of weight and thresholds, while most conjugate gradient algorithms have their step updating automatically. It has the function of the second-order method without calculating or storing the second derivative information.

In most of the conjugate gradient algorithms the step-size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step-size, which will minimize the performance function along that line.

All the conjugate gradient algorithms' search direction 'P_0' is started from the steepest descent direction 'g_0' as described in equation (6).

$$P_0 = -g_0 \quad (6)$$

And then decide the weight and threshold value (X) with the line search method as shown in equation (7).

$$x_{k+1} = x_k + \alpha_k P_k \quad (7)$$

where ‘P’ is the search direction and the parameter ‘α’ is to decrease the gradient in the search direction.

2.7 Locally Distributed Web-Server

Distributed computing systems are built over a large number of autonomous computer nodes. These computing nodes are uniquely identified in a network with their IP address and interconnected by SANs, LANs, or WANs in a hierarchical manner, and capable of collaborating on a task. Distributed computing platforms are designed to deliver parallel computing environment for various potential computing and non-computational problems. The potential of distributed computing systems are related to the management and allocation of computing resources relative to the computational load of the system [31].

Distributed computing systems can be deployed in multiple geographically dispersed data centers or locally distributed on multiple machines in the same data center. Cardellini et al [30] presents, a taxonomy of locally distributed web server system architectures, given a set of server nodes that host a web site at a single location. The authors [30] identify three main classes of architectures depending on the name virtualization being extended at the IP layer or not. With the current situation; in ethio telecom, there are services provided through a locally distributed web server systems for internal and external users. And these web servers are distributed on multiple machines in the same data center in a single location [33].

2.8 Chapter Summary

In this chapter, we presented an overview of the basic concepts of time series and application of artificial neural network in predicting of time series data’s. On the one hand, we discussed taxonomy of artificial neural network architecture and neural network training algorithms. In particular, we presented an overview on the nonlinear autoregressive network with exogenous inputs (NARX) neural network and the three training algorithms: levenberg-Marquardt (LM), bayesian regularization (BR) and scaled conjugate gradient (SCG), which are the common theme in this thesis. Finally, we described locally distributed web server system architectures to have a common understanding on the terminology.

3. Related Works

Researches have been working on resource utilization prediction using the advantage of neural network models. Load balancing, managing disk IO, scheduling processes, resource requirements, and performance research establishes a management framework that is enhanced by prediction [7]. The ability to predict resource utilization is a critical part of improving system availability, since overloading causes increased response time or dropped service [6]. As mentioned in Chapter 1, researchers have been developing different approaches to provide more accurate resource predictability to support these improvements.

Huang et al [27], presents NARX neural network based CPU Load prediction to define data mapping for dynamic resources. They used the predicted interval CPU load and variance of future resource capabilities to obtain the CPU load decision, which can be used to guide the scheduling decision. In their experiment, the authors have used an application resource utilization trace of CPU load for 20 minutes, total of 240 load time series data, to estimate average CPU load an application will experience during execution.

In a similar fashion, the authors in [22], studied a multi-step ahead response time predictor for database queries based on NARX NNs model. The experiment was done on the interaction of one application server with one database server. They used Apache Jmeter load generator to stress test. And they measured the response times of the queries sent to the database server from the application server, which later fed back to the input layer for training. The authors chosen bayesian regularization algorithm as a training algorithm, the tapped delay parameter of three delay and sum of squared errors (SSE) as performance metric. The maximum prediction error was less than 5%. From this, they conclude that the proposed predictor benefits for several promising characteristics which turns it into a viable candidate for being implemented in admission control products for computing systems.

Viswanath and Valliyammai [8] proposed a prediction approach that combined Adaptive Neuro based Fuzzy Inference Systems (ANFIS) and clustering process to predict the future CPU load based on the historical data in a grid environment. They used 10,000 series of CPU load data for their experiment, load trace collected by Dinda [34]. The historic CPU load data was first divided into sub-clusters (4-clusters of CPU time) using the fuzzy C-means clustering. Each sub-cluster was then fed to local ANFIS prediction models. The appropriate

ANFIS cluster is then used to predict the future CPU load value, one provides a very minimum error in result.

Naseera et al [9] explore four neural network learning techniques: Back Propagation (BP), Quick Propagation (QP), Back Propagation with Momentum (BPM) and Resilient Propagation (RP) algorithm for predicting future CPU load of the server. They designed a multilayer neural network and trained with learning algorithms for the input patterns collected from the grid environment, CPU load traces given by Dinda [34], to predict future CPU load statistics. The mean and standard deviation of the predicted values were computed and analyzed against the mean and standard deviation of actual values for all the algorithms. The results of RP algorithm were compared with among the other learning algorithms, it was exhibited better performance and least prediction error. BP was registered more prediction error.

In [4], the authors devised two set of experiments to predict CPU utilization of the server in future time, using different dynamic neural network prediction technique, given CPU, memory and total number of processes as input parameters. They collected 2,016 sample records of data from webmail server of Palestine Polytechnic University (PPU). The objective of the experiment was to perform one-step ahead prediction of the server load. The first experiment was conducted using time delay neural network (TDNN). In their experiment, they only used average CPU utilization time series of data to predict future CPU utilization of the server. And the best result was obtained with root mean squared error (RMSE) of 0.0257 at a maximum time delay parameter of 6. In the second experiment, they used another time series of CPU, memory and total number of processes to predict CPU usage of the server with nonlinear autoregressive neural network (NARX) model, and they have obtained the list prediction error of (RMSE) 0.0244 at maximum time delay parameter of 6. The results show that NARX performs better than TDNN.

Jina et al [2], also offer a composite model by combining the back propagation (BP) neural network with autoregressive integrated moving average (ARIMA) model to improve the predictive result of CPU utilization in virtual machine (VM) environment. First they preprocessed (normalize) the CPU utilization series data, then they fitted the CPU utilization by the ARIMA prediction technique, after that they calculated nonlinear residual part by the BP neural network, and finally they obtained the real predictive result by adding both parts. To do their experiments, they used 300 records of CPU utilization with the interval of 15

seconds to predict the CPU utilization in future 30 intervals. The result shows that ARIMA-BP model was more accurate than the single model. The mean relative error of ARIMA-BP was 0.0186 whereas ARIMA and BP separately was 0.0370 and 0.0265. The mean square error of ARIMA-BP was 0.0126, similarly ARIMA and BP was 0.303 and 0.0228. The mean absolute error was ARIMA-BP (0.097), ARIMA (0.208) and BP (0.125). Finally, they conclude that the composite approach can improve the predictive result, more close to the real value. And useful in strategy of VM resource dynamic deployment process.

Many studies have been explored the area of resource utilization prediction. However, Bianchi et al [17] discussed, there is no a specific neural network model that outperforms the others in every prediction problem. The choice of the most suitable architecture depends on the specific task at hand and it is important to consider more training strategies and configurations for each neural network. Moreover, the authors in [16, 24] states, for a given problem, it is very difficult to know which training algorithm will be the fastest, due to several factors including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and whether the network is being used for pattern recognition or function approximation. Thus, in this thesis, we aimed to explore predictability of the NARX neural network resource usage series of locally distributed web servers (CPU, memory, and queue time). It is pertinent to mention that (to the best of our knowledge) no author has considered this combination of load descriptors.

4. Proposed Methodology

In this chapter, we were proposed prediction approach that comprises four phases and is based on the non-linear autoregressive with exogenous inputs (NARX) model (Figure 4.1). The proposed multi-phase methodological approach is constructed in order to build a platform for the performance evaluation of three different neural network training algorithms (levenberg-Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG)).

The first phase of the proposed approach consists, acquiring and preprocessing of data. In this phase, we have collected resource usage series of data from the web servers. After that, we have formed the collected dataset in to two categories, which later be used for two Cases of experiments. During the second phase, we have implemented the NARX model. Then the datasets constructed during the first phase were used to train and test the three learning algorithms. In the third phase, we have obtained the best forecasting solution for both Cases of the experiments. Lastly, fourth phase, the algorithms were compared and best performing predictive learning algorithms were used to predict server loads. Details of each phases are discussed in the immediate sections below.

4.1 Acquiring and Preprocessing the data collected from the web server

In this work, we had been obtained 7 days load data from 5 servers of ethio telecom business support system, total of 10,080 samples. The samples load data includes average CPU, memory usage and queue time of the servers. Each of the servers' datasets comprises a number of 2,016 samples with a total of 6,048 records. We were used I2000 monitoring tool to collect these series of data. Afterwards, we have divided each of the server's datasets into two subsets and each subset's further divided into two categories called Case-1 and Case-2. The first subset contains 1728 samples and it will be used in the subsequent phases in order to train the forecasting NARX model, while the second subset comprises 288 samples will be used later in order to obtain a final validation of the forecasting solution, through a comparison between the predicted values and the real ones.

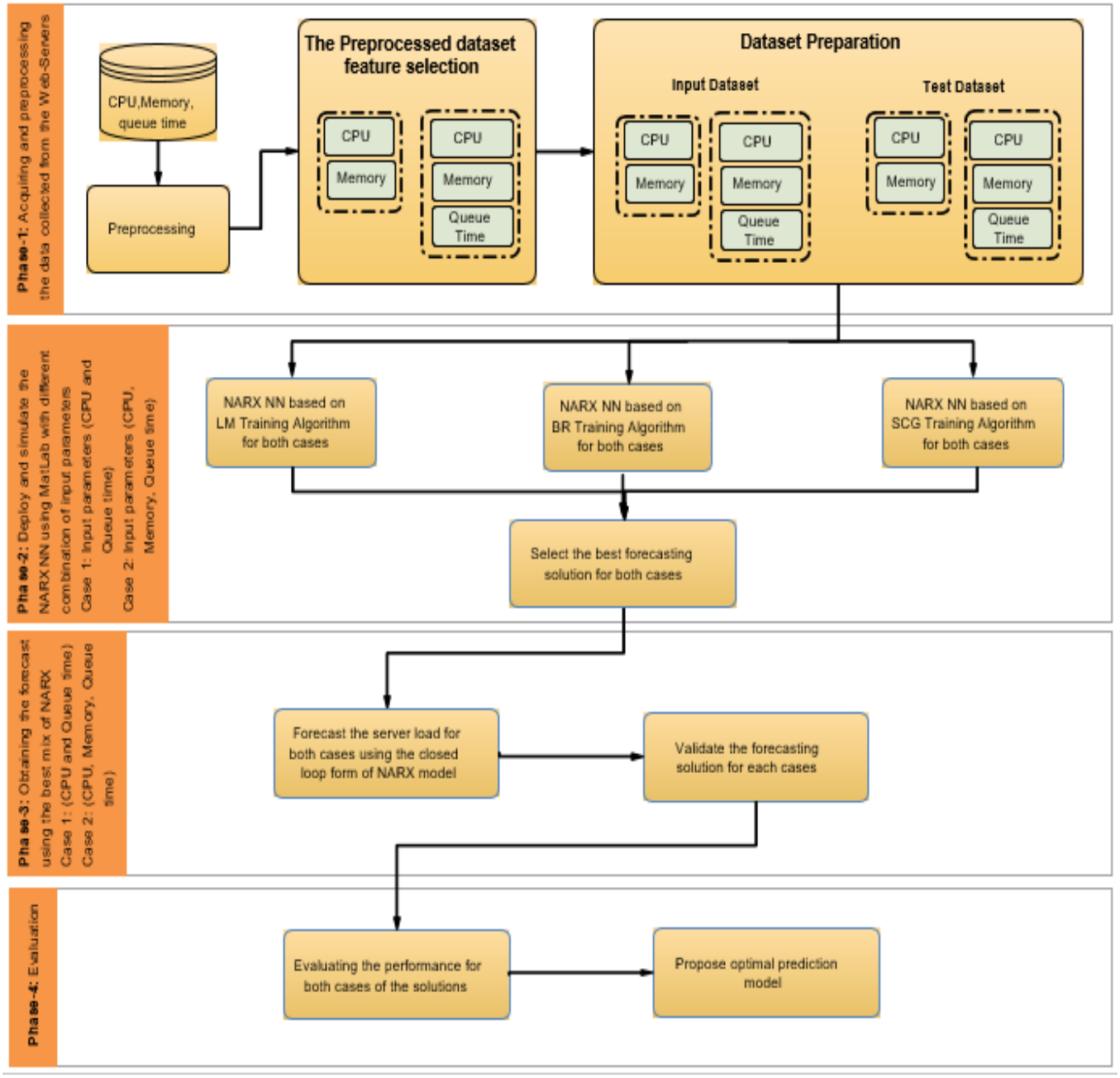


Figure 4. 1 : Proposed Methodological Approach

During this phase, the collected web server load data's (CPU, memory and queue time) are normalized to values between 0 and 1. Normalization was carried out by finding the highest value from each servers' within each input in the 2,016 sample datasets, and dividing all the values within the same feature by the maximum value. The main advantage for normalizing is to avoid attributes in greater numeric ranges dominating those in smaller numeric range [16, 29].

4.2 Developing the NARX Prediction Model

In the second phase, we have been employed the NARX neural network (NN) prediction solution covering two Cases, in the first Case using CPU and memory as input, and in the second Case CPU, memory and queue time. In both Cases, in order to obtain the best forecasting solution, we were examined various settings, regarding the training algorithms (LM, BR, SCG) and the delay parameter $d \in \{2, 3, 4, \ldots, 10\}$ for the NARX model. Matlab neural network time series analysis toolbox is utilized to build the network model and then the corresponding Matlab scripts are generated and further developed to serve the purpose.

As recalled from Chapter 2, we had been used the default configuration, 10 neurons in the hidden layer, to avoid the complexity of the network. Similarly, for each input dataset distribution, we were also used the default allocation, 70% of it for the training process, 15% for the validation one and the remaining 15% for the testing process. Each of these percentages being composed by randomly chosen samples. The training data is used by the network during training, and the network is adjusted according to its error, the validation data is used to measure network generalization, and to halt training at the point where it has generalized, test data have no effect on training and so provide an independent measure of network performance during and after training [16].

In order to identify the network that offered the best forecasting accuracy for each of the Cases and to all the selected delays, we have run 12 iterations, generating performance by mean squared error (MSE), regression (R), and response (validation) time to compare and identify for each of the Cases the best prediction accuracy out of the 12 ones. This was done in the open loop form of the NARX model. After that, we were compared for each of the two Cases by analyzing the performance metrics of all the NARX NNs, trained using the LM, BR and SCG algorithms, to obtain the best mix of networks in each of the Cases.

4.3 Obtaining the Forecast using the best mix of NARX NNs

In this phase, after having the best mix of NARX NNs for both Cases (Case-1 and Case-2), we were putted the model into the closed loop form, so as to perform a multi-step ahead prediction [16, 17]. We are forecasted using the closed loop form of the NARX model for the next 12-step ahead utilization of the server load (CPU, memory and queue time). In this purpose, for both of the Cases, we were used the same dataset presented in Section 4.1. Afterwards, we were validated the predicted results by comparing with the corresponding real

values of the consumption dataset for each of the Cases using root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) performance metrics. Finally, in the fourth phase, we were evaluated the prediction accuracy for both Cases of the solutions. In the following, we are obtained the optimal prediction model.

4.4 Performance Metrics

The evaluation of performance is essential with the purpose of finding the best neural network architecture, which gives the most reliable and accurate predictions. The following performance metrics were used for measuring and identifying the performance of the training algorithms and the network architecture that have provided the best prediction accuracy.

Response time (T): The time required for training the networks.

Correlation coefficient (R): It measures how strong a relationship is between two variables.

$$\text{Output} \sim = \propto \times \text{Target} + \beta \tag{8}$$

where "$\propto$" and " β" are constant value parameters related to the developed model which are obtained from the leaner approximation of the output (predicted) and the target (actual) values.

Mean squared error: Measure the average squared difference between outputs and targets.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(P_t - A_t)^2 \tag{9}$$

Where "A" and "P" are the actual and predicted values, "n" is the number of samples.

Mean absolute error: Measure the average magnitude of the errors in a set of prediction, without considering the direction.

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} | P_t - A_t| \tag{10}$$

Root mean squared error: It is a quadratic scoring rule that also measures the average magnitude of the error. It is square root of the average of squared differences between predicted and actual value.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(P_t - A_t)^2}{n}} \quad (11)$$

Mean absolute percentage error: It provides the error value as a percent of the true value compared to the prediction (expresses the percentage error).

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - P_t}{A_t}\right| X\ 100\% \quad (12)$$

4.5 Tool

MATLAB is a numerical computing environment and programming language, maintained by the MathWorks. The base MATLAB environment contains a lot of implemented functions, but for the more sophisticated projects, it offers additional toolboxes which contain special functions for the given problematic [16]. We used neural network time series analysis tool (ntstool) available in MATLAB (R2018a) to perform the prediction and analysis on the collected data. MATLAB offers four different levels at which the neural network toolbox can be used: Graphical User Interface (GUI), command-line, customization of the toolbox (requires advanced capability, allows us to create our own custom neural networks) and using the generated code files contained in the toolbox. In this thesis experiment, the generated code files and the GUI options are employed.

4.6 Chapter Summary

In this chapter, we are presented a four phases methodological approach based on the non-linear autoregressive with exogenous inputs (NARX) model. The proposed approach has been constructed in order to build a platform for the performance evaluation of three different neural network training algorithms (levenberg-Marquardt (LM), bayesian regularization (BR) and scaled conjugate gradient (SCG)). We had been collected 7 days web server load data (CPU, Memory, and Queue Time), to do our experiments. We used Matlab neural network time series analysis toolbox and the corresponding generated Matlab scripts to serve the purpose. Eventually, six performance evaluation criteria, which are: response time, correlation coefficient, mean squared error, mean absolute error, root mean squared error, and mean absolute percentage error, are defined and proposed as validation metrics.

5. Simulation and Evaluation

This chapter presents the findings of the various experimental simulations for determining the prediction capability of the NARX NNs. To realize this, we were evaluated and compared the selected training algorithms in their ability to build web server load prediction model, and then we were selected the most suitable training algorithm to predict especially for multi-step ahead utilization of the web server load.

Results and Discussion

Before evaluating the performance of the NARX NNs model, we are randomly selected one server sample dataset from the available 5 servers to do our experiments. The servers are homogeneous. Afterwards, in accordance with the proposed approach (see Chapter 4), we were employed the NARX NNs model for both of the Cases (Case-1 and Case-2).

To identify the best combination between the training algorithm (LM, BR or SCG) and the delay parameter (d), we were executed 12 iterations per each delay settings (see Appendix A and B). After numerous iterations, we were obtained the best performance records for each of the algorithms for both Cases of the tests.

We had been set priority (selection criteria) for the performance matrices to choose the best mix of performance records for the NNs (1^{st}, by response time T (< = 10sec), then by Correlation Coefficient (R) values, lastly by MSE values).

In the experiment, regarding the response time, we were noticed that for both of the Cases, the fastest networks were being the ones developed based on the LM and SCG training algorithms, while the slowest ones were being developed based on the BR training algorithm. Table 5.1 and Table 5.2 depicts, the experimental results of the NNs that we have obtained when developing NARX model.

Table 5. 1 : Experimental results for the Case-1 (CPU and memory as input parameter) when developing the NARX model.

Levenberg- Marquardt Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.001910	0.993250	0:00:01
3	0.001140	0.995520	0:00:01
4	0.000955	0.996080	0:00:01
5	0.000839	0.996200	0:00:01
6	0.000649	0.996650	0:00:01
7	0.000653	0.996480	0:00:01
8	0.000707	0.996440	0:00:01
9	0.000811	0.996460	0:00:01
10	0.000792	0.996380	0:00:01
Bayesian Regularization Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.00171	0.99348	0:00:03
3	0.00108	0.99594	0:00:04
4	0.000851	0.99645	0:00:05
5	0.000929	0.99669	0:00:08
6	0.000848	0.99674	0:00:08
7	0.000821	0.99678	0:00:10
8	0.000789	0.99694	0:00:18
9	0.000748	0.99681	0:00:15
10	0.000703	0.9969	0:00:20
Scaled Conjugate Gradient Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.00202	0.99244	0:00:01
3	0.0013	0.99518	0:00:01
4	0.00113	0.99578	0:00:01
5	0.00122	0.99582	0:00:01
6	0.00107	0.99583	0:00:01
7	0.00106	0.99586	0:00:01
8	0.00102	0.99595	0:00:01
9	0.00114	0.9958	0:00:01
10	0.00117	0.99579	0:00:01

Table 5. 2 : Experimental results for the Case-2 (CPU, memory and queue time as input parameter) when developing the NARX model.

Levenberg- Marquardt Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.00119	0.99561	0:00:01
3	0.000737	0.99696	0:00:01
4	0.00063	0.99641	0:00:01
5	0.000577	0.99716	0:00:01
6	0.000538	0.99739	0:00:01
7	0.000505	0.99612	0:00:01
8	0.000496	0.99541	0:00:02
9	0.000483	0.99639	0:00:02
10	0.000453	0.99669	0:00:03
Bayesian Regularization Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.00107	0.99549	0:00:05
3	0.000697	0.996	0:00:05
4	0.000592	0.99638	0:00:07
5	0.000562	0.9977	0:00:10
6	0.000553	0.998	0:00:17
7	0.000495	0.99295	0:00:30
8	0.000444	0.99782	0:00:45
9	0.000436	0.99641	0:01:25
10	0.000416	0.99754	0:01:06
Scaled Conjugate Gradient Training Algorithm			
Delay Parameter (d)	**MSE**	**R**	**T**
2	0.00147	0.99478	0:00:01
3	0.000952	0.996	0:00:01
4	0.000755	0.99625	0:00:01
5	0.000703	0.9966	0:00:01
6	0.000774	0.99656	0:00:01
7	0.000751	0.99645	0:00:01
8	0.000707	0.99637	0:00:01
9	0.000729	0.99637	0:00:01
10	0.000726	0.99663	0:00:01

5.1 Case-1 Simulation Result Discussion

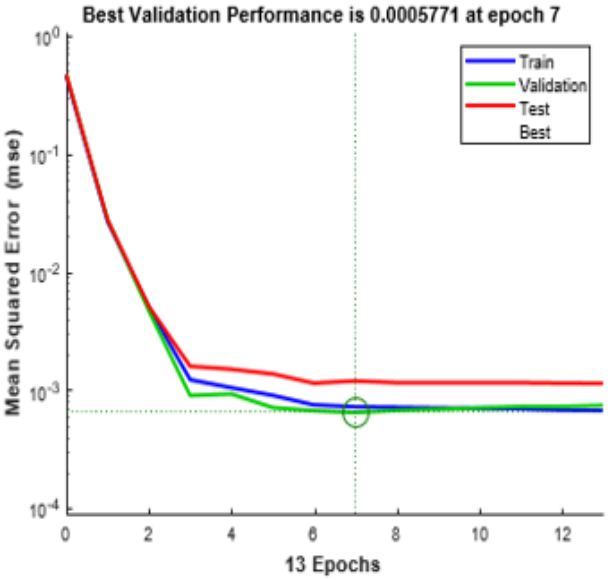
From Table 5.1, we were analyzed the performance of the LM training algorithm, the best prediction results were offered by d = 6 and d = 7. In those delay parameters, MSE (0.000649), R (0.99665), T (00:00:01) and MSE (0.000653), R (0.99648), T (00:00:01) had been obtained respectively. Contrasting to these results registered in this experiment (LM training algorithm), delay parameter of d = 2, has depicted the worst prediction accuracy highlighted by the highest value of MSE (0.00191) and the furthest R value from 1 (0.99325).

The result registered in the case of BR training algorithm (Table 5.1), the best prediction result is being obtained at delay parameter of d = 7, MSE (0.000821), R (0.99678) and T (00:00:10). From the tested 9 delay parameter settings, the slowest performing network out of the 9 employed delay settings based on the BR training algorithm, is the one developed using delay parameter of d = 10, T (00:00:20). The delay parameter d = 8 exhibited the closest value of R (0.99694) to the value of 1.

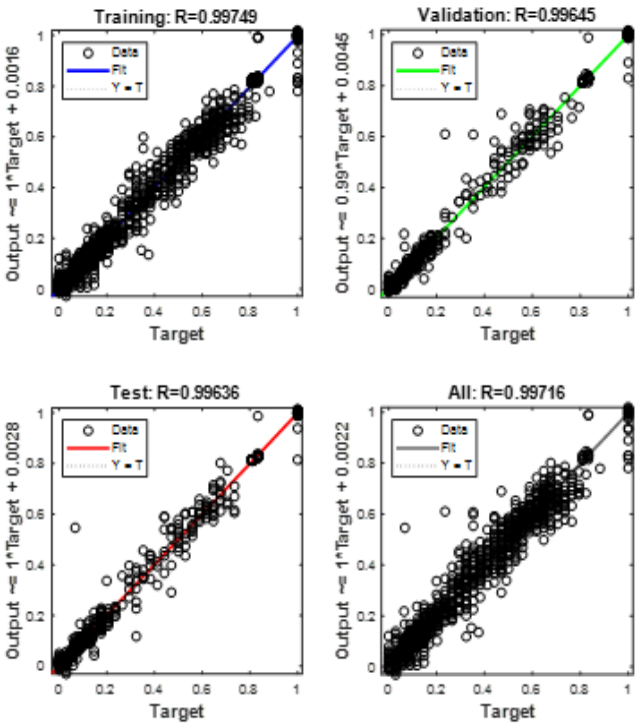
Evaluating the performance metrics registered by the networks employed based on the SCG training algorithm (Table 5.1), delay parameter of d = 8 has provided best values of MSE (0.00102) and R (0.99595), while the highest error value has been observed in the case of d = 2, MSE (0.00202) and the furthest value of R from 1 (0.99244).

5.2 Case-2 Simulation Result Discussion

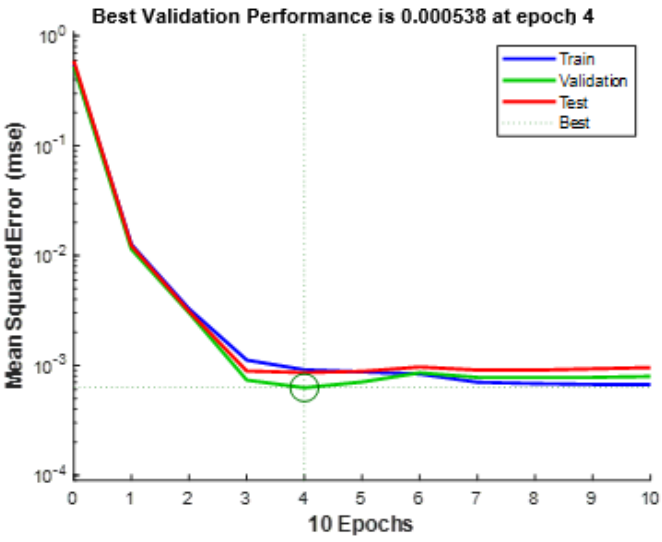
Subsequently, Table 5.2 remarked that, the best prediction results had been obtained when we were trained the NARX model using LM training algorithm were d = 5 and d = 6. In these NNs, we were obtained MSE (0.000577) and R (0.99716) at delay parameter d = 5, and MSE (0.000538) and R (0.99739) at d = 6. Figure 5.1 depicts, performance plots of the NARX model trained with LM training algorithm. The NARX model trained by the BR training algorithm, the best prediction result was found with error of (0.000562), R (0.9977), and T (00:00:10) at d = 5. The evaluation of the performance metrics reveal that the best prediction accuracy when we used SCG training algorithm, one uses the network that configured with delay parameter of d = 5, in this configuration we were obtained the lowest value of MSE (0.000703) while the correlation coefficient R has the closest value to 1 (0.9966).



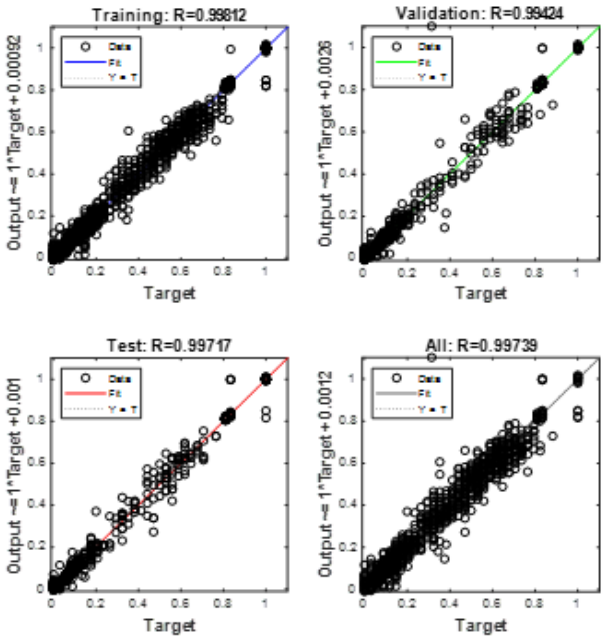
(A) Validation performance (when d = 5)



(B) Correlation between the network target and outputs (d=5)



(C) Validation performance (when d = 6)



(D) Correlation between the network targets and outputs (d=6)

Figure 5. 1 : Performance graphs of NARX neural networks trained with the LM learning algorithm for Case-2 solution

5.3 Performance of the Closed Loop Form of the NARX Model

According to the phases of our proposed methodology, after we were compared open loop form of the NARX NNs prediction accuracy in terms of registered performance metrics; these open loop form of the NARX NNs achieved the best mix for both Cases of the experiments (see Table 5.3) were placed into the closed loop form of the NARX NNs to predict 12-step ahead loads of the web server (see Figure 5.2).

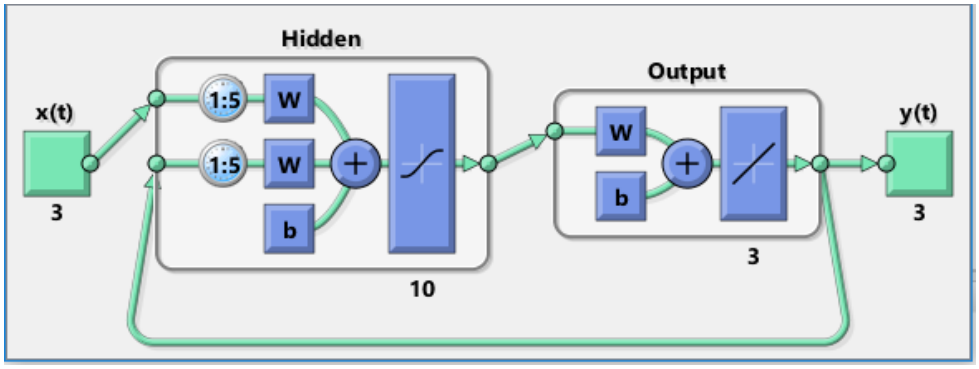


Figure 5. 2 : Closed loop form of NARX NN.

Table 5. 3 List of NARX neural networks registered best prediction accuracy, when using open loop form of the NARX NNs.

	Training Algorithms	**Delay Parameter (d)**
Case-1	Levenberg- Marquardt	d = 6
		d = 7
	Bayesian Regularization	d = 7
	Scaled Conjugate Gradient	d = 8
Case-2	Levenberg- Marquardt	d = 5
		d = 6
	Bayesian Regularization	d = 5
	Scaled Conjugate Gradient	d = 5

Therefore, in order to validate both Cases (Case-1 and Case-2) of the solutions and find out which NARX NNs have the best performance, we have been evaluated and compared the values of the differences between the real values of the web server resource consumption

(CPU, memory, queue time) with the predicted ones. For both Cases of the solutions, we used the same sample data from 288 sample datasets to predict the future consumption of the web server. Table 5.4 and 5.5 depicts, the accuracy of the three learning algorithms developed based on the NARX NNs, 12-step ahead prediction of web server loads.

Table 5. 4 : Performance comparison of the three learning algorithm, 12-step ahead web server load prediction for the Case-1 (CPU and Memory)

Case-1			
Training Algorithms		**Performance Metrix**	
		Average root mean squared error (RMSE)	Average mean absolute error (MAE)
LM	d = 6	0.0294	0.0186
	d = 7	0.0305	0.0188
BR	d = 7	0.0284	0.016
SCG	d = 8	0.0316	0.0194

Table 5. 4 demonstrates, that BR training algorithm with a delay parameter of d = 7 has recorded the smallest RMSE of 0.0284 and MAE value of 0.0160 compared to the other learning algorithms, calculated for the Case-1 experiments, while the second lowest value was noted using LM training algorithm delay parameter of d = 6, RMSE (0.0294) and MAE (0.0186). Moreover, the LM training algorithm with delay parameter of d = 7 has showed a moderate RMSE of 0.0305 and MAE value of 0.0188 compared to the SCG learning algorithms. Therefore, a conclusion may be drawn in favor of BR training algorithm with delay parameter of d = 7 as a superior prediction algorithm for a multi-step ahead CPU and memory utilization of the web server with the LM training algorithm following closely.

Figure 5. 3 shows the comparison between the actual and the prediction values of 12-step ahead average utilization of the web server loads forecasted based on the NARX model using BR training algorithm with delay parameter of d = 7. The numbers are presented in normalized form of the data.

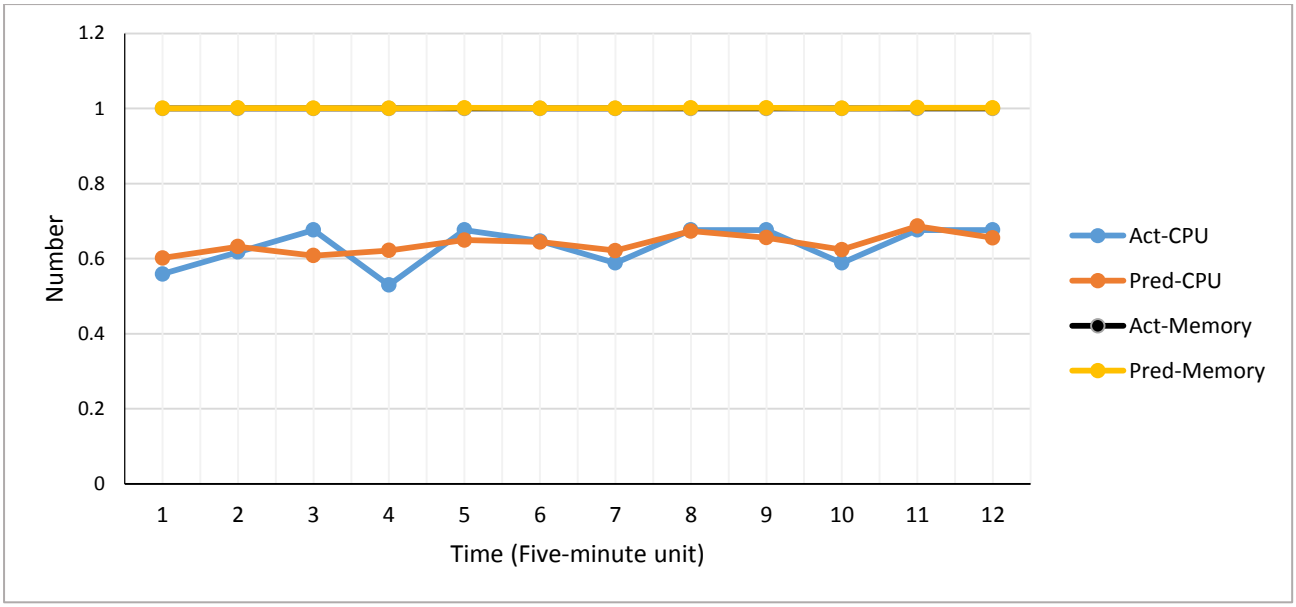


Figure 5. 3 : A comparison of actual values and predicted values generated by the BR training algorithm, delay parameter d = 7, when predicting 12-step ahead average CPU and memory utilization of the web server.

Table 5. 5 : Performance comparison of the three learning algorithm, 12-step ahead web server load prediction for the Case-2 (CPU, Memory and Queue Time)

Case-2			
Training Algorithms		**Performance Metrix**	
		Average root mean squared error (RMSE)	Average mean absolute error (MAE)
LM	d = 5	0.0239	0.0137
	d = 6	0.0246	0.0144
BR	d = 5	0.0263	0.0150
SCG	d = 5	0.0269	0.0177

The results in Table 5.5 showed that, for all values we have been obtained in this case of the experiment (Case-2), has a better accuracy than the ones we had obtained in Case-1 experiments for all cases of the solutions. Moreover, the result obtained by the LM training algorithm with delay parameter of d = 5 has registered the smallest error value of RMSE (0.0239) and MAE (0.0137) for both Cases (Case-1 and Case-2) of the proposed solutions.

Besides, even if, SCG training algorithm has presented the highest error values of RMSE (0.0269) and MAE (0.0177) in Case-2 (see Table 5.5) 12-step ahead web server load prediction, while we compared to the results we were obtained in Case-1 (see Table 5.4) 12-step ahead web server load prediction, it has registered a better accuracy then we were obtained in Case-1 12-step ahead prediction accuracy.

Figure 5. 4 exhibit, the comparison between the actual and the prediction values of 12-step ahead average CPU, memory and queue time consumption of the web server when predicting using the LM training algorithm.

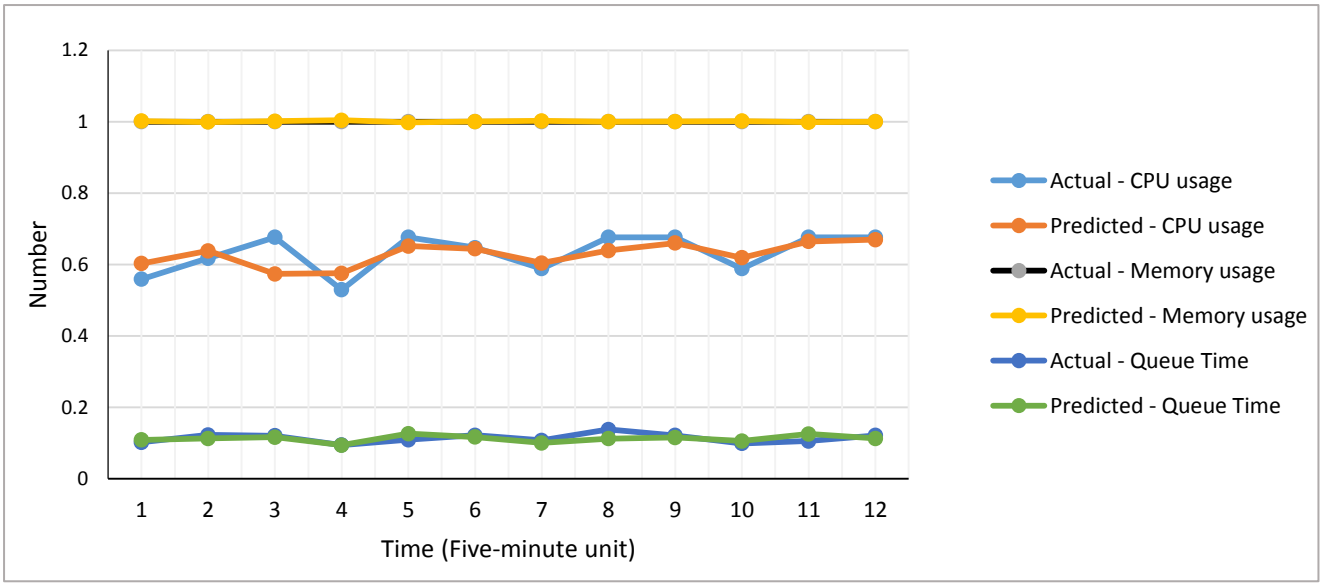


Figure 5. 4 : A comparison of actual values and predicted values generated by the LM training algorithm, delay parameter d = 5, when predicting 12-step ahead average CPU, memory and queue time utilization of the web server.

NARX NNs, delay parameter d = 5, trained with the LM training algorithm, Case-2 solution has shown the best prediction accuracy. Thus, we can conclude, that Case-2 proposed solution best for 12-step ahead web server load usage series prediction.

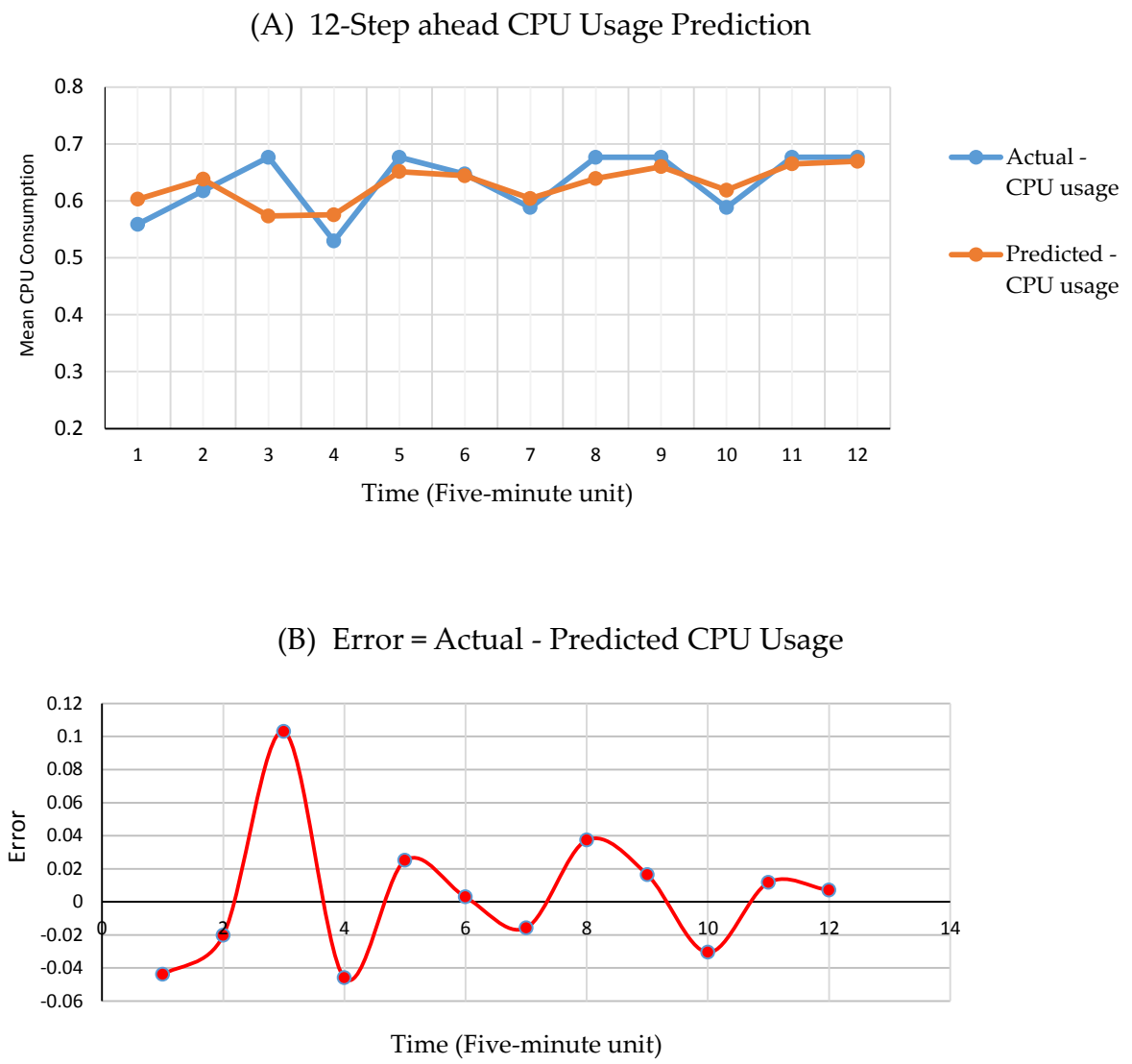


Figure 5. 5 : The differences between the actual CPU usage and predicted ones when predicting 12-step ahead consumption of the web server, using the LM training algorithm delay parameter d = 5 of the NARX model.

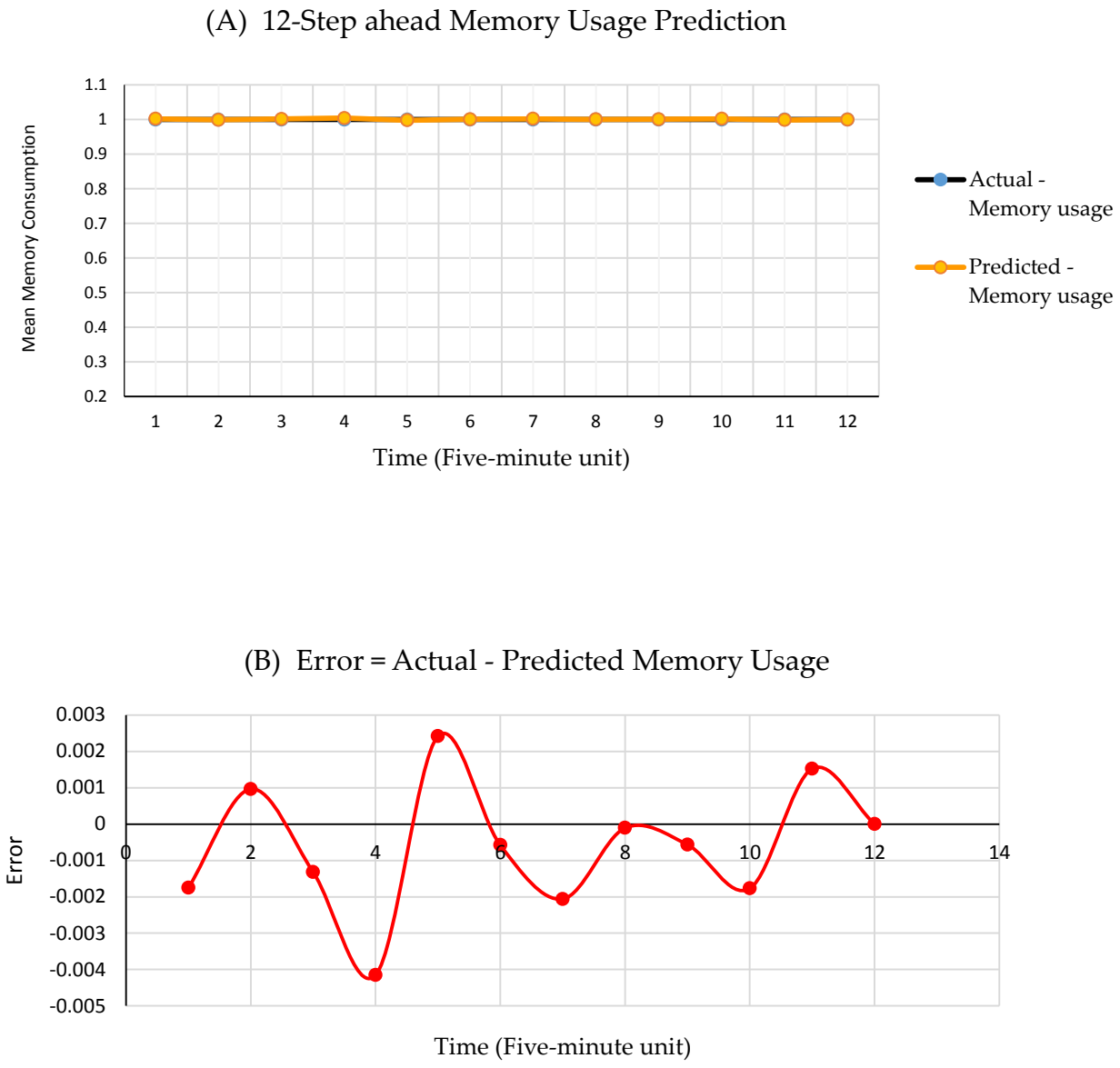


Figure 5. 6 : The differences between the actual Memory usage and predicted ones when predicting 12-step ahead consumption of the web server, using the LM training algorithm delay parameter d = 5 of the NARX model.

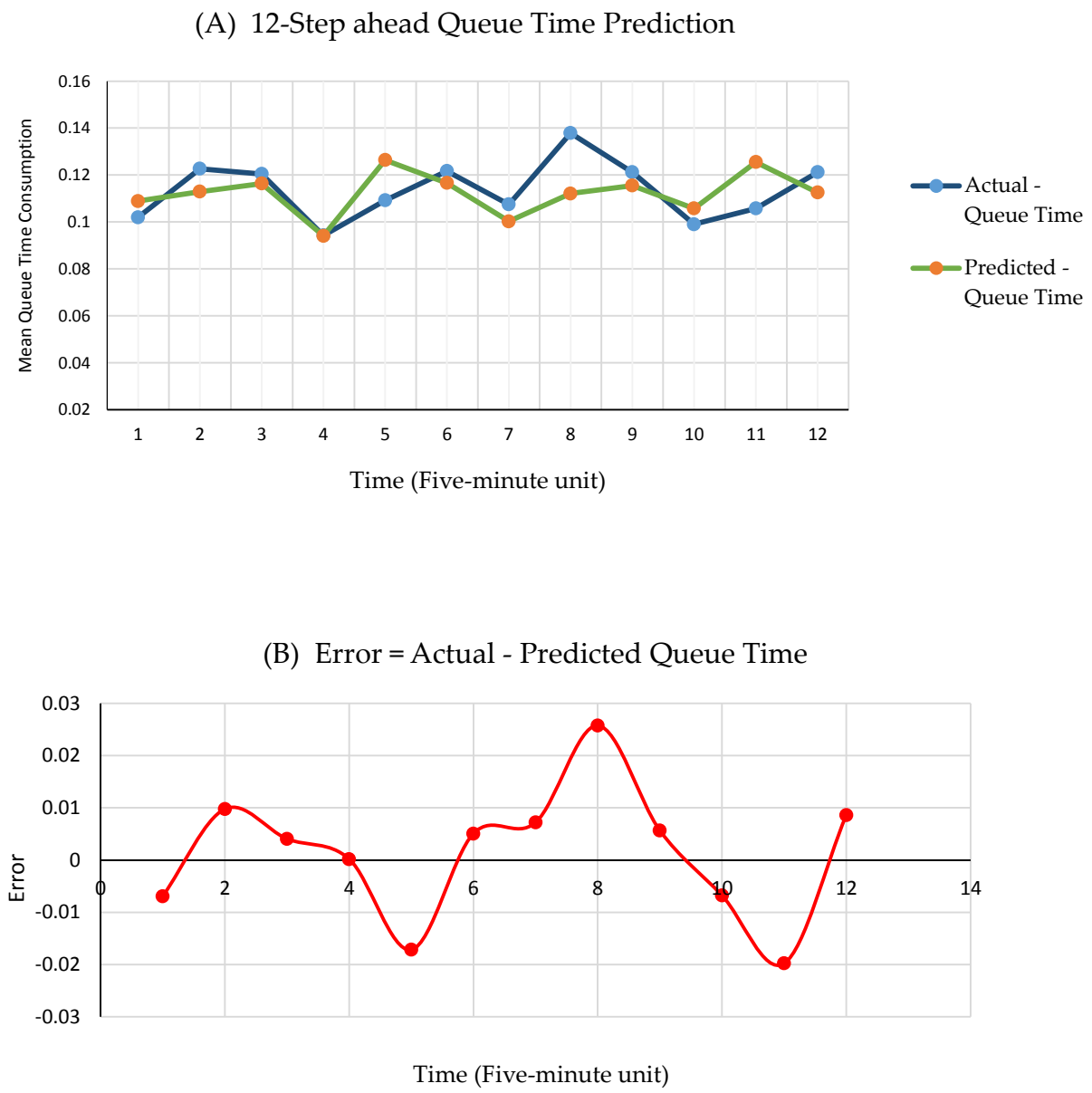


Figure 5. 7 : The differences between the actual Queue Time and predicted ones when predicting 12-step ahead consumption of the web server, using the LM training algorithm delay parameter d = 5 of the NARX model.

Figures 5. 5 – 5. 7 highlights the differences between the actual consumption of the web server and the predicted ones (12-step ahead web server loads). The forecasting has been done after having the best NNs NARX prediction solution (i.e. Case-2 LM training algorithm, delay parameter d = 5; NARX model).

Table 5. 6 : 12-step ahead average prediction results of Case-2 (CPU, Memory, Queue Time) NARX model.

Steps	LM Training Algorithm		BR Training Algorithm (d = 5)	SCG Training Algorithm (d = 5)
	d = 5	d = 6		
	MAPE	MAPE	MAPE	MAPE
1	4.941	6.590	4.796	6.354
2	4.363	6.344	4.906	4.993
3	4.991	6.849	5.357	5.952
4	4.519	8.170	6.479	6.676
5	4.926	7.488	6.189	6.649
6	4.365	6.868	5.510	5.780
7	4.199	6.281	4.965	5.569
8	4.683	6.216	5.540	5.331
9	4.427	5.775	5.037	5.098
10	4.392	5.597	5.145	5.708
11	4.617	5.326	4.993	5.901
12	4.459	5.002	4.649	5.610

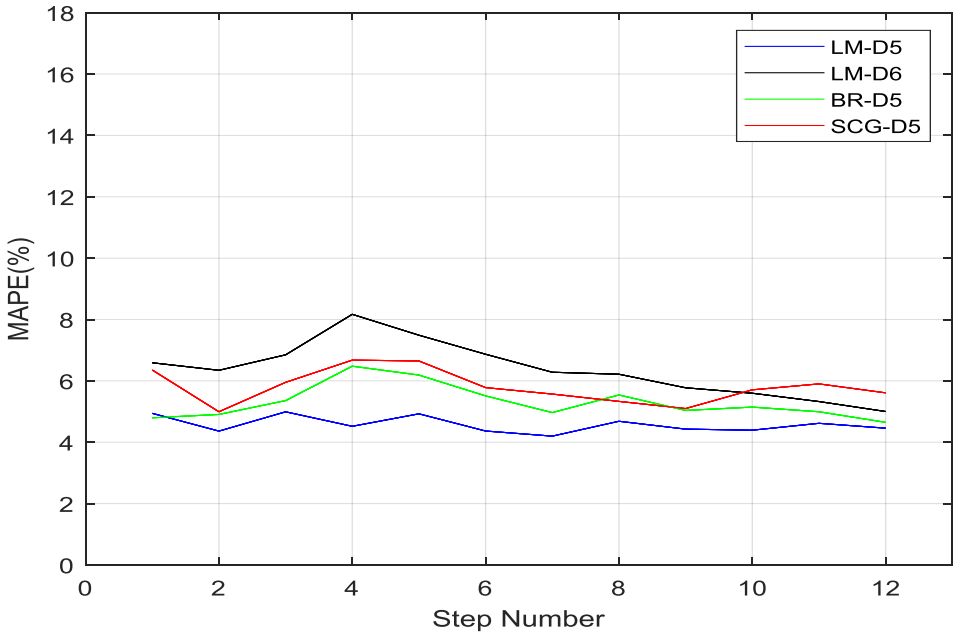


Figure 5. 8 : Performance evaluation of the three learning algorithms, Case-2 solution.

As it is seen from Table 5. 6 and Figure 5. 8, all algorithms start with low prediction accuracy. Up to step 7, BR (d =5) outperforms LM (d = 6) and SCG (d =5). In step 8, SCG (d = 5) has registered a better accuracy than the BR (d = 5) and LM (d = 6). Step number 10 is a stage of transition for LM (d = 6), the performance gets better whereas BR (d =5) falls behind it. We have been analyzed the trends of all algorithms, it exhibits decreasing performance but the best result was obtained by LM (d = 5). LM (d = 5) outperformed LM (d = 6), BR (d = 5) and SCG (d = 5) by MAPE of 0.543%, 0.19% and 1.151% accordingly in terms of overall prediction accuracy.

When analyzing our work with the work proposed by the authors in [4], in our approach using the NARX NNs, we had been tested 12-step ahead usage series of web server load prediction through evaluation of three different training algorithms (LM, BR, SCG), While in paper [4], the authors had used only the LM training algorithm in order to develop the time delay neural network (TDNN) and NARX models. The authors target was to perform one-step ahead prediction of the server load while in our devised method we are targeted not only one-step ahead but also multi-step prediction of the web server load. And we were obtained better accuracy than the one obtained by the authors in [4]. In Case-2 LM training algorithm (d = 5 and 6) one-step ahead prediction, we were obtained MSE of 0.000577 and 0.000538 respectively; and using the BR training algorithm (d = 5), we were obtained MSE of 0.000562. Whereas, Aljabari et al [4], obtained MSE of 0.00059536 (d = 6) using NARX NNs and MSE of 0.00066049 (d =6) using TDNN NNs.

As previously mentioned papers (see Chapter 3), Jina et al [2], analyzes an approach for developing multi-step ahead utilization of server load prediction solution. The authors target CPU utilization prediction in VM environment, and in this purpose, they were offered a composite model by combining the back propagation (BP) neural network with autoregressive integrated moving average (ARIMA) model, while in our case we had been used NARX model with three learning algorithms (LM, BR, SCG). The authors remarked that the best results with MAE value of 0.097 and MSE value of 0.0126. Regarding the time horizon of the prediction, the authors aim to obtained 30-step ahead CPU utilization of the server with interval of 15 seconds. In our approach, we had been tested 12-step ahead CPU, memory and queue time utilization of the web server with interval of 5 minutes. And obtained MAE value of 0.0137 and RMSE value of 0.0239 using LM leaning algorithm Case-2 proposed solution.

Server load prediction has gained an ever-increasing interest in the decades due to the multiple implications that it possesses. Being able to provide an accurate forecast of the server consumption creates the premises of achieving resource efficiency by minimizing the waste of resources and attaining balance between the application systems and the IT infrastructure.

5.4 Summary of Results

We are conducted a study that provide an accurate prediction of multi-step ahead usage series of web server load. To achieve this, we were employed a series of NARX NNs, based on the LM, BR and SCG training algorithms, to predict future resource utilization of the web server using as a time series record of data; CPU and memory as input parameter (Case-1) and CPU, memory and queue time as input parameter (Case-2).

In both Cases, we were tested various setting of delay parameter (d) for each of the three algorithms. Afterwards, we were selected networks that have registered better prediction accuracy by comparing their performance metrics, open loop form of the NARX NNs. we had been used response time (T), correlation coefficient (R), and mean squared error (MSE) values as a selection criterion.

We were observed, that Case-2 tests have registered better prediction accuracy. Furthermore, we compared the performance metrics of the three learning algorithms 12-step ahead resource usage predictability of the web server using closed loop form of the NARX NNs, the network developed based on the LM training algorithm has showed the best result with MAPE value of 4.459%. Thus, this prediction value can bring a significant advantage in managing proactively the resources of the web servers. Therefore, we are suggested the NARX model using the LM training algorithm; Case-2 solution, to build multi-step ahead web servers load prediction.

6. Conclusion

Service assurance for the telecom operator is a challenging task and is continuously being addressed by academics and industry. To address these concerns, one promising approach is realizing machine learning technique to predict future loads of servers in order to take early mitigation actions. The results of these methods will provide improvements in load balancing, job dispatching, job distribution, and overload prevention by permitting a system to better anticipate resource needs further into the future. There are several state-of-the-art methods, i.e., analytical, statistical and hybrid prediction models, developed for this purpose. Recent studies show that application of artificial neural networks (ANN) has been growing to a great extent.

In this thesis, the NARX NNs and three different training algorithms (LM, BR, and SCG) prediction accuracy were investigated. A multi-phase methodological approach has been constructed and applied in order to build a multi-step ahead web server load prediction model. To evaluate the performance of the proposed model, two Cases were employed using CPU and memory as input parameter (Case-1) and CPU, memory and queue time as input parameter (Case-2).

The results of the experiment show that for 12-step ahead web server load (CPU, memory, and queue time) prediction, LM learning algorithm delay parameter d = 5, Case-2 solution has shown the best prediction accuracy with MAPE of 4.459%, followed by BR (d = 5) and LM (d = 6) algorithms with MAPE of 4.649% and 5.002% respectively. The SCG (d = 5) learning algorithm was least performer with MAPE of 5.610%.

Thus, the proposed model offers very good prediction results for a time horizon of 12-steps ahead resource usage series of the web server. Accuracy in prediction is necessary since, these who intend to use the information to improve the system. Therefore, having such a model, the company will greatly enhance the process of working towards a reliable services and customer satisfaction.

6.1 Future Work

In this thesis, we have been investigated the NARX NNs multi-step ahead usage series prediction of web serve load (CPU, memory and queue time). And we have been obtained promising results. However, in order to further validate the prediction strength of the model,

as a future work, we plan to extend in other tiers of application workloads that are not web based; database, application layer or combination can be modeled.

References

[1] C. Witt, M. Bux, W. Gusew, and U. Leser, “Predictive Performance Modeling for Distributed Computing using Black-Box Monitoring and Machine Learning,” 2018.

[2] W. Jina, Y. Yongming, and G. Jun, “Research on the Prediction Model of CPU Utilization Based on ARIMA-BP Neural Network,” vol. 03009, 2016.

[3] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, “PRACTISE: Robust prediction of data center time series,” Proc. 11th Int. Conf. Netw. Serv. Manag. CNSM 2015, pp. 126–134, 2015.

[4] G. Aljabari and H. Tamimi, “Server Load Prediction Based on Dynamic Neural Networks,” no. February, 2015.

[5] S. Saha, “A Survey on Resource Management in Cloud Computing,” vol. 5, no. 3, pp. 3887–3889, 2014.

[6] S. Sharifian, S. A. Motamedi, and M. K. Akbari, “An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads,” J. Supercomput., vol. 53, no. 3, pp. 440–463, 2010.

[7] D. W. Yoas and G. Simco, “Resource Utilization Prediction : A Proposal for Information Technology Research,” 2012.

[8] C. Viswanath and C. Valliyammai, “CPU load prediction using ANFIS for grid computing,” IEEE-International Conf. Adv. Eng. Sci. Manag. ICAESM-2012, pp. 343–348, 2012.

[9] S. Naseera, G. K. Rajini, N. Amutha Prabha, and G. Abhishek, “A Comparative study on CPU load predictions in a computational grid using Artificial Neural Network Algorithms,” Indian J. Sci. Technol., vol. 8, no. 35, pp. 182–191, 2015.

[10] A. M. Alakeel, “A Guide to Dynamic Load Balancing in Distributed Computer Systems,” vol. 10, no. 6, pp. 153–160, 2010.

[11] P. K. Chandra and B. Sahoo, “Prediction Based Dynamic Load Balancing Techniques in Heterogeneous Clusters,” Int. Comput. Sci. Technol. Conf., no. May, pp. 189–192, 2008.

[12] O. Ostashchuk, “Time Series Data Prediction and Analysis,” Master’s Thesis, no. May, 2017.

[13] W. Yan, “Toward automatic time-series forecasting using neural networks,” IEEE Trans. Neural Networks Learn. Syst., vol. 23, no. 7, pp. 1028–1039, 2012.

[14] C. A. Mitrea, C. K. M. Lee, and Z. Wu, “A Comparison between Neural Networks and Traditional Forecasting Methods : A Case Study,” vol. 1, no. 2, pp. 19–24, 2009.

[15] E. Diaconescu, “The use of NARX neural networks to predict chaotic time series,” WSEAS Trans. Comput. Res., vol. 3, no. 3, pp. 182–191, 2008.

[16] M. H. Beale and M. T. Hagan, “Neural Network ToolboxTM User’s Guide, The MathWorks; Inc. 3 Apple Hill Drive Natick; MA 01760-2098,” p. 410, 2015.

[17] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, “An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting,” pp. 1–41, 2018.

[18] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” Heliyon, vol. 4, no. 11, p. e00938, 2018.

[19] E. Bradley, “Nonlinear time-series analysis revisited,” no. July, 2017.

[20] M. Kayri and O. Cokluk, “Data optimization with multilayer perceptron neural network and using new pattern in decision tree comparatively,” J. Comput. Sci., vol. 6, no. 5, pp. 606–612, 2010.

[21] H. Xie, H. Tang, and Y. H. Liao, “Time series prediction based on narx neural networks: An advanced approach,” Proc. 2009 Int. Conf. Mach. Learn. Cybern., vol. 3, no. July, pp. 1275–1279, 2009

[22] P. Amani, M. Kihl, and A. Robertsson, “NARX-based multi-step ahead response time prediction for database servers,” Int. Conf. Intell. Syst. Des. Appl. ISDA, pp. 813–818, 2011.

[23] X. Pan, B. Lee, and C. Zhang, “A comparison of neural network backpropagation algorithms for electricity load forecasting,” Proc. - 2013 IEEE Int. Work. Intell.

Energy Syst. IWIES 2013, pp. 22–27, 2013.

[24] Ö. Ki and E. Uncuo, “Comparison of three back-propagation training algorithms for two case studies,” vol. 12, no. October, pp. 434–442, 2005.

[25] J. Wang, Y. Ren, D. Zheng, and Q. Wu, “A Machine-Learning Based Load Prediction Approach for Distributed Service-Oriented Applications,” pp. 462–465, 2007.

[26] A. Matsunaga and J. A. B. Fortes, “On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications,” 2010, pp. 495–504.

[27] J. Huang, H. Jin, X. Xie, and Q. Zhang, “Using NARX neural network based load prediction to improve scheduling decision in grid environments,” Proc. - Third Int. Conf. Nat. Comput. ICNC 2007, vol. 5, no. Icnc, pp. 718–722, 2007.

[28] C. Fu and L. Zhang, “Adaptive Load Balancing Strategy Based on LVS 2 Server Load Balance and LVS 3 Traffic State Prediction with RBF and,” vol. 03028, pp. 1–5, 2017.

[29] Chih-Wei, H. et al, “A practical guide to support vector classification”. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003.

[30] V. Cardellini, E. Casalicchio, M. Colajanni, P. S. Yu, and I. B. M. T. J. Watson, “The State of the Art in Locally Distributed Web-Server Systems,” vol. 34, no. 2, pp. 263–311, 2002.

[31] B. Sahoo, “Dynamic Load Balancing Strategies in Heterogeneous Distributed System,” 2013.

[32] Huawei. Ethio telecom,“HUAWEI mVAS I2000 Function Requirement Specification For ethio-telecom Project”, 2013.

[33] Huawei. Ethio telecom,“ethio-NGBSS Low Level Design” ethio-telecom Project, 2015.

[34] Load Trace Archive. Available from: http://www.cs.cmu.edu/~pdinda/LoadTraces/. [CrossRef].

Appendices

A. Experimental results for Case-1

Developing the NARX model to predict CPU and Memory when delay parameter d ∈ {2, 3, 4, … ,10} for 12 iterations.

CASE-1													
Levenberg-Marquardt Training Algorithm													
Delay Parameter (d)	Performance Metrix	Iteration Number											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.001780	0.002060	0.001940	0.001770	0.001890	0.001910	0.001890	0.001970	0.001830	0.001860	0.001800	0.001880
	R	0.992670	0.992790	0.992670	0.992700	0.992680	0.993250	0.973110	0.992670	0.992820	0.992780	0.992880	0.992710
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
3	MSE	0.001070	0.001140	0.001240	0.001200	0.001140	0.001170	0.001170	0.001200	0.001200	0.001170	0.001160	0.001220
	R	0.995770	0.995320	0.995470	0.995470	0.995520	0.995380	0.995340	0.995280	0.995230	0.993500	0.995470	0.995380
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
4	MSE	0.000943	0.000977	0.001060	0.001010	0.000955	0.001050	0.001010	0.001020	0.001120	0.001010	0.001080	0.001040
	R	0.995660	0.995780	0.996020	0.996000	0.996080	0.995930	0.996050	0.995950	0.995960	0.996010	0.995940	0.996020
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
5	MSE	0.000953	0.000974	0.000103	0.000839	0.000858	0.001080	0.000890	0.000896	0.001070	0.000839	0.000884	0.000966
	R	0.996150	0.995900	0.995980	0.996200	0.996140	0.996130	0.995950	0.996110	0.996070	0.996140	0.995780	0.996100
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
6	MSE	0.000961	0.000889	0.000825	0.000835	0.000817	0.000793	0.000649	0.000645	0.000774	0.000724	0.000717	0.000866
	R	0.996270	0.962500	0.996340	0.995830	0.996090	0.996500	0.996650	0.996180	0.996230	0.996180	0.996320	0.996170
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
7	MSE	0.000722	0.000797	0.000781	0.000771	0.000653	0.000671	0.000729	0.000745	0.000717	0.000793	0.000853	0.000753
	R	0.996100	0.996060	0.996380	0.996380	0.996480	0.996270	0.996410	0.996430	0.996260	0.994100	0.996210	0.996290
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

8	MSE	0.000817	0.000843	0.000702	0.000638	0.000707	0.000804	0.000805	0.000731	0.000889	0.000857	0.000775	0.000804
	R	0.996050	0.995400	0.996020	0.996060	0.996440	0.996170	0.996310	0.996380	0.996250	0.996210	0.996360	0.996090
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
9	MSE	0.000846	0.000776	0.000811	0.000616	0.000749	0.000818	0.000872	0.000713	0.000786	0.000844	0.000855	0.000883
	R	0.996460	0.996260	0.996460	0.996070	0.996330	0.996430	0.996250	0.996420	0.996290	0.996450	0.996420	0.996320
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
10	MSE	0.000856	0.000884	0.000792	0.000848	0.000729	0.000831	0.000855	0.000722	0.000692	0.000785	0.000867	0.000821
	R	0.996130	0.996210	0.996380	0.996140	0.996260	0.996360	0.996130	0.996110	0.996160	0.996180	0.996380	0.996170
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

CASE-1

Bayesian Regularization Training Algorithm

Delay Parameter (d)	Performance Metrix	Iteration											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.00176	0.00167	0.00207	0.00197	0.00169	0.00179	0.00177	0.00181	0.00171	0.00171	0.00173	0.0017
	R	0.99334	0.99344	0.99267	0.99273	0.9933	0.99341	0.99341	0.99345	0.99345	0.99348	0.99341	0.99339
	T	0:00:03	0:00:04	0:00:03	0:00:04	0:00:03	0:00:04	0:00:03	0:00:03	0:00:04	0:00:03	0:00:04	0:00:03
3	MSE	0.00108	0.00108	0.00108	0.00104	0.00114	0.00107	0.00114	0.0011	0.00108	0.00106	0.0011	0.0011
	R	0.99535	0.99581	0.99594	0.99585	0.99585	0.99574	0.99581	0.99583	0.99591	0.9959	0.9957	0.99593
	T	0:00:06	0:00:06	0:00:04	0:00:04	0:00:05	0:00:03	0:00:05	0:00:03	0:00:05	0:00:05	0:00:06	0:00:05
4	MSE	0.000927	0.000989	0.000951	0.000918	0.000933	0.000969	0.000974	0.000958	0.000989	0.000851	0.000913	0.000916
	R	0.99641	0.99622	0.99575	0.99596	0.99629	0.99639	0.99612	0.99638	0.99635	0.99645	0.99634	0.9962
	T	0:00:05	0:00:04	0:00:06	0:00:04	0:00:04	0:00:04	0:00:04	0:00:04	0:00:06	0:00:05	0:00:05	0:00:04
5	MSE	0.00087	0.000908	0.000899	0.000878	0.000898	0.000929	0.000879	0.000891	0.000892	0.000938	0.000898	0.000912
	R	0.99631	0.99668	0.99654	0.99576	0.99666	0.99669	0.99646	0.99637	0.99666	0.9966	0.99652	0.99657
	T	0:00:04	0:00:06	0:00:05	0:00:06	0:00:06	0:00:08	0:00:04	0:00:07	0:00:06	0:00:08	0:00:05	0:00:07

6	MSE	0.000773	0.000822	0.000848	0.000854	0.00085	0.00084	0.000823	0.000864	0.000869	0.000846	0.000803	0.000829
	R	0.99679	0.99682	0.99674	0.99674	0.99681	0.99653	0.99667	0.99659	0.99672	0.99667	0.99658	0.99651
	T	0:00:15	0:00:10	0:00:08	0:00:06	0:00:06	0:00:10	0:00:14	0:00:18	0:00:08	0:00:12	0:00:14	0:00:18
7	MSE	0.000813	0.000821	0.000736	0.000809	0.000787	0.000775	0.000818	0.000762	0.000802	0.000768	0.000809	0.000811
	R	0.99669	0.99678	0.99675	0.99668	0.9969	0.99685	0.99688	0.99682	0.99664	0.99662	0.99672	0.99662
	T	0:00:15	0:00:10	0:00:08	0:00:09	0:00:12	0:00:14	0:00:08	0:00:14	0:00:15	0:00:12	0:00:08	0:00:14
8	MSE	0.000795	0.000784	0.000789	0.000737	0.000752	0.000769	0.000733	0.000801	0.000776	0.000784	0.000753	0.000772
	R	0.99685	0.99688	0.99694	0.9969	0.99687	0.99661	0.99692	0.99691	0.99692	0.99685	0.99693	0.99691
	T	0:00:16	0:00:18	0:00:18	0:00:22	0:00:17	0:00:16	0:00:16	0:00:23	0:00:18	0:00:26	0:00:16	0:00:18
9	MSE	0.000776	0.000724	0.000741	0.0008	0.000782	0.000748	0.000725	0.000763	0.000751	0.000754	0.000774	0.000794
	R	0.99689	0.99686	0.99657	0.9969	0.99689	0.99681	0.99684	0.99676	0.99677	0.99699	0.99703	0.99677
	T	0:00:21	0:00:22	0:00:19	0:00:17	0:00:18	0:00:15	0:00:30	0:00:16	0:00:27	0:00:21	0:00:20	0:00:21
10	MSE	0.000692	0.000716	0.000719	0.000736	0.000738	0.000703	0.000678	0.000718	0.000725	0.000674	0.000714	0.000737
	R	0.9967	0.99706	0.99695	0.99679	0.99672	0.9969	0.99694	0.99693	0.99698	0.99662	0.9969	0.99691
	T	0:00:23	0:00:30	0:00:38	0:00:35	0:00:27	0:00:20	0:00:40	0:00:37	0:00:38	0:00:24	0:00:28	0:00:21

CASE-1													
Scaled Conjugate Gradient Training Algorithm													
Delay Parameter (d)	**Performance Metrix**	**Iteration**											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.00216	0.00195	0.00195	0.00216	0.00213	0.00202	0.00211	0.0021	0.00232	0.00207	0.00229	0.00204
	R	0.99219	0.99262	0.99235	0.99228	0.99228	0.99244	0.99228	0.99165	0.9911	0.99189	0.99172	0.99242
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
3	MSE	0.00129	0.0013	0.00126	0.00122	0.0013	0.00129	0.00142	0.00142	0.0014	0.00143	0.00139	0.00142
	R	0.995	0.99516	0.99516	0.99495	0.99518	0.99501	0.99436	0.99471	0.99454	0.99441	0.99451	0.9945
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

4	MSE	0.00126	0.00112	0.00113	0.00127	0.001135	0.00132	0.00129	0.000117	0.00112	0.00113	0.00127	0.00116
	R	0.99532	0.99586	0.99571	0.99509	0.99528	0.9949	0.99481	0.99576	0.99537	0.99578	0.99522	0.99545
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
5	MSE	0.00117	0.00116	0.00153	0.0012	0.000118	0.00012	0.000125	0.00122	0.00138	0.00121	0.00109	0.00112
	R	0.99577	0.99534	0.99438	0.99578	0.99563	0.99575	0.99527	0.99582	0.99507	0.99558	0.9955	0.99548
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
6	MSE	0.00129	0.0012	0.00126	0.00112	0.00133	0.00124	0.0014	0.00131	0.00123	0.00127	0.00128	0.00107
	R	0.995	0.99533	0.99556	0.99569	0.99456	0.99539	0.99487	0.99543	0.99535	0.9947	0.99516	0.9958
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
7	MSE	0.00118	0.00116	0.00106	0.0011	0.00111	0.00116	0.00114	0.0012	0.00128	0.00115	0.0011	0.00113
	R	0.99521	0.99586	0.99586	0.99565	0.99555	0.99544	0.99545	0.99544	0.99528	0.99542	0.9958	0.9956
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
8	MSE	0.00105	0.00113	0.001132	0.00123	0.000116	0.00012	0.00102	0.00115	0.00113	0.00132	0.00119	0.0013
	R	0.99554	0.99535	0.99464	0.99531	0.99572	0.99541	0.99595	0.99562	0.9958	0.99525	0.9955	0.99543
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
9	MSE	0.00123	0.00119	0.00107	0.00133	0.00107	0.00127	0.00145	0.00104	0.000153	0.00114	0.00127	0.00117
	R	0.99508	0.99488	0.99564	0.99478	0.99568	0.99539	0.99449	0.99564	0.994	0.9958	0.99551	0.99557
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
10	MSE	0.00113	0.00133	0.00117	0.00133	0.00113	0.00126	0.00154	0.00164	0.0013	0.00102	0.00129	0.0012
	R	0.99576	0.99465	0.99579	0.99478	0.99557	0.99535	0.99447	0.99374	0.99492	0.99568	0.99489	0.99539
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

B. Experimental results for Case-2

Developing the NARX model to predict CPU, Memory and Queue Time when delay parameter d ∈ {2, 3, 4, … ,10} for 12 iterations.

CASE-2													
Levenberg- Marquardt Training Algorithm													
Delay Parameter (d)	**Performance Metrix**	**Iteration**											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.00139	0.00129	0.00135	0.00133	0.00131	0.00124	0.00119	0.00125	0.00130	0.00136	0.00134	0.00137
	R	0.9948	0.99536	0.99482	0.99489	0.9952	0.99462	0.99561	0.99483	0.99551	0.99523	0.99465	0.99496
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
3	MSE	0.000849	0.000857	0.000803	0.000791	0.000770	0.000758	0.000739	0.000737	0.000800	0.000768	0.000831	0.000858
	R	0.994642	0.99665	0.99631	0.99672	0.99547	0.99625	0.99671	0.99696	0.99685	0.99658	0.99678	0.99667
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
4	MSE	0.000726	0.000704	0.000685	0.000658	0.000648	0.000643	0.000630	0.000651	0.000655	0.000675	0.000689	0.000802
	R	0.9963	0.99414	0.99624	0.99632	0.99622	0.99614	0.99641	0.99637	0.99638	0.99615	0.99344	0.99618
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
5	MSE	0.000619	0.000656	0.000745	0.000612	0.000729	0.000602	0.000604	0.000635	0.000588	0.000577	0.000673	0.000659
	R	0.9971	0.99674	0.99678	0.99688	0.99485	0.99618	0.99632	0.99679	0.99651	0.99716	0.99636	0.99537
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
6	MSE	0.000571	0.000691	0.000562	0.000538	0.000515	0.000516	0.000649	0.000565	0.000638	0.000630	0.000668	0.000688
	R	0.9969	0.99664	0.99728	0.99739	0.99574	0.9971	0.99566	0.99691	0.99285	0.99664	0.99734	0.99692
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:02	0:00:01	0:00:03	0:00:01	0:00:01	0:00:02	0:00:01	0:00:04
7	MSE	0.000634	0.000601	0.000566	0.000526	0.000532	0.000505	0.000684	0.000591	0.000520	0.000512	0.000513	0.000552
	R	0.99469	0.99602	0.99609	0.99597	0.99603	0.99612	0.99611	0.99608	0.99575	0.99606	0.99316	0.99602
	T	0:00:02	0:00:01	0:00:04	0:00:01	0:00:01	0:00:01	0:00:01	0:00:02	0:00:02	0:00:01	0:00:01	0:00:02

8	MSE	0.000658	0.000506	0.000508	0.000496	0.000515	0.000617	0.000608	0.000603	0.000676	0.000627	0.000567	0.000565
	R	0.99467	0.99514	0.99498	0.99541	0.99536	0.99493	0.994662	0.99499	0.99519	0.99515	0.99539	0.99526
	T	0:00:02	0:00:02	0:00:03	0:00:02	0:00:01	0:00:01	0:00:01	0:00:01	0:00:02	0:00:01	0:00:01	0:00:02
9	MSE	0.000507	0.000609	0.000542	0.000558	0.000517	0.000538	0.000521	0.000512	0.000483	0.000540	0.000538	0.000557
	R	0.99627	0.99616	0.99637	0.99635	0.99630	0.99605	0.99632	0.99631	0.99639	0.99624	0.99615	0.99636
	T	0:00:02	0:00:01	0:00:01	0:00:01	0:00:02	0:00:02	0:00:01	0:00:02	0:00:02	0:00:02	0:00:02	0:00:02
10	MSE	0.000484	0.000661	0.000586	0.000494	0.000519	0.000453	0.000525	0.000619	0.000516	0.000539	0.000620	0.000578
	R	0.99638	0.99622	0.99629	0.99669	0.99665	0.99669	0.9965	0.99658	0.99601	0.99648	0.99607	0.99613
	T	0:00:02	0:00:02	0:00:02	0:00:03	0:00:03	0:00:03	0:00:02	0:00:02	0:00:03	0:00:02	0:00:02	0:00:03

CASE-2

Bayesian Regularization Training Algorithm

Delay Parameter (d)	Performance Metrix	Iteration											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.00117	0.00118	0.00109	0.00113	0.00107	0.00109	0.00108	0.00107	0.00114	0.00113	0.00110	0.00113
	R	0.99544	0.99446	0.99397	0.99509	0.99549	0.99533	0.99547	0.99489	0.99548	0.99518	0.9935	0.99526
	T	0:00:15	0:00:07	0:00:03	0:00:04	0:00:05	0:00:04	0:00:06	0:00:06	0:00:07	0:00:08	0:00:05	0:00:07
3	MSE	0.000712	0.000703	0.000704	0.000697	0.000699	0.000698	0.000706	0.00072	0.000702	0.000668	0.0007	0.000726
	R	0.99537	0.99539	0.99581	0.996	0.99532	0.99539	0.99549	0.996	0.99542	0.992	0.99586	0.99534
	T	0:00:07	0:00:06	0:00:06	0:00:05	0:00:05	0:00:21	0:00:09	0:00:07	0:00:10	0:00:08	0:00:06	0:00:06
4	MSE	0.000605	0.000562	0.000592	0.000608	0.000593	0.000582	0.000632	0.000595	0.000565	0.000614	0.000591	0.000598
	R	0.9963	0.99611	0.99638	0.99638	0.99619	0.99648	0.99629	0.99635	0.9954	0.99565	0.996	0.9975
	T	0:00:13	0:00:10	0:00:07	0:00:15	0:00:11	0:00:13	0:00:25	0:00:07	0:00:17	0:00:21	0:00:12	0:00:31
5	MSE	0.000583	0.00058	0.000568	0.000562	0.000527	0.000564	0.000576	0.000568	0.000551	0.000587	0.000585	0.000535
	R	0.9979	0.9977	0.99405	0.9977	0.99768	0.99779	0.99713	0.99731	0.99312	0.99739	0.9979	0.99767
	T	0:00:10	0:00:15	0:00:10	0:00:10	0:00:20	0:00:25	0:00:40	0:00:17	0:00:32	0:00:22	0:00:35	0:00:32

6	MSE	0.00056	0.000534	0.000553	0.000559	0.000554	0.000497	0.000537	0.000516	0.000521	0.000549	0.000508	0.000552
	R	0.99609	0.99698	0.998	0.99785	0.998	0.99073	0.9966	0.98771	0.98789	0.99756	0.99745	0.99767
	T	0:00:19	0:00:23	0:00:17	0:00:17	0:00:23	0:00:22	0:01:05	0:00:52	0:01:01	0:00:30	0:00:35	0:00:34
7	MSE	0.00054	0.000495	0.000487	0.000517	0.000446	0.000505	0.000511	0.000484	0.000525	0.000487	0.000491	0.000515
	R	0.99767	0.99295	0.99741	0.99786	0.9968	0.99805	0.99666	0.99636	0.99662	0.99654	0.9955	0.99808
	T	0:01:23	0:00:30	0:01:02	0:00:49	0:00:54	0:00:38	0:00:46	0:00:47	0:01:27	0:01:11	0:00:59	0:01:01
8	MSE	0.000464	0.000475	0.000483	0.000466	0.000477	0.000488	0.000444	0.000462	0.000503	0.000475	0.000455	0.000457
	R	0.99545	0.99481	0.99335	0.99773	0.99591	0.9974	0.99782	0.99716	0.99603	0.99781	0.99774	0.99705
	T	0:00:49	0:00:45	0:00:57	0:01:17	0:00:54	0:00:45	0:00:45	0:00:48	0:00:58	0:01:06	0:01:07	0:01:00
9	MSE	0.000472	0.000446	0.000436	0.000445	0.000437	0.000470	0.000450	0.000456	0.000457	0.000456	0.000434	0.000467
	R	0.99794	0.99528	0.99641	0.99794	0.99637	0.99673	0.99482	0.99641	0.99632	0.99593	0.99709	0.99626
	T	0:02:05	0:01:27	0:01:25	0:01:48	0:01:29	0:01:45	0:01:25	0:01:30	0:01:27	0:01:35	0:02:26	0:02:37
10	MSE	0.000425	0.000416	0.000437	0.000462	0.000411	0.000411	0.000416	0.000386	0.000423	0.000435	0.000452	0.00041
	R	0.99712	0.99754	0.99528	0.99682	0.99712	0.9955	0.99557	0.99596	0.99796	0.99586	0.99793	0.99351
	T	0:01:35	0:01:06	0:03:06	0:02:56	0:01:40	0:01:42	0:01:55	0:03:19	0:02:40	0:02:25	0:02:58	0:02:25

CASE-2													
Scaled Conjugate Gradient Training Algorithm													
Delay Parameter (d)	**Performance Metrix**	**Iteration**											
		1	2	3	4	5	6	7	8	9	10	11	12
2	MSE	0.00168	0.00152	0.00157	0.00154	0.00155	0.00147	0.00147	0.00162	0.00150	0.00159	0.00170	0.00167
	R	0.99426	0.99441	0.99411	0.99471	0.99386	0.99478	0.99469	0.99444	0.99478	0.99422	0.99337	0.99393
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
3	MSE	0.0013	0.00122	0.00125	0.00103	0.000959	0.000957	0.00105	0.000952	0.000987	0.00118	0.00121	0.00109
	R	0.9959	0.99582	0.99582	0.996	0.99555	0.99594	0.99589	0.996	0.99564	0.99533	0.99564	0.99591
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

4	MSE	0.000802	0.00104	0.000968	0.00102	0.000893	0.000925	0.000885	0.000783	0.000907	0.000755	0.00113	0.000999
	R	0.99604	0.99619	0.996	0.99614	0.99591	0.99618	0.99616	0.99609	0.99614	0.99625	0.9961	0.99623
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
5	MSE	0.00104	0.009	0.000936	0.000828	0.00101	0.000761	0.000703	0.000983	0.00097	0.00102	0.000781	0.00102
	R	0.99644	0.9966	0.9961	0.99648	0.99633	0.99652	0.9966	0.99655	0.9959	0.9963	0.9966	0.99658
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
6	MSE	0.00105	0.000992	0.00112	0.00085	0.000986	0.000802	0.000882	0.000774	0.00104	0.00084	0.00124	0.000921
	R	0.99642	0.99598	0.99492	0.99654	0.9961	0.99625	0.99647	0.99656	0.99605	0.99624	0.99607	0.99651
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
7	MSE	0.000858	0.000901	0.000889	0.000878	0.000751	0.000776	0.00084	0.000811	0.00101	0.000937	0.000805	0.000867
	R	0.9964	0.99638	0.99635	0.99635	0.99645	0.99633	0.99643	0.99631	0.99645	0.9964	0.99605	0.99644
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
8	MSE	0.000958	0.000952	0.000944	0.000912	0.000904	0.000819	0.000971	0.000707	0.000743	0.000934	0.000997	0.00103
	R	0.99608	0.99633	0.9962	0.99629	0.99633	0.99632	0.9963	0.99637	0.99635	0.99637	0.99566	0.99634
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
9	MSE	0.000907	0.000945	0.000893	0.000854	0.000803	0.000729	0.000747	0.000743	0.000916	0.00105	0.000974	0.00103
	R	0.99616	0.99631	0.99632	0.99622	0.99621	0.99637	0.99628	0.9955	0.99615	0.99587	0.99625	0.9956
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01
10	MSE	0.00105	0.000959	0.000767	0.000832	0.00076	0.000859	0.000857	0.00133	0.000905	0.000726	0.000972	0.00116
	R	0.99645	0.99628	0.99663	0.99638	0.99659	0.99617	0.99632	0.99528	0.99661	0.99663	0.99588	0.99611
	T	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01	0:00:01

C. Sample MATLAB code

```
% Created 14-Aug-2019 11:01:06
% This script assumes these variables are defined:
% ThesisInputCPUMemoryQueueTime - input time series.
% ThesisTargetCPUMemoryQueueTime - feedback time series.
X = tonndata(ThesisInputCPUMemoryQueueTime,true,false);
T = tonndata(ThesisTargetCPUMemoryQueueTime,true,false);
% 'trainlm' Levenberg-Marquardt backpropagation.
% 'trainbr' Bayesian Regularization backpropagation.
% 'trainscg' Scaled conjugate gradient backpropagation.
trainFcn = 'trainlm';
% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:5;
feedbackDelays = 1:5;
hiddenLayerSize = 10;
net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};
% Prepare the Data for Training and Simulation
x,xi,ai,t] = preparets(net,X,{},T);
net.divideFcn = 'dividerand';
net.divideMode = 'time';
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
% Performance Function
net.performFcn = 'mse';  % Mean Squared Error
net.plotFcns = {'plotperform','plottrainstate', 'ploterrhist', ...
    'plotregression', 'plotresponse', 'ploterrcorr', 'plotinerrcorr'};
% Train the Network
[net,tr] = train(net,x,t,xi,ai);
% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
% Recalculate Training, Validation and Test Performance
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
```

```
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)
% Closed Loop Network
netcloseCPUMemoryQueLMDelay5 = closeloop(net);
netcloseCPUMemoryQueLMDelay5.name = [net.name ' - Closed Loop'];
view(netcloseCPUMemoryQueLMDelay5)
[xc,xic,aic,tc] = preparets(netcloseCPUMemoryQueLMDelay5,X,{},T);
yc = netcloseCPUMemoryQueLMDelay5(xc,xic,aic);
closedLoopPerformanceCPUMemoryQueueDelay5 = perform(net,tc,yc)
% Multi-step Prediction
numTimesteps = size(x,2);
knownOutputTimesteps = 1:(numTimesteps-12);
predictOutputTimesteps = (numTimesteps-11):numTimesteps;
X1 = X(:,knownOutputTimesteps);
T1 = T(:,knownOutputTimesteps);
[x1,xio,aio,t1] = preparets(netcloseCPUMemoryQueLMDelay5,X1,{},T1);
[y1,xfo,afo] = netcloseCPUMemoryQueLMDelay5(x1,xio,aio);
x2 = X(1,predictOutputTimesteps);
[netcloseCPUMemoryQueLMDelay5,xic,aic] =
closeloop(netcloseCPUMemoryQueLMDelay5,xfo,afo);
[y5,xfc,afc] = netcloseCPUMemoryQueLMDelay5(x2,xic,aic);
multiStepPerformanceCPUMemoryQueueDelay5 =
perform(netcloseCPUMemoryQueLMDelay5,T(1,predictOutputTimesteps),y5)
CPUMemoryQueueLMDelay5PredictionValue = cell2mat(y5);
Error = (TestFor12timestepCPUMemQueue -CPUMemoryQueueLMDelay5PredictionValue)
LMCPUMemoryQueueTimemae5 = mae(Error)
LMCPUMemoryQueueTimeRmse5 = sqrt(mse(Error))
plot(MAPEALLCPUMemQueueTimeLMD5, 'b')
grid on, hold on
plot(MAPEALLCPUMemQueueTimeLMD6, 'black')
plot(MAPEALLCPUMemQueueTimeBRD5, 'g')
plot(MAPEALLCPUMemQueueTimeSCGD5, 'r')
axis([0, 13,0,18]);
xlabel('Step Number');
ylabel('MAPE(%)');
grid on, hold on
legend('LM-D5','LM-D6','BR-D5','SCG-D5','location','northeast')
```

D. Snapshot of Training NARX Model