2017-07-14

Short-Term Forecasting of Electric Loads Using Nonlinear Autoregressive Artificial Neural Networks with Exogenous Multivariable Inputs

Jaime H. Buitrago
University of Miami, jaime.buitrago@gmail.com

UNIVERSITY OF MIAMI

SHORT-TERM FORECASTING OF ELECTRIC LOADS USING NONLINEAR AUTOREGRESSIVE ARTIFICIAL NEURAL NETWORKS WITH EXOGENOUS MULTIVARIABLE INPUTS

By

Jaime H. Buitrago

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2017

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SHORT-TERM FORECASTING OF ELECTRIC LOADS USING NONLINEAR AUTOREGRESSIVE ARTIFICIAL NEURAL NETWORKS WITH EXOGENOUS MULTIVARIABLE INPUTS

Jaime H. Buitrago

Approved:

BUITRAGO, JAIME H. (Ph.D., Industrial Engineering)
Short-Term Forecasting of Electric Loads Using Nonlinear Autoregressive Artificial Neural Networks with Exogenous Multivariable Inputs (August 2017)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Shihab Asfour
No. of pages in text. (125)

Short-term load forecasting is crucial for the operations planning of an electrical grid. Forecasting the next 24 h of electrical load in a grid allows operators to plan and optimize their resources. The purpose of this study is to develop a more accurate short-term load forecasting method utilizing non-linear autoregressive artificial neural networks (ANN) with exogenous multi-variable input (NARX). The proposed implementation of the network is new: the neural network is trained in open-loop using actual load and weather data, and then, the network is placed in closed-loop to generate a forecast using the predicted load as the feedback input. Unlike the existing short-term load forecasting methods using ANNs, the proposed method uses its own output as the input in order to improve the accuracy, thus effectively implementing a feedback loop for the load, making it less dependent on external data. Using the proposed framework, mean absolute percent errors in the forecast in the order of 1% have been achieved, which is a 30% improvement on the average error using feedforward ANNs, ARMAX and state space methods, which can result in large savings by avoiding commissioning of unnecessary power plants. In addition, in order to improve the robustness of the forecast to variations in the number of neurons and other network parameters, the author proposes a method using an exponential decay of the error weights for training the neural network. The modification consists in giving higher error weight to more recent values and lower weight to older values of the training set. By doing this, mover recent values have a higher influence on the calculation of the synaptic weights and therefore the forecast produced by the NARX network is more accurate. This method, combined with the use of Bayesian regularization for training, results in improved forecast accuracy of up to 25% and robustness to variation in parameter selection. The New England electrical load data are used to train and validate the forecast prediction.

To my wife, Monica, for her constant love, support, understanding and inspiration, she has been my beacon, my partner and my lifetime friend. To her, forever, my love. To my daughters Helena, Sofia and Daniela, whose lives have filled my life with happiness and encouragement. To my parents from whom I have drawn character and motivation throughout my life, my eternal gratitude. To my sister Leonor, and my brother Rafael, whose exemplary lives have taught me the values of integrity, kindness and humanity.

Acknowledgments

I would like to express my special appreciation and thanks to my advisor, Dr. Shihab Asfour, for his constant encouragement, financial support and mentorship. His advice on both research as well as on my work at the Industrial Assessment Center have been invaluable.

I would also like to thank my committee members, Dr. Ramin Moghaddass for his thorough questions on neural networks and forecasting; Dr. Murat Erkoç and Dr. Moataz Eltoukhy for serving as my committee members with diligence and commitment. They have made my proposal and defense a challenging endeavor, which I greatly value. Without their input and valuable suggestions this work would not have been possible.

I would like to thank the U.S. Department of Energy, whose financial support through a Research Assistanship at the Industrial Assessment Center, allowed me to pursue my Ph.D. program. The DoE's IAC Program data was used for some of the work published under this research.

My special gratitude to my friend and colleague Ahmed Abdulaal for all his help during my Ph.D. program. He served as a sounding board for all my ideas, good and bad, and his great knowledge, good listening skills, but specially his talent and constant support have made my years at this program a valuable learning time. I value the most his friendship which I will forever hold.

I would like to thank my other IAC Ph.D. fellow student Patricia Gomez for her constant encouragement and great disposition. She was a driving force for all the work we performed as a team. I greatly admire her for her dedication and ability to manage many different things at the same time, all of them successfully without ever losing her calm and nice character. As she will be the next Ph.D. candidate, all the best wishes and encouragement.

To my close friends at the Industrial Assessment Center and the College of Engineering throughout the years, they made my experience a pleasant one: Gabriela Cabrera, Joel Zahlan, Jacobo Saldarriaga, and Zahra Karim. Thank you for your friendship.

Special thanks to Augusto Roca, for his support in all the academic and registration issues. Daritza Berrio and Praxie Alegria for their continuous help with the day-to-day administrative tasks necessary for the completion of my Ph.D. program. I always enjoyed their nice demeanor and graceful conversation.

A special thanks to my family. Words cannot express how grateful I am to my wife Monica. She supported my and encouraged me to get my Ph.D after so many years of professional practice, and made my dream finally come true. Her friendship and company kept me going when I needed it the most. My daughters Helena, Sofia and Daniela, from whom I have learned so much. They have served as my inspiration and motivation. I hope my persistence serves them as example to pursue and accomplish all their life goals. My parents, Jaime and Constanza, whose sacrifices have made may education and life possible. My sister Leonor, and my brother Rafael, the best examples in my life for character, discipline and their never ending pursuit of higher ideals. They have supported me every step of the way. Their lifetime encouragement has made me advance even when facing difficulties.

Jaime H. Buitrago

University of Miami

August 2017

Contents

List of Figures

List of Tables

List of Abbreviations

AGC	Automatic Generation Control
AIC	Akaike's Information Criterion
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Networks
ARIMA	Autoregressive Integrated Moving Average model
ARIMA	Autoregressive Moving Average
ARIMAX	Autoregressive Integrated Moving Average with Exogenous Variables
ARX	Auto Regressive Model with Exogenous Input
BIC	Bayesian Information Criterion
CI	Computational Intelligence
DFT	Discrete Fourier Transform
EP	Evolutionary Programming
ERNN	Elman Recurrent Neural Network
FBTFS	Feedback Tuning Fuzzy System

FFT	Fast Fourier Transform
FL	Fuzzy Logic
FLS	Fuzzy Logic System
GA	Genetic Algorithm
MAPE	Mean Absolute Percent Error
MARS	Multivariate Adaptive Regression Spline
MLFFN	Multi Layer Feed Forward Network
MLP	Multi Layer Perceptron
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RBFN	Radial Basis Function Network
SOM	Self Organizing Maps
STLF	Short-Term Load Forecasting
SVM	Support Vector Machine
SVR	Support Vector Regression
WNN	Wavelet Neural Network

Chapter 1

Introduction

1.1 Motivation

Electrical load forecasting plays an important role in the economics of modern power systems. Decisions are made based on the expected value of energy consumption at different time scales. Long-term forecasts, those including time frames between 5 and 20 years are useful for planning the resources that need to be available at any point in time and expanding them if needed. Power generation, transmission and distribution grids are built based on long-term economic analyses. Medium-term forecasts, those including predictions for energy demand from 1 month to 5 years are used for allocating resources, planning operations and establishing economic factors such as energy tariffs, rate schedules and governmental regulations. Short-term load forecasting, usually those including time frames of 1 hour to 1 week, are crucial in scheduling power generation, optimization of transmission network flow rates, generator shut downs, etc.

The forecasting horizon has great impact on the types of predictors that need to be selected for a comprehensive model:

Long-term Forecast Horizon Long-term forecasts depend mostly on economic variables, such as population growth, economic expansion and economic development, cost of fuels, and long-term factors such as global warming. A long-term forecast seeks general trends in energy consumption in order to determine the convenience of building new energy generation systems, reveals geographic locations where demand is rising faster and therefore aids in planning transmission network construction, and helps analyzing the economics of generation, transmission and distribution on a large scale v. microgrid implementation and the general impact of renewable energy in the overall energy economy.

Medium-term Forecast Horizon Medium-term forecasts are usually more dependent on large scale meteorological phenomena, such as El NiÃśo weather patterns, global trends and costs in energy production, such as the adoption of micro-grids, solar energy and wind energy. Medium-term forecasts help in deciding maintenance schedules for power generation facilities and transmission lines, plan for equipment lifespan, commissioning of existing and new power plants and demand location for operations planning.

Short-term Forecast Horizon Short-term forecasts are more dependent on weather variables, season, time of day and whether the particular day is a working day, a holiday or a weekend. Short term electrical load forecasting (STLF) is vital for the efficient operation of electric power systems. A power grid integrates many stake holders who can be affected by an inaccurate load forecast: Power generation utilizes 24-hour ahead forecasts for operations planning, i.e. to determine which power sources should be allocated for the next 24 hours; transmission grids need to know in advance the power transmission requirements in order to assign resources; end users utilize the forecast to calculate energy prices based on estimated demand. Contingency planning, load shedding, management strategies and commercialization strategies are all influenced by load forecasts. Forecast errors result in increased operating costs [1]: Predicting a lower load than the actual load results in utilities not committing the necessary generation units and therefore incurring higher costs due to the use of peak power plants; on the other hand, predicting a higher load than actual will result in higher costs because unnecessary baseline units are started and not used. Reliable forecasting methods are essential for scheduling sources and load management [2].

Univariate time-series statistical analysis has been proposed with success in long-term forecasting but inaccuracy on short-term forecasts. The Box-Jenkins method has been used successfully and it will be covered in 3 in detail, with good results. Other univariate methods used include logistic modeling and Kalman filtering. Generally speaking, univariate statistical methods are slow, require constant user intervention and may not converge.

As a result of the above, the correct selection of a subset of variables to analyze the problem is crucial in the outcome of the forecast. This topic will be analyzed further in Chapter 3.

In order to understand the motivation of the present research, we must understand the problems associated with the signal under study. The next section focuses on this topic. The last section of the chapter presents the approach that this work takes to tackling the problem of short term load forecasting.

1.2 The Problem

Electrical energy load for a large system is a variable with many complexities. The very nature of power consumption corresponds to a countless number of scenarios that cannot be analytically solved and reflect individual decisions on power usage of small to large loads that make its study an exercise of a stochastic nature. Deterministic models are difficult because of the number of variables and decision triggers that are complex and in many occasions random in nature. As an aggregate, the need for energy and its demand are more predictable because the aggregated trends and behavior mimic many characteristics of well-studied deterministic and stochastic models. However, the superimposition of different models does not necessarily result in a good forecast model for energy usage. It is then important to take a first step by looking into the nature of the data and drawing some conclusions about its behavior and try to fit existing techniques to model it. In light of this study, we will start to reveal the advantages of the proposed model. Section 3.1 describes the nature of the data in detail and provides insight in the different characteristics that make this time-series model unique.

1.3 The Proposed Solution

This paper proposes a method of obtaining accurate short-term load forecasts by using Artificial Neural Networks in recurrent mode with weather-related variables used as Exogenous Inputs. The method has been developed in two parts: One initial model, and then an analysis of all the parameters that affect the accuracy and robustness of the initial model.

The model proposed produces a 24-hour forecast as the output vector of a fully connected ANN with one hidden layer and sigmoid functions as activation functions in the hidden nodes. The output nodes use a linear activation function. Inputs are historical values of load with redundancy (more recent values fed again as input) in order to increase the weight of recent data values. The input vector also includes day, month, day of the week, and whether or not the day is a working day. Weather information is used as exogenous input. The architecture of the network is such that the neural network is trained in open-loop using actual load data in order to calculate the node weights, and then the network is used in closed-loop to generate the forecast applying the forecast load as input. The results show an improvement of the forecast when compared with ANN models without exogenous inputs.

The second part of the work focuses on the analysis of the network parameters and their impact in the forecast accuracy, the error standard deviation and the robustness of the model to changes in parameters such as number of neurons. The author proposes the use of an exponential decay of error weights as a method to train the network such that the newest data has greater impact in the generalization of the solution. This method, in combination with the use of Bayesian regularization for training, results in improved accuracy and robustness so that changes in the number of neurons do not impact the forecasting accuracy. Monte Carlo simulations are run with different parameters and their relative impact is studied.

This research lies in the intersection of the fields of data forecasting, electrical load forecasting, artificial neural networks and time-series forecasting. Therefore we will briefly review the relevant information for each topic.

Chapter 2

Literature Review

The proposed research lies in the intersection of three main fields: Data Forecasting, Energy Forecasting, and Artificial Intelligence. More specifically, Time-series Forecasting, Short Term Load Forecasting, and Artificial Neural Networks (see Figure 2.1). Methods developed for data forecasting have applicability in energy forecasting. In particular, methods dedicated to predicting future values for time series have applicability. On the other hand, energy forecasting, and in particular short-term load forecasting contribute insight into data behavior, such as seasonality and cyclical studies. Artificial Intelligence, and in particular Artificial Neural Networks contribute methods of data analysis that can be used to increase the accuracy of the forecast. The combined study of all three disciplines provides a multi-disciplinary approach that benefits from the contributions of each subject. There are synergies derived from the use of statistical methods to select data subsets, to insights on data behavior by energy studies, to forecasting methods using ANNs that combined yield a much superior result. That is why we will select some specific topics from each discipline for the literature review.

The history of short term load forecasting includes many different approaches. Economists have developed multivariate causal mathematical models that include

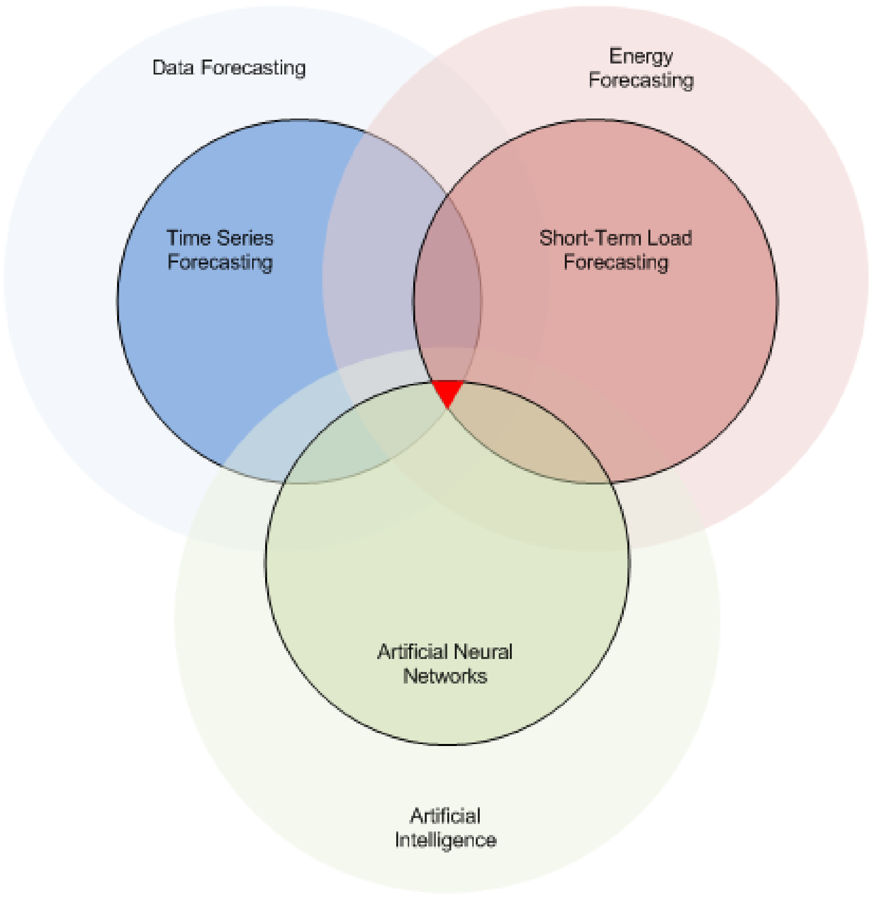


Figure 2.1: Research Intersection: Where this research belongs

population growth, human behavior, economic variables such as gross domestic product, etc. by using traditional econometric models [3]. Statisticians have developed univariate [4] and multiple regression models using many different variables [5, 6] according to the forecasting horizon as discussed in 1.1. Univariate models of different success rates have been presented with success over the long-term horizon – exponential smoothing has been particularly accurate in predicting annual energy consumption in local markets.

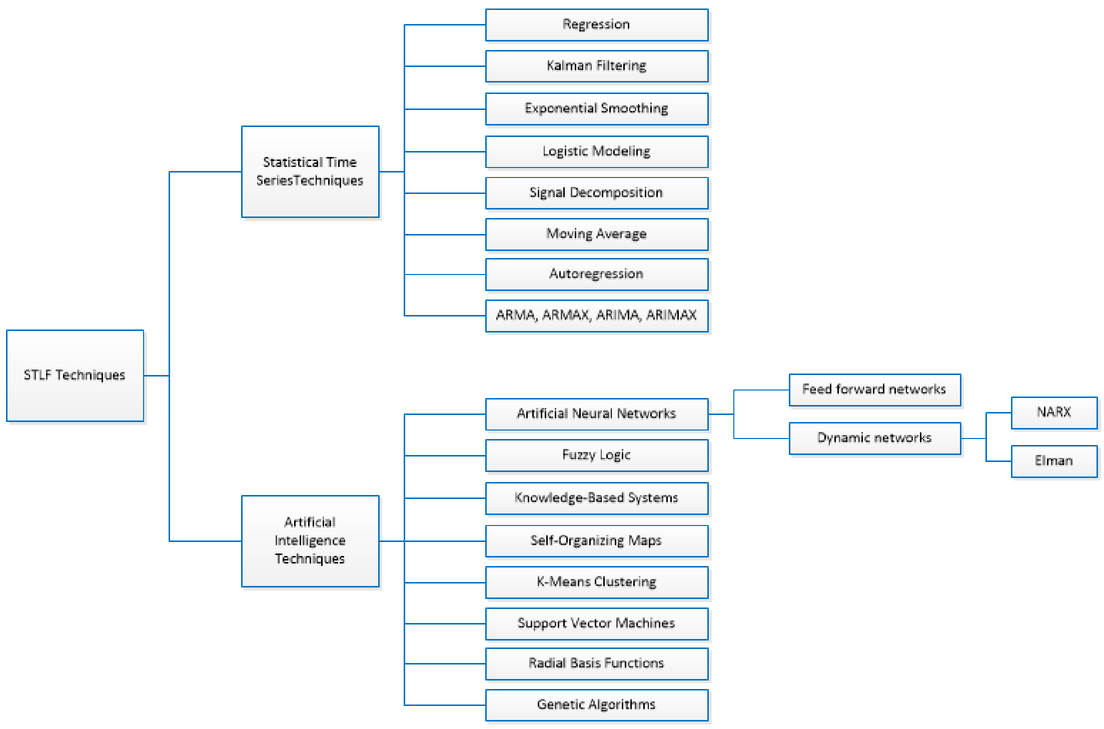


Figure 2.2: Short-Term Load Forecasting Techniques

2.1 Data Forecasting

2.2 Time-series Forecasting

Time-series forecasting is a discipline that arises from the need to predict values for a series of points that are arranged in a sequential historical manner, typically on even time intervals. Most common time-series forecasting methods include moving average, weighted moving average, linear quadratic estimation, exponential smoothing, autoregressive moving average (Box-Jenkins), extrapolation, linear prediction, trend estimation and growth curve.

In causal forecasting methods, the assumption is that the variable y that we want to forecast has a cause-effect relationship with one or more independent variables x. We can express this relationship as

$$y = f(x_1, x_2, \ldots, x_n) \tag{2.1}$$

Notice that the output y can be predicted at any time if the input variables x are known. On the other hand, in time-series forecasting the output for the next time interval y_{t+1} is calculated based on the previously generated outputs y_t, and even on previous time interval outputs y_{t-p}. This means that it is only possible to predict a future value if we know the past values of the output. This relationship can be expressed as

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-p}) \tag{2.2}$$

The statistical approach to time-series forecasting includes a step in which a mathematical model is selected to fit the data. This usually requires knowledge of the causal relationship between input and output. In some cases this is a simple decision, such as in the case of linear, quadratic or exponential processes in which the dynamics of the process can be inferred, but most likely, especially in the case of non-linear relationships, the model is selected by trial and error or from experience.

2.3 Short Term Load Forecasting

Different types of load forecasts are used in the industry for multiple purposes. Long-term forecasts (5-20 years ahead) are used for capacity planning and investment purposes. Medium-term forecasts (a few months to 5 years) are used for commissioning, maintenance and distribution planning. Short-term forecasts (hours to weeks) are used for operations planning and scheduling. One of the main differences between short-term and long-term load forecasting is that in long-term forecasting a single value is sought whereas in short-term forecasting a load curve is desired (for the next 24 hours). Hahn [7] presents a review of how these forecasts are used in decision making processes. Also, Rahman [8] reviewed the techniques for short-term data forecasting in 1994 and proposed a standard for comparing accuracy, which includes the development of a standardized database, two standard lead times (24 hours and 168 hours), and a single error measurement definition (absolute relative error). This section presents a brief review of the methods used for calculating a short-term load forecast. The purpose is to reflect the different approaches by some authors and attempt to highlight their contribution in relation to the current research.

2.3.1 Statistical Methods

Several statistical forecasting methods have been successful in predicting short-term load forecasts. El-Hawary [9] utilizes statistical autocorrelation and partial autocorrelation models to calculate the weights on a parameter estimation model. In his paper, Nengling [2] highlights the importance of properly studying the differences in load characteristics of weekdays, holidays and weekends. The author uses the wavelet transform to perform a decomposition of the load signal into its components in frequency domain in order to improve forecast accuracy on weekdays, weekend and holidays separately. Wang [10] uses ARMAX and particle swarm optimization to avoid local minima in the search for the minimum error. In contrast, Pappas [11] uses an ARMA model to estimate loads. In his work Kang [12] proposes the use of an integrated forecasting model making use of different techniques. A comparison study is presented in Goh [13] for extrapolative methods versus stochastic models. In his paper Chen [14] proposes the use of an ARMA algorithm for short-term load forecasting. In his work Almeshaie [15] uses decomposition and segmentation in an algorithm for power forecasting.

2.3.2 Computational Intelligence Methods

The most common computational intelligence forecasting methods are artificial neural networks, group methods for data handling, fuzzy logic, support vector machines, data mining, machine learning and pattern recognition. Other methods include simulation, prediction market and probabilistic forecasting, and ensemble forecasting.

Artificial Neural Networks

A comprehensive review of the current techniques for using Artificial Neural Networks in short-term load forecasting can be found in [16, 17, 5]. For a review of the use of ANNs in energy systems in general, see Kalogirou [18].

From the theoretical point of view, an ANN can approximate any continuous function under loose conditions. Several decomposition theories have been used going from the simplest linear regression to ANNs. However, the main reason why ANNs are effective time-series forecasters is that there is no need to start from a mathematical model of the data. For example, electric load signals could be decomposed into constant, trend, polynomial, periodic and noise components. Each one of the components could be modeled accurately by one of many methods including Taylor series, Fourier Series, linear regression, etc. But all of these methods assume a shape for the data, and therefore require *a priori* assumptions. ANNs on the other hand make no assumptions and the node weights are the result of a training mechanism that minimizes the error between the data and the network output. The reason ANNs work is explained below.

Weierstrass Theorem In a restricted definition, if $C([a,b])$ denotes the space of continuous real valued functions defined on the interval $[a,b]$ with the norm of

$f \in C([a,b])$ defined by

$$||f|| = \sup_{t} \{|f(t)| : t \in [a,b]\ \} \tag{2.3}$$

Where f is a continuous function $f : R^n \to R^m$, then the Weierstrass Theorem states that any such f function in $C([a,b])$ can be approximated arbitrarily closely by a polynomial [19]. This is a very important first step. We can use polynomials to approximate functions arbitrarily closely.

Artificial Neural Networks If there is a functional relationship between input vector $u(t)$ and the output vector $y(t)$ such that $y(t) = G(u(t))$, then there exists an integer n where $0 \leq n < \infty$ such that given an arbitrary $\delta > 0$,

$$|G(u(t)) - \sum_{i=1}^{n} w_i \sigma_i(u(t))| < \delta \tag{2.4}$$

In equation 2.4, similar to the definition of the limit, w_i are the layer weights, and σ_i are the activation functions for the nodes. The importance of this is that n is a finite number, which means that given any continuous function, it is always possible to find a neural network with n nodes capable of mapping it [20, 21]. This is valid for any functional relationship regardless of differentiability, periodicity or continuity. In practical terms, this is the reason why ANNs are widely used for forecasting. General functions can be represented by a finite linear combination of weighted activation functions. The goal of an artificial neural network is to find the weights of the activation function in each node that minimize the error between the training data set and the output resulting from the linear combination of activation functions. Once those weights are calculated, the neural network can be used to predict future values of the function.

One of the challenges facing the widespread application of ANNs as the *de facto* forecasting method for time-series data is the lack of a methodology for building parsimoniously parametrized neural networks. The objective is to obtain a neural network architecture that uses the simplest possible structure and smallest set of input data and still obtain good forecasting results with acceptable computational performance. Terasvirta [22] presents some models and viewpoints on accomplishing this based on statistical inference, but further research is necessary in the field. White [23] proposes a statistical correlation model of the main structural parameters for estimating the optimal number of hidden layers in an ANN.

ANNs are used in combination with other techniques in order to address specific deficiencies. Ghayekhloo [24] uses a hybrid time-series and regression analysis to select the best sets on inputs, and then a genetic algorithm (GA) to process the weights. Satish [1] uses a multi-ANN method to address the day-of-week problem which affects most statistical methods. Buitrago *et al* [25] and Abdulaal *et al* [26] propose a framework combining parameter estimation, clustering and ANNs to group load patterns. Lopez [27] uses self-organizing maps (SOM) for addressing the groupings of load patterns. Liang [28] uses differential dynamic programming together with ANNs trained with supervised and unsupervised learning algorithms. Zhang [17] uses wavelet decomposition and an adaptive ANN model for pricing purposes. The motivation of this research is that no attempt has been found in the references to model short-term loads using non-linear autoregressive ANNs. Some of the methods utilize very short data sets, up to 1 week in order to make the prediction for the next 24 hours [29]. Other methods use a multi-layer perceptron (MLP) to determine the forecast utilizing one-year data and temperature as a predictor [30]. Most ANN methods have results with a MAPE error of around 2-4 %, compared to statistical methods which show errors in the order of 5-6%. The goal is to obtain a consistent forecast error which is lower than prevailing statistical methods.

The work by Andalib [31] shows a NARX implementation of a neural network for forecasting energy prices. Although unrelated to load forecasting, the author compares results with other ANN techniques and the results clearly show an advantage of the NARX implementation over multivariate adaptive regression spline (MARS) and wavenet networks.

2.4 Short Term Load Forecasting Utilizing Artificial Neural Networks

Review Papers

Raza [32] wrote a review of some of the most common techniques used in STLF forecasting, and included description of the ANN model and transfer functions most commonly used. The authors included a comprehensive bibliography on the methods described in their paper.

Tzafestas [6] wrote a review paper on the computational techniques for short term electric load forecasting. The techniques reviewed were MLFFN, FL, GA, and chaos. Then the authors went on to review hybrid techniques: ANFIS, GARIC, Fuzzy ART, KB-NN, Chaos-FL, and NF GA. The author found a MAPE for MLFFN of 1.7%, with some improvement by using combined techniques.

Multi-layer Feedforward Neural Network Approach

Abdel [4] developed a univariate model for medium-term load forecast utilizing abductive and neural networks. The authors compared the performance (MAPE based) between the abductive networks model and the neural network model. The MAPE of the neural network model was about 4%. The ANN used by the authors in this research was a multi-layer feedforward neural network (MLFFN). Since their model was univariate, only historical values of load data were used. The authors utilized 6 years of load data, normalized in order to remove the trend, and used a single-next month model for their iterative forecasting. The authors compare two different approaches: a single 12-month stream of data and 12 separate one-month streams of data. The single stream method proved more accurate, 3.2% vs. 3.8%. The authors claim better MAPE performance with the ANN than ARIMA and other forecasting methods.

Alfuhaid *et al* [33] utilize a cascaded artificial neural network (CANN) to produce a forecast that includes peak, minimum and daily energy as additional input data for the final forecast stage. The authors utilize two neural networks: one small network to predict the peak, maximum and minimum loads; another network that uses the output of the first network, to predict half-hourly loads for the next day. The first network cascades onto the second. Researchers used data from the network in Kuwait. The average (MAPE) forecasting error by applying CANN is 2.707% as opposed to 3.367% when using conventional ANN (FitNet).

The work by Bennett [34] describes the comparison between an ARIMAX model and a MLFFN ANN model to forecast low voltage next day total energy use and next day peak demand. The authors propose a hybrid model incorporating a double exponential smoothing algorithm, autoregressive terms, relative humidity and day of the week dummy variables to increase accuracy. The ARIMAX model included linear and quadratic terms for the temperature and relative humidity as a multiplying term for both. The double exponential regression term was introduced in order to account for the general trend. The ARIMAX model yields a MAPE of an average 7% and the ANN model a MAPE of 6%. Data was taken from a transformer serving 128 residential customers. The authors found that daily average temperature explained half of the observed variance in the model. The ANN used was a feedforward classical FitNet network. The hybrid ARIMAX-ANN model showed improvement in forecast accuracy and fit.

The work done by Bilgic [35] shows an application of an ANN network for forecasting hourly load 24-hours ahead. The authors used previous research by others for comparison purposes. The research used a multilayer ANN with 5 years of scaled data. The best results are with one, three and five hidden neurons. They tested the algorithm for each of the 9 geographical regions, with MAPE results in the order of 3%. Then, they tested the model on the aggregate of all regions with MAPE around 1.85% for the aggregate. Compared with previous results for the same data set by other authors, the ANN model showed improvement in accuracy. Kandil [36] uses a MLFFN network trained with a backpropagation algorithm to achieve 0.981% error in forecasting 24-hour loads for Hydro-Quebec.

Bugwan[37] studies the energy consumption of Mauritius with different training schemes. The first scheme, used for comparison purposes uses a supervised training MLFFN ANN. The second scheme uses a three step hybrid process in which data clustering, by means of a Kohonen Self Organizing Maps (SOM) clustering technique, is used to group load by type, identifying the different day use patterns. The second step updates the information manually in order to incorporate the results of the first phase into an input model for the third step. The third step is to run a MLFFN ANN with a backpropagation learning algorithm using supervised learning. MAPE results for the first model run in the order of 4.15%, whereas using the hybrid model MAPE is in the order of 3.9%.

The work presented by Charytoniuk [38] focuses on predicting very short term

(60-90 minutes) forecast based on the incremental changes of load during the day before. The author claims that incremental changes of load during a day are more predictable than actual loads and therefore his model focuses on incremental load forecasting. The research mentions that this method is more robust because it is less dependent on the actual weather conditions, especially when the forecast weather is different from the training weather values. The author uses a MLFFN ANN with only one day data. The author also breaks down the forecast by day type, generating one forecast for each type, and using up to 40 neural networks per day type.

The work by Gooi [39] reveals an ANN network using peak and valley load with a single hidden layer. Inputs include load and predicted weather data. The output predicts either a peak or a valley load depending on whether the model is trained to forecast the peak or the valley load. Once the peak and valley load are predicted, the hourly forecast can be obtained from a linear combination of both. 5 years of historical data were used. Statistical methods are used to classify the inputs according to patterns. *Ad hoc* groups are created based on their similarity, with some groups, like weekend or holiday, added, and some days grouped into similar day types. The model parameters are varied manually in order to improve the forecast. The parameters are number of input nodes, number of hidden nodes, momentum rates and learning rates. Once the model is trained, it is put online in order to use the latest 1-hour load data. MAPE errors obtained are in the range 1.43% to 4.14% for days of the week, and 5.8% for special days.

Tee [40] proposes a model using a model using a MLFFN with 51 inputs, 16 hidden neurons and one output layer. The inputs include 24 dummy variables for time of day, past hour load for 24 hours, one dummy variable for weekend or weekday, temperature at hour before, and month of the year. The author supports the choice of just temperature as a sole weather variable, citing [36, 5] that wind speed, humidity and cloud cover have little influence on the performance of the ANN. The training method is a Levenberg-Marquardt back propagation algorithm. The MAPE achieved by this study is 0.439% with a maximum MAPE of 7.986% for the month of December.

The work by Harun [41] shows a feedforward ANN which has two data preprocessing schemes for comparison: one group does not use differencing of the data and the other uses first order differencing in order to achieve stationarity. The authors run different simulations using lags of 24, 48 and 72 hours, with the latter giving the best results. Also, the stationary model gave better results for most forecast results.

Hernandez [42] uses a MLFFN in order to forecast 24-hour load for a region in Spain. The author's approach is a three-layer MLFFN (input, hidden, output) with inputs variables day of week, month, total day load. The model uses 16 neurons in the hidden layer. A heuristic method is used to select the number of hidden neurons. Training was done via a Bayesian Regulation Backpropagation training function. The mean error MAPE was 2.4037%. Special days had errors in the order of 4%. In a different research work, Hernandez [43] tests two different models of MLFFN in order to forecast total load for a city. The first model is a traditional MLFFN, and the second one is a two-stage modified MLFFN in which a first MLFFN model is run to predict peaks and valleys in load for the next day, and the output of that model is used as input for the regular MLFFN model. The variables considered were precipitation, air temperature, average wind speed, average wind direction, relative humidity, atmospheric pressure and solar radiation. The MAPE obtained by the dual stage MLFFN model was 1.62% and it was 2.47% for the MLFFN.

The work by Kalaitzakis [44] is a comparison of the performance of 9 different methods, ordered from the highest error to the lowest error: Autoregressive (stochastic AR) method; MLFFN trained with backpropagation; adaptive learning rate with back propagation; MLFFN with Gaussian encoding; random activation weight neural network; a weight matrix random activation weight neural network; a zero-order regularization radial basis function neural network; a MLFFN using a real-time recurrent learning algorithm; and an autoregressive recurrent neural network (ARNN). The authors found that the best performance was obtained by an ARNN, with a relative error of 1.22%.

Khotanzad [29] presents an approach with two different generations of the same MLFFN ANN. The first generation broke down the model into three separate ANNs: hourly (yesterday's and two days ago data at this hour), daily (yesterday's data for 24 hours), and weekly (24-hour data for last week on the same day). The second generation broke down the model into different hours of the day (early morning, mid morning, afternoon peak, and late night). Both models utilize an adaptive combiner in order to provide a forecast based on the output of the individual models. The team achieved 2.19% MAPE for the second generation model, compared with 2.52% MAPE for the first generation for a day-ahead forecast.

Kiartzis [45, 46] presents a MLFFN structure capable of achieving 2.52% MAPE by using a three-layer MLFFN and backpropagation training. The model uses 64 inputs: loads for one and two days ago (24 hours), maximum temperature for 1 and 2 days ago, square of the temperature deviation for both days, maximum temperature difference for the past two days, day of the year as a sine wave and a cosine wave, and day of the week. No comparison data is provided with other established models in the work presented.

Matsumoto [47] presents a short-term load forecasting technique for summer loads, using a two-part predictor. The first part is a MLFFN which uses data from the same year only; the second part is another MLFFN that uses data from consecutive years. The forecast produced by the first part is adjusted by the trend found from second module. This way, the model can accommodate variations in trend from one year to another. The first ANN is used to classify inputs using norm as the classifier, then grouping the data in order to select the forecast load to input the second module. The authors found that they can generate forecasts with MAPE of 2.52%.

Moharari [48] shows the implementation of a MLFFN ANN for forecasting short term loads considering special days. The input to the MLFFN network includes day variables such as weekends and holidays, as well as weather variables such as minimum and maximum forecast temperatures, and historical loads for the past 15 days, for a total of 23 inputs. The implementation results in a forecast with a MAPE of 1.43%.

Raza [32] presents a model using an MLFFN trained with a gradient descent algorithm. The inputs to the network include day of week, working day, hour of day, dew point, dry bulb temperature and loads for current day, day before and week before. 20 neurons were used in the hidden layer. The forecast accuracy achieved was separated by season, and it varied from 3.81% in the spring to 4.59% during the summer. The analysis included a statistical insight on the resulting MAPE error with confidence level intervals for the error.

Papalexopoulos [49] shows a model with a MLFFN trained with 77 inputs. The architecture is a feedforward network (FitNet) that uses seasonal input variables in the shape of sine and cosine functions of a period equal to one year in order to accommodate seasonal variations; temperature for today and tomorrow (forecast), indirect temperature variables (average maximum temperature forecast, average minimum temperature forecast) for tomorrow and last week; temperature trend variables such as difference in temperature between today and average maximum, etc.; cooling and heating degrees by day; and historical load variables (averages, trends, peaks). The model achieved a MAPE of 1.783%.

Reinschmidt [50] describes a model with two modules: the first module is a feedforward neural network which produces an ARMA model of the load data; the second module, a recurrent ANN, uses the output from the first module, and in addition other weather variables in order to train the network to produce a load forecast. One subnet is developed for each of the 24 hours of the day, and then their results are combined to produce the complete forecast. The author claims that traditional models utilizing only ANN networks do not have the capability of adapting to sudden changes in weather conditions. The authors do not discuss the accuracy of their forecast.

Santos *et al.* [51] discuss a univariate ANN model that uses minimal load data, only current day, past week and past two weeks load in order to make a forecast using a feedforward ANN. The activation function is a hyperbolic tangent function. One hidden layer was used. In order to incorporate the effects of temperature one additional load variable is added to the input, which is the average expected load for the next day, which is calculated offline by another method. The MAPE obtained by this arrangement is 1.71%. Also, Santos [52] includes reactive power in the input of an ANN model in order to reduce the forecast error. The reason the authors included this variable in their study is that the test data was taken in an area of mild climate, and therefore the correlation between load and weather variables was very low. The approach mentioned in the paper includes pre-processing the data in order to influence the composition of the input vector in such a way as to reduce the margin of discretion in its definition. The results yield MAPE values in the range 3.27% to 4.96%.

Shimakura [30] describes a two-step method for generating a forecast. The idea is based on the fact that there can be a trend factor in the load from one year to another in the same comparable season. In order to remove the problems associated with ANN learning time-series data that contains trends, the author proposes a system in which the trend is first calculated and removed from the data by means of a data compensation process, and then the data is processed in the regular manner by an MLFFN. The trend extracted by the preprocessing algorithm is then added to the forecast of the MLFFN to generate the final forecast values. The author also describes a technique in which some of the weights are not allowed to change (restricted change) in order to minimize overfitting. The MAPE obtained by this model is in the range 2.1% to 2.7%.

Zhang [53] proposes a MLFFN model which uses three layers on the ANN (input, hidden, output). The input includes information on weather, load, and whether the day is a working day, weekend or holiday. Even though the network is not recurrent by design, past values of the load are used in the input vector. The network is trained with a backpropagation algorithm. The MAPE obtained by this method is in the range 1.87% to 3.051%.

Sinha [54] presents a research on a MLFFN that is broken into 6 subnetworks in order to improve training time. Each subnetwork processes 4 hours of the day.

The ANN is trained using a backpropagation algorithm. The error reported was in the order of 3%.

Input Selection Methods

Da Silva [55] focuses his work on the development of the correct set of inputs to the model. The author explains that neural network models are very accurate as long as they do not overfit the data. Neural network models must match the data regularity to the model structure in order to be accurate. Current input selection methods rely on linear correlation analysis. Higher order statistical information is necessary to optimize the model. The author proposes and compares two models: a filtering model and a wrapping model. The first method looks for statistical information relevant to the inputs; the second method uses inference to estimate the relevant error caused by an input, and selects those that reduce the error. The ANN selected varies from 4 to 8 neurons. The resulting MAPE is 2.5%.

The work by Ferreira [56] describes a similar approach as [55], but in this case Support Vector Machines (SVM) are used. SVMs are used as classifiers. The point on which the maximum margin is obtained yields the support vectors. Here the margin is the difference between the training value and the test accuracy. A support vector regression (SVR) is performed on the inputs in order to classify the inputs. The best results were obtained with 84 inputs: 24 dummy variables for the hour of the day, lags, load, temperature and temperature square, maximum forecast temperature value and its square, and their lags by one hour. The assumption is forecast temperatures are perfect. The idea was to present the model with a large number of inputs to identify those that are more significant. The objective was to create a robust model that could select the best input vectors for each forecasting period. Bayesian inference was applied for clustering load dynamics to feed different SVM load forecasting models, and to estimate the SVM learning parameters. The latter model seems to be more promising for STLF.

Santos [57] studies the selection of the input vector to the ANN model. In order to avoid discretionary selection of the input vector, the author proposes the use of an entropy analysis to measure the level of complexity of the finite length time series of the active power. As a result, a minimal number of input variables is used, including the day of the forecast load data, and two preceding weeks of corresponding values. The authors first do a data preprocessing, in order to fill the gaps left by the data acquisition method, then a correlation analysis is made on the different factors, then an entropy analysis (SampEn) on the load time series, then an autocorrelation analysis of the time series to identify the data sequences that contribute the most, then the forecast is done using a MLFFN ANN. Based on the entropy results, only the values with the higher entropy are selected, and then a correlation analysis is done on the inputs. The forecast MAPE obtained by the method was in the range 1.38% to 2.53% for both substations under study.

NARX ANN Networks and ARX Knowledge-Based Models

The study by De Andrade [58], presents an implementation of a dynamic recurrent NARX ANN for load forecasting. The application is an electric substation, and the prediction forecast is for very short term load forecasting (5 minutes) in order to feed an Automatic Generation Control (AGC) in order to maintain the balance between demand and supply of electricity. The network is used in open-loop, i.e. it is trained in parallel identification mode. The researcher used cross-validation in order to determine the structural parameters and the training of the NARX-neural network. The authors divided the data into five days for training, one day for validation and one day for testing. The study was done using one step regressive terms. The number of neurons was fixed at 5 in the hidden layer for the work presented.

Research by Chen [59] presents an implementation of traditional approaches by performing knowledge-based weather segmentation and utilizing multiple autoregression with exogenous variables (ARX) models. The authors utilize load and lagged load variables, dummy variables for special days, past and forecast values for weather variables, but in addition, they classify weather pattern variables into four types of days: normal days, abnormal days, extreme days and transition days. These latter variables are the knowledge-based portion of the model. This model is site-dependent, and requires that the load forecasters have knowledge of and experience with the weather patterns.

Other Computational Intelligence Methods

The research by Fattaheian [60] uses Support Vector Regression (SVR) and Radial Basis Function (RBF) on an ANN network trained with a backpropagation algorithm. First, the authors applied a SVR as a regression mechanism. Then 4 kernels are tried on the results: linear, polynomial, sigmoid and RBF, selecting the best of the 4 in terms of the MAPE of the fit. Then an optimization problem is solved with an objective function which is the same as the objective of the SVR model. A combinatorial model is run in order to obtain the model with the lowest MAPE. That model is the one used for forecasting. A correlation analysis is used on the input variables in order to determine the best subset of inputs.

Hayati [61] compared the performance of three different ANN structures: a) MLFFN with one hidden layer and various configurations for number of hidden neurons; b) Elman recurrent neural network (ERNN), which is a modified MLFFN with feedback from the hidden layer output to the input layer, also with a varying number of hidden neurons; and c) RBFN where the hidden layer clusters the inputs based on a radial basis function, and the output layer performs a linear transformation of the hidden layer to generate the output. The author found that the RBFN network yields the smallest error. The number of centers (neurons in the input layer) was selected manually by the user. The MAPE results were 0.17% for RBFN, 0.38% for MLFFN, and 0.76% for ERNN.

Hernandez [62] tested a three-stage model with a Kohonen SOM as a previous stage, then a K-means clustering algorithm and finally an MLFFN for post-processing. The groupings found by the SOM were: a) working days and holidays; b) seasonal months; c) weekends and weekdays. The clusters used by the K-means were: i) Working days of January, February, March, April, November, December and October 15th to 31st.; ii) Saturdays, Sundays and Holidays; and iii) working days of May, June, July, August, September and October 1st to 14th. The MAPE found by this arrangement was in de order of 2.76%. The work by Hsu [63, 64] is very similar: using a SOM to group the demand patterns and predict the peaks and valleys for the next 24 hours, and the second phase uses a MLFFN to produce the 24 hour load as output by means of a linear transformation. MAPE for the combined system is in the range 1.14% to 1.25%. Similarly, Marin [65] describes a model that uses a SOM for pre-classification of the input into similar load profiles, then uses a recurrent ANN to generate forecast load and then the model is put online in a recall phase with a simplified model, and retraining of the forecasting model is done once a year for adjusting the weights and biases. Their Kohonen SOM uses 15 classes (patterns) of load, e.g. ”Saturdays in August”, which demonstrate similarities. The authors create one Elman ANN (ERNN) for each class, and the results are presented for each network. The average MAPE ranges from

1.03% for Class 14 (Tuesday to Friday of third and fourth weeks of June and July) to 1.71% for Class 8 (Mondays from October to March).

Another method used for comparison is Fuzzy Logic. Badri [66] compares a feedforward neural network (MLFFN) with a Fuzzy Logic (FL) algorithm for 24-hour ahead forecast. The FL algorithm starts from a linear regression on the available data. The fitted data provide a predicted peak load and maximum daily temperature. The authors did not provide information on the length of time for the data used for the MLFFN model. Their results were an accuracy of 0.5% MAPE for the MLFFN and 4.91% for the FL model.

Khosravi [67] utilizes an Interval Type-2 fuzzy logic system (FLS). The idea is to utilize three dimensional fuzzy membership functions in order to accommodate uncertainty in the data. By using this FLS system, the researchers accomplished an improvement over the Interval Type-1 FLS. The RMSE found was in the order of 0.170.

Kim [68] demonstrates an implementation of an MLFFN network which produces a provisional forecast which is later adjusted for weather and holiday behavior by a fuzzy logic algorithm to produce a final forecast. The MAPE achieved this model is 1.3%. The MLFFN utilizes load data only. The training algorithm is backpropagation.

Mahmoud [69] presents a post-processing module that takes the output from traditional forecasting module and using some of the inputs to the models adjusts their forecast output by means of a fuzzy logic algorithm. The idea is to make a forecasting system more robust to operational scenarios not reflected in the training data. The goal is to increase forecasting intelligence in order to optimize parameter selection and cover any missing knowledge in the model. The fuzzy algorithm optimizes the error by means of converging into an acceptable range. The system works similarly to a PI controller in which two gains need to be tuned in order to achieve the optimum error range. The fuzzy logic mechanism finds the optimal K_p and K_i. The technique is applied on SVM, MLFFN and FBTFS with improvement in the forecast accuracy of 4.9%, 4.2% and 4.6% respectively.

Santos [70] and Rafael [71] developed a model that uses a neuro-fuzzy approach utilizing Gaussian membership functions to group signals into specific days based on fuzzy logic. The fuzzy logic parameters were adjusted by using backpropagation. The number of parameters for the model were determined using trial and error. The result is a 20-member function model that yields a forecast accuracy of 3.64%.

Srinivasan [72] proposes a Genetic Algorithm (GA) to deal with some shortcomings of the ANNs: dependence on initial parameters, long training time, lack of problem-independent way to choose appropriate network topology, and incomprehensive (black box) nature of ANNs. The GA is used to evolve a MLFFN and connecting weights in order to improve its forecasting accuracy. One network is developed for each day of the week, and Mondays are lumped together with days after holidays. The achieved MAPE of the evolved ANN ranges between 0.8% and 1.01%, compared with the MAPE of the comparable statistical model used by the same utility of 1.28% to 1.89% respectively.

Subbaraj [73] presents an approach using two modules: Evolutionary Programming (EP) and Particle Swarm Optimization PSO. The modules do linear combinations between the results of different MLFFN networks: instead of selecting the best fit ANN, it selects an optimal combination of ANNs to produce the forecast. The input consists of current and time-delayed values for load, temperature, relative humidity, and forecast values of temperature and relative humidity for the forecast period. The EP algorithm searches for and finds the optimal solution by evolving a population of candidate solutions over a number of iterations. The PSO algorithm is a stochastic global optimization technique in which all the solutions tend to follow an optimal. The technique results are as follows: the unprocessed best ANN result by a MLFFN gives a MAPE of 2.95%. By using EP, the solution improves to a MAPE of 2.24%; by using PSO, the solution improves to a MAPE of 2.27%. Therefore, EP gives a better forecast error.

Sun [74] describes a method that uses two different models for different days: a fuzzy logic (FL) support vector (SVM) method to forecast low load days, such as weekends, and Mondays, and a linear extrapolation method for the rest of the weekdays. The linear extrapolation method is adapted to include weather variables. The method reduces the MAPE from 2.32% to 1.63%

Yang [75] proposes a forecasting scheme where the input is partitioned by a FL algorithm into different groups (fuzzy sets), and an ARMAX forecast is done on each one of the groups. The input variables are pre-filtered based on their correlation with the system load. The ANN used for comparison, a MLFFN used 6 inputs, 10 hidden layer neurons and one output neuron. The result is a model with a MAPE of 1.98% compared with 2.31% for MLFFN and 2.22% for ARMAX run in SAS.

Carpinteiro [76] presents a model with two SOM networks one on top of each other. Seven input are used in the representation. 5 loads, and a sine/cosine function for the hour on a 24-hour period. The comparing ANN is a MLFFN with univariate input of hourly loads (2160 instances). The SOM networks utilize load feedback in between the two networks. The results of the SOM networks show a

MAPE o 2.33% vs. 2.64% for the MLFFN for a Friday, and 2.03% vs. 5.92% for a Sunday.

Crone [77] proposes an empirical comparison between MLFFN and SVR models using radial basis function (RBF) and linear kernel functions, by analyzing their forecasting power on five time series. Their results show that RBF SVR models have problems in extrapolating trends, while MLFFN and linear SVR models without data preprocessing provide robust accuracy across all patterns and clearly outperform the commonly used RBF SVR on trended time series. The MAPE results are: 1.391% for MLFFN, 1.632% for linear SVR, and 1.590% for RBF SVR.

ANNs are used in combination with other techniques in order to address specific deficiencies. Ghayekhloo [24] uses a hybrid time-series and regression analysis to select the best sets on inputs, and then a genetic algorithm to process the weights. Satish [1] uses a multi-ANN method to address the day-of-week problem which affects most statistical methods. Buitrago [25] and also Abdulaal et al. [26] propose a framework combining parameter estimation, clustering and ANNs to group load patterns. Lopez [27] uses self-organizing maps (SOM) for addressing the groupings of load patterns. Liang [28] uses differential dynamic programming together with ANNs trained with supervised and unsupervised learning algorithms. Zhang [17] uses wavelet decomposition and an adaptive ANN model for pricing purposes. The motivation of this research is that no attempt has been found in the references to model short-term loads using non-linear autoregressive ANNs. A different implementation using a feedforward neural network can be found in [42, 43].

2.5 Improving Robustness and Accuracy of NARX Networks

The present work combines techniques from different sources. Bayesian regularization, exponential weight decay, back propagation training and NARX neural networks, among others. We will break down the literature review into these different topics in order to give some order.

2.5.1 NARX Neural Networks

Feedforward neural networks have been widely studied in their ability to learn nonlinear behavior from a data set and be able to predict an output when given an input outside the range of the training range. Dynamic neural networks utilize time-lagged inputs in order to improve their feature retention abilities. Recurrent neural networks, on the other hand, use their own output as an input in order to improve their accuracy. NARX networks are a class of dynamic recurrent networks which in addition use lagged outputs. The inputs are both external and fed back outputs.

Much work has been done in forecasting time-series with artificial neural networks [78]. The work by Seidl [79] showed that recurrent neural networks are capable of accurately representing nonlinear dynamic systems, and Siegelmann [80] showed that they are computationally powerful. Lin [81] shows in his research that recurrent neural networks using back propagation training may not be able to accurately retain long-term dependencies, i.e. when the information needed to calculate the forecast is highly dependent on information given to the network in training a long time ago. However, the author goes to prove that NARX networks are actually much better at forecasting long-time dependencies than conventional recurrent networks because the feedback connections propagate the gradient in a more efficient manner, making NARX networks the better choice for time-series forecasting.

The simulations on the present work are based on previous work on NARX neural networks by Buitrago *et al* for the purpose of forecasting short-term loads (24-hour horizon) on electric power grids [82, 83]. The resulting forecasting accuracy was about 1% (MAPE). The benefit of using a NARX network is that for nonlinear time-series data NARX has a better ability to generalize periodic behavior because of the memory effect of the time-lagged inputs associated with dynamic networks, in addition to the long-term effects stored in the recurrent network by using output feedback. Recurrent networks, called infinite impulse response (IIR) networks, carry out the response to an impulse asymptotically and therefore the effects of every input are reflected in the output. The work showed the effectiveness of the NARX network in forecasting 24 hours ahead when compared with statistical (ARMAX), feedforward neural networks, and state space methods. All the detailed descriptions of the network used are in the cited work.

2.5.2 Back Propagation Training

Back propagation training is the most widely used training method for neural networks. It evolved from the steepest descent method in which a solution is sought by minimizing an objective function (error) in a space by seeking the largest error solution in the space. Many adjustments have been made because of the tendency of many methods to get stuck in local minima, or the lengthy computations. Methods that use the gradient, Jacobian or Hessian are used as explained by the seminal work by Rumelhart. The most well-known backpropagation method in use is the Levenberg-Marquadt method, which interpolates between the Gauss-

Newton method and the gradient descent method in order to find the optimum next step. Since the proposed research uses back propagation training, we just mention those papers that are relevant to the current work.

Marquadt [84] presented an interpolation approach in 1963. The minimum error was sought by interpolating between a Taylor series expansion (Gauss-Newton) and a gradient descent method. The idea was to use the neighborhood in which the Taylor solution approximates the data in a reasonably good manner. This was the basis for the Levenberg-Marquadt method which uses the same approach.

Hagan [85] provided a comprehensive methodology for the application of the Levengerg-Marquadt method to feedforward networks. Hinton [86] shows how the gradients can be calculated with simple linear algebra for both the forward and back propagations.

Jacobs [87] proposes modifying the learning rate so that each weight in the network has its own weight. This rates are not supposed to remain constant either, but they should be varied in time. The learning adaptation proposed is base on heuristics. First, every parameter of the performance measure should have its own individual learning rate. Second, every learning rate should be allowed to vary over time. Third, the learning rate for a parameter must be increased if the derivative of the adjustment is the same sign for several consecutive time steps; and fourth, when the sign alternates, for several consecutive steps, the learning rate must be increased.

Karnin [88] describes a method for obtaining an adequate sized neural network by starting with a large net and pruning the synapses that do not contribute to the solution. The author describes a method for determining the sensitivity of the error function to the inclusion or exclusion of a synapse, thus avoiding having to calculate all the permutations of synapses that would be required. An array is created every time that the backpropagation algorithm makes a pass, and the incremental changes to the synaptic weights are recorded in the array. Then, starting by the most sensitive synapse, the weights are ordered and the last items in the array are discarded. This method alleviates the computational burden of large networks.

Moriarty [89] presents a survey of the temporal difference methods for solving reinforcement learning problems. Evolutionary algorithms are used for the solution and their strengths and weaknesses are presented and analyzed.

Ho [90] presents an approach to short-term load forecasting using adaptive learning. By using adaptive learning, the convergence of the solution is much faster. The method is tested using Taiwan's electric load. The adaptation rule proposed is changing the learning rate by 1% up or down according to whether the error is increasing or decreasing respectively.

Chawla [91] addresses the topi of semi-supervised learning but focuses on the effect of increasing or decreasing the amount of labeled versus unlabeled data. The analysis includes the differentiation made by the author between the original distributions of the labeled and unlabeled data: both types of data may come from different distributions and thus may have different impacts on the solution.

McKay [92] described a quantitative and practical Bayesian framework for learning of mappings in feedforward networks. The framework includes an Occam's razor technique, penalizing the most complex solutions in order to find the solution with the best match to the problem. When that solution is found, the correlation between generalization ability and the Bayesian evidence is usually high.

2.5.3 Bayesian Regularization

Regularization techniques have been applied to neural networks with great success. Here is a list of the relevant work as it applies to the current research.

Burden [93] states that Bayesian-regularized ANNs (BRANNs) are more robust than standard back propagation nets, and can reduce the need for cross-validation. The cross-validation process scales as $O(N^2)$, and therefore it is one of the most computational intensive parts of using neural networks. BRANNs provide other advantages including lower overfitting probability, and simpler networks.

Halpern [94] defines a measure of algebraic conditional plausability represented using Bayesian networks. Uncertainty is represented using plausability measures. The paper also shows the conditions under which conditional plausability must be defined. The notion applies to both quantitative and qualitative Bayesian networks.

Sanghai [95] introduces the concept of relational dynamic Bayesian networks, which are an extension of Bayesian networks to first-order logic. They are used for developing accurate fault diagnosis in factories, which are processes that have discrete variables with very large domains. The methods use by the authors are tested and show to outperform standard particle filtering.

MacKay [96] associates Bayesian regularization with exponential weight decay in the sense that "learning models that are well matched to the problem under study show a good correlation between generalization ability and the Bayesian evidence obtained".

Moody [97] introduces the Generalized Prediction Error (GPE) estimate of generalization performance. The GPE index is based on the same author's work [98] on the effective number of parameters in order to determine an optimal structure for the neural network. The idea is to balance the tendency of larger networks to overfit data with the value of the learning rate thus optimizing the generalization performance of the net. The author provides a method for calculating GPE in the case of quadratic weight decay.

2.5.4 Error Weight Decay Method

All the research found in error weight decay focus on ensuring that synaptic weights that are relevant are able to increase, but those whose values are small and thus have little influence on the generalization ability of the network are forced to decay, effectively pruning the network. These are the most relevant papers on the subject.

Jacobs [87] introduces the concept of momentum. It introduces a new term to the weight update equation, a momentum factor, that determines the relative contribution of the current and past partial derivatives to the current weight change. In its proposed solution, this weight is the exponentially weighted sum of the weight's current and past partial derivatives. The exponential momentum factor in effect implements the heuristics explained above and allow changes in the learning rate.

Orr [99] presents a series of regularization techniques to improve generalization, including weight decay. It provides some rules for the selection of the optimal decay parameter, and the early stopping technique which requires a partition of the training set for validation, and measuring the performance of the training set versus the validation set in order to apply a heuristic stopping criterion. In their book, a chapter written by Rognvaldsson [100] describes a technique for calculating the optimal weight decay parameter. It consists on computing the gradient at the early stopping solution and dividing it by its norm, a heuristic form to ensure better performance. The accuracy of the weight decay parameter obtained is good according to the author.

Dillon [101] addresses the question on improving convergence, and proposes the solution to include a momentum term to be added to the back propagation rule. This momentum has the effect of filtering out high frequency variations in the error, and the author proposes a value of 0.9 for the decay constant.

Hanson [102] proposes an analysis of the Rumelhart method for selecting minimal representation during learning in back propagation networks. The author cites Rumelhart ”The simplest, most robust network that accounts for a data set will, on average, lead to the best generalization to the population from which the training set has been drawn”. The approach adds penalty terms the error function in order to constrain the search and cause weights to differentially decay. The end result is a weight decay in which the larger weights are kept and the smaller weights tend to go to zero. This is an effective way of pruning the network.

Treadgold [103] proposes combining a global optimization algorithm (simulated annealing) with gradient descent back propagation training. The idea is to introduce noise in order to stimulate removal from local minima while applying gradient descent.

Krogh [104] introduces the idea of introducing weight decay to limit the growth of weights. The method prevents the growth of the synaptic weights unless it is absolutely necessary. There is a penalization factor associated with the weights and this is the factor that the other authors cited above focus on finding.

As shown by the existing research presented above, the focus on weight decay is on synaptic weight, regardless of the time dimension. The proposed research concentrates in weight decay in the time sense, i.e. assign more weight to recent data and less weigh to older data. By assigning a higher weight to the errors in recent data, the rationale is that training will take into consideration a higher adjustment for newer data, and will tend to ignore or assign less value to the response to errors in older data. No papers were found on the proposed technique.

Chapter 3

Statistical Approach to Forecasting

The proposed research claims to improve the accuracy of the 24-hour energy forecast by using a NARX artificial neural network. Statistical methods are used for comparison, and therefore an explanation of what those methods are is necessary for clarity and completeness. This chapter will also present the descriptive statistics of the data set selected with the purpose of understanding the characteristics of the data and how traditional methods have addressed them.

3.1 Data Description

In order to understand the types of statistical features that the selected data set includes, we must first go into some detail of what it includes and the types of variations and time dependencies encountered.

Hourly real-time system demand load data was obtained from the ISO-NE grid operator for New England from 2005 to 2015 [105]. Figure 3.1 shows the average hourly load for the New England region in GW. Notice the sinusoidal shape of the load. At 4 am, for example, the load reaches a minimum, and at 7 pm it reaches a maximum.

Looking at the data in a long-range manner, we can see the seasonal variations

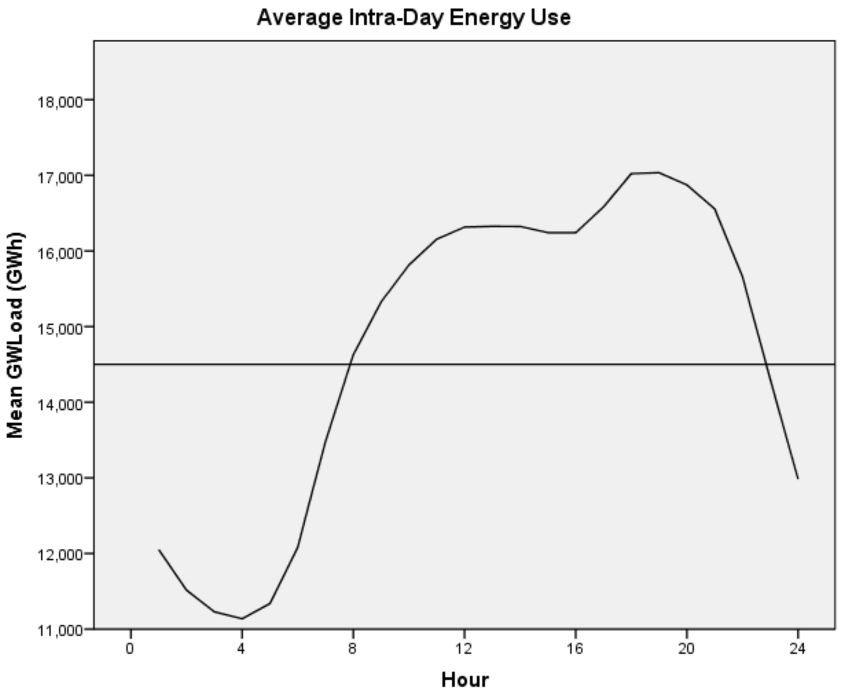


Figure 3.1: Average Hourly Load

in the load which are clear in Figure 3.2. The graph clearly shows winter peak loads (high peaks) and summer peak loads (smaller peaks). This data is noisy, so if we use only the hourly average load, we see the data in Figure 3.3.

By filtering even further, we obtain the average daily load for 1 year, which is more clear, as shown in Figure 3.4. Here we see the fluctuations from one week to another: every week is distinct, with its own peak, which occurs Tuesday through Friday, and lows for Friday to Sunday.

The statistical characteristics of the load data are shown in the box plots of Fig.3.5. Notice the seasonal characteristic of the load, i.e. variability month to month; The particular reduction in the load on weekends, and the variation in the load according to the hour of the day.

But now we need to focus on the predictors. What types of variables allow us

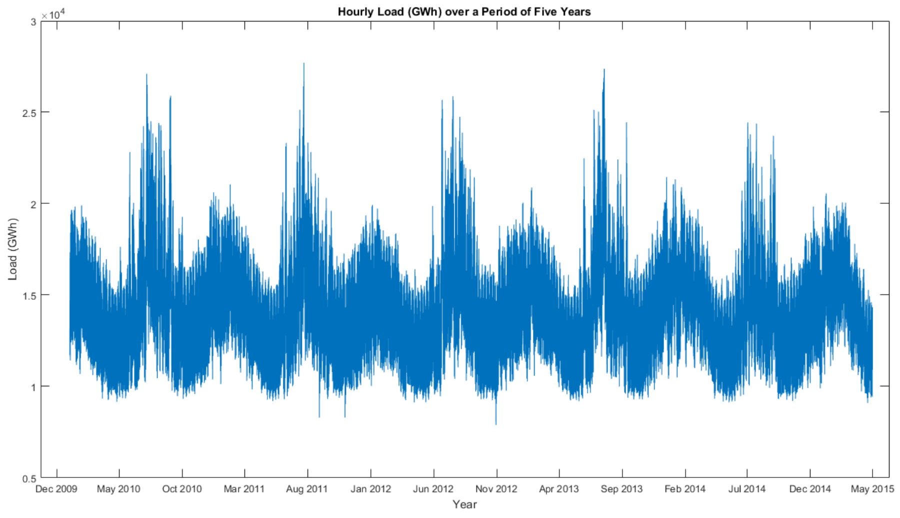


Figure 3.2: 5-Year Hourly Load

to make accurate forecast on the data. Figure 3.6 show the scatter plots for Dew Point vs. Load and Dry Bulb Temperature vs. Load. We can clearly see that when the temperature is very low the energy consumption increases, due to use of electric heating, and when the temperature is very high the energy use is also very high due to the use of air conditioning. The ”comfort zone” is around 70 degrees fahrenheit, when users do not utilize air conditioning or heating.

Other variables where correlations can be found with the total load are seasonal variables: time of the year (winter, spring, summer, fall), which can be described by the variable month. Day of the week, because the load varies throughout the week depending on the day, which can be described by the numbers 1 (Sunday) through 7 (Saturday). Hour of the day, because the load also varies depending on what time it is: late nights are usually a minimum, when everyone sleeps and there is very little use of appliances and lighting. We describe the hour of the day from 1 to 24. Also whether a day is a working day has an impact on the energy use: holidays tend to have a noticeable lower energy consumption than working days.

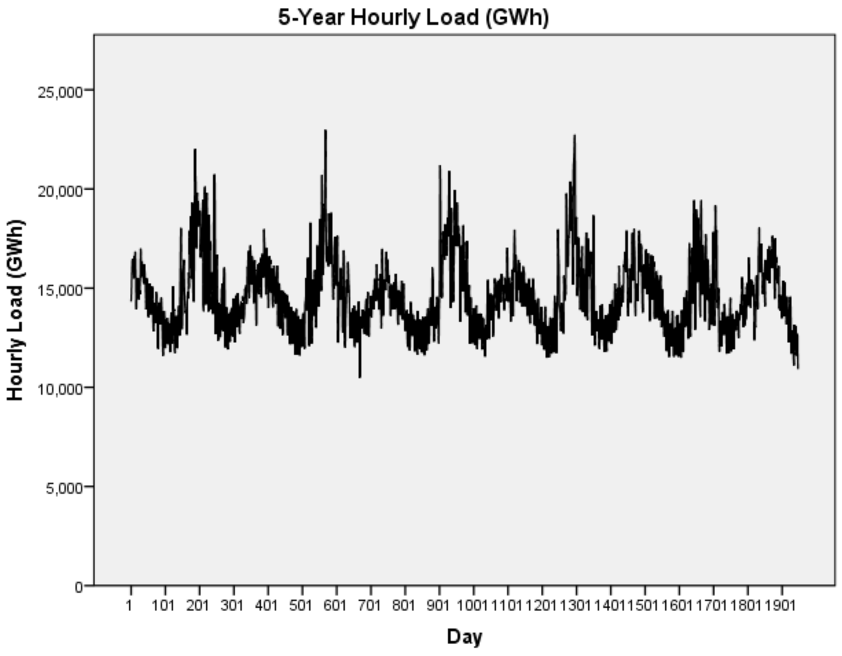


Figure 3.3: 5-Year Average Hourly Load

We describe this with a 0 (holiday or weekend) or a 1 (working day, non-holiday).

A correlation analysis was carried out to determine the level of correlation between the predictors and the load. Figure 3.7 shows the SPSS output for the correlations between the predictors discussed and the hourly load. The results show that the Pearson correlation of all variables is significant at the 90% confidence level. This means that all the variables have an important incidence in the load, as expected by the qualitative analysis of the box plots.

Controlling for year, month, day and hour, Figure 3.8 shows the correlations between the other non-time variables, which results in highly correlated predictors (dry bulb temperature, dew point temperature, working day boolean and day of the week) with the output (load). In a separate analysis, not shown, the relative humidity and wet bulb temperature) resulted highly correlated with the dry bulb temperature and the dew point temperature, and therefore were left out of the

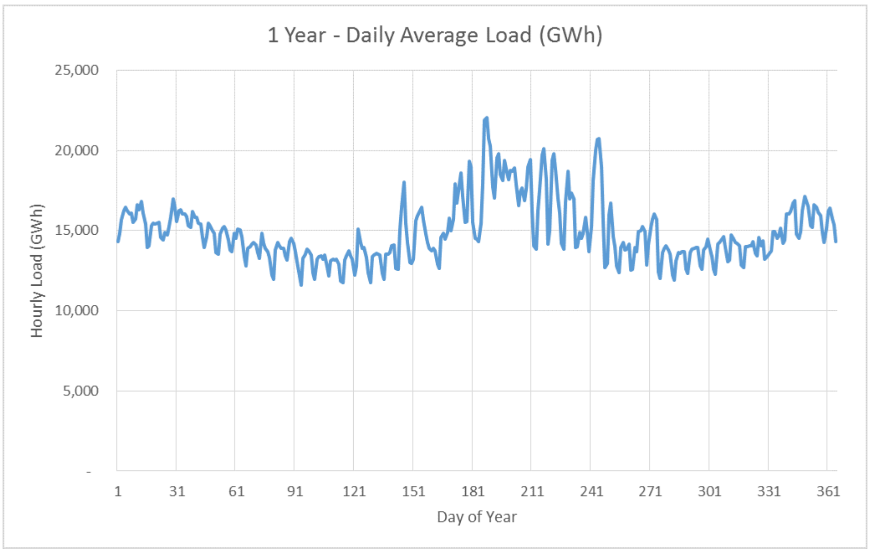


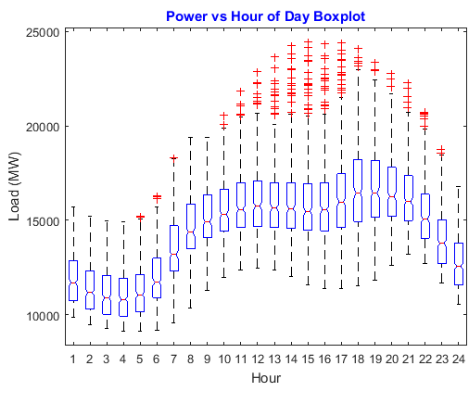
Figure 3.4: 1-Year Average Daily Load

predictor set.

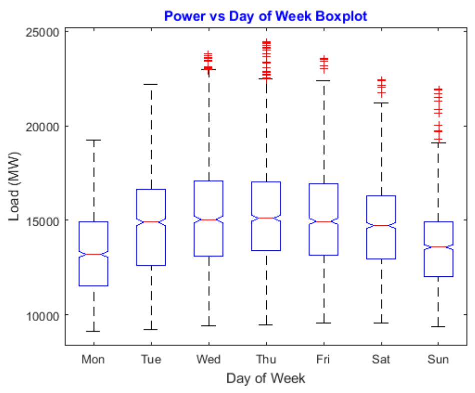
Studying the autocorrelations we can understant to what level previous values of the variables affect future values. We look first at the serial correlation of the stochastic series to see if there are patterns behind the data. Figure 3.9 shows the decaying pattern of the serial correlation, and it demonstrates that current values depend on the values of the most recent variables and decay as the lag increases. Also, there is a cyclical pattern which repeats at 24-hour intervals, as expected. The sample partial autocorrelation does not exhibit a white noise distribution (normal distribution centered at zero) which means that extensive modeling is required to pick up all the harmonics in the signal.

3.1.1 Signal Decomposition

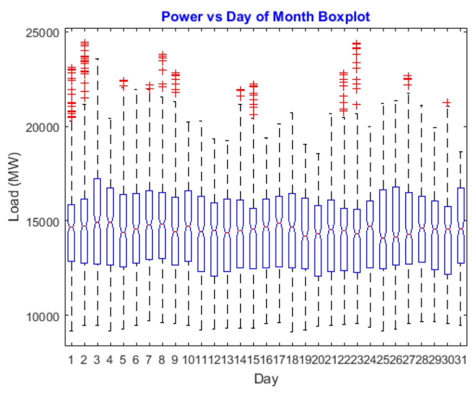
In order to understand the different components of the load signal, we use the signal decomposition method. The method is based on trying to decompose the signal into mathematical models. Many approaches are available, starting from a Taylor series, which fits a polynomial to the series, a linear regression, or frequency



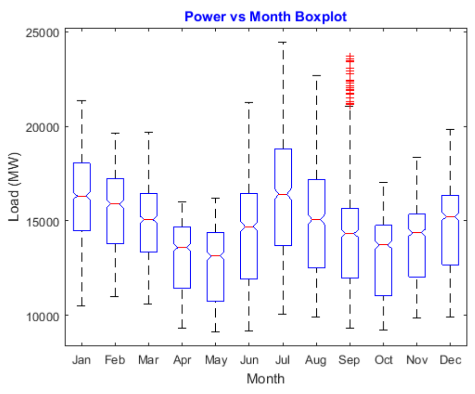
(a) Hour of Day



(b) Day of Week



(c) Day of the Month



(d) Month of the Year

Figure 3.5: Statistical Characteristics of Electrical Load Data

methods. We will study them separately with the ultimate goal of finding different components that can be modeled into a linear combination.

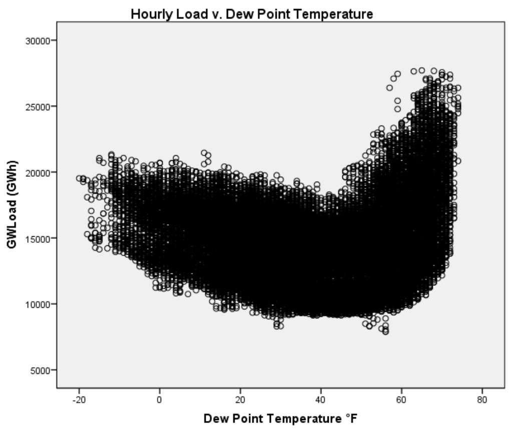
Taylor Series If $f : \vec{R} \rightarrow \vec{R}$ is a function n times differentiable at point $a \in \vec{R}$ and $n \geq 1$ an integer, then there exists a function $h_n : \vec{R} \rightarrow \vec{R}$ such that

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^n(a)}{n!}(x-a)^n + h_n(x)(x-a)^n \tag{3.1}$$

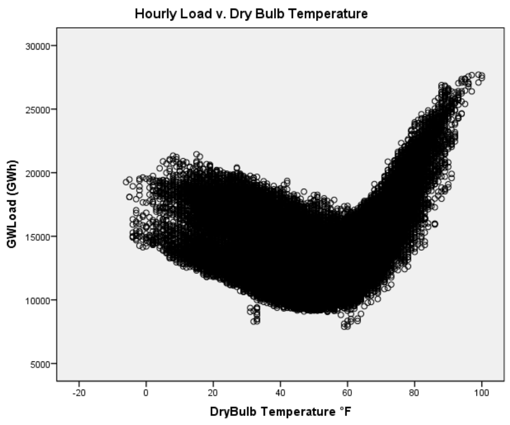
and

$$\lim_{x \rightarrow a} h_n(x) = 0$$

Taylor series allows us to calculate the coefficients of the approximation poly-



(a) Dew Point



(b) Dry Bulb

Figure 3.6: Load vs. Dew Point and Dry Bulb Temperatures

nomial.

Fourier Series For periodic functions, a Fourier series consists of an expansion of a function $f : \vec{R} \rightarrow \vec{R}$ which may or may not be differentiable at all points in $\vec{R}$ into an infinite sum its harmonic components [106] such that

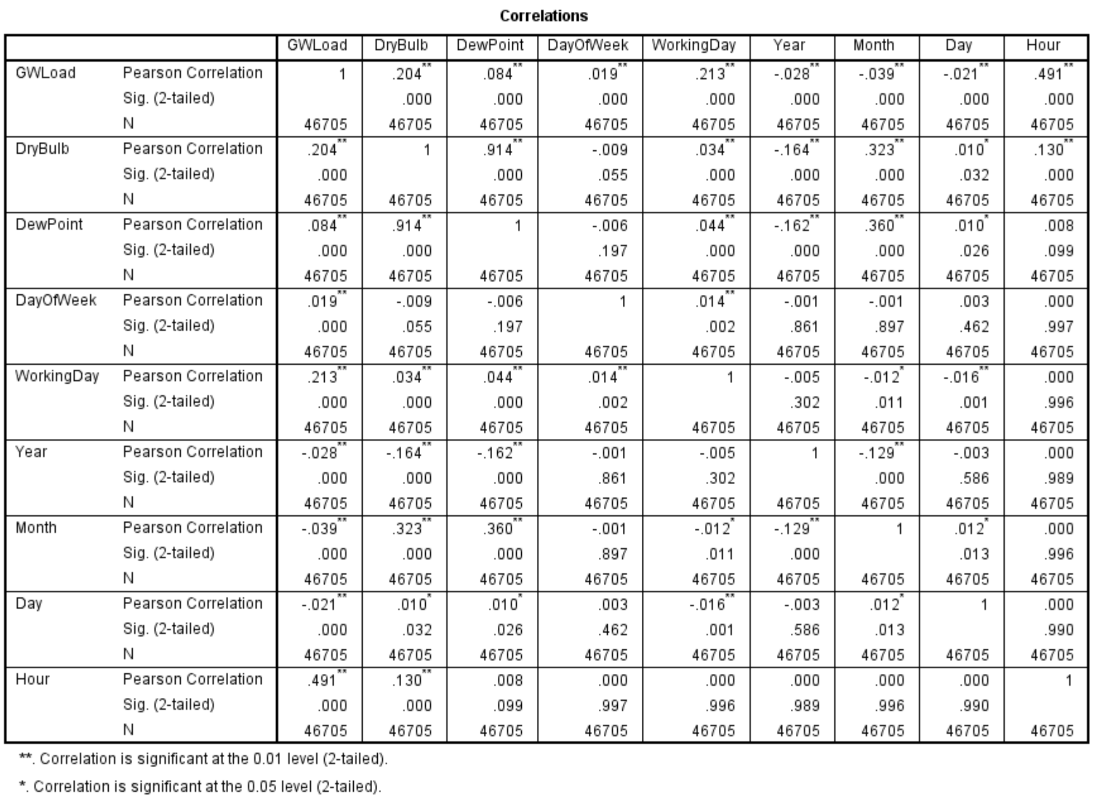
$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx) \tag{3.2}$$

where

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$
$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)cos(nx)dx$$
$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)sin(nx)dx$$

Fourier series allows us to model periodic functions by decomposing their signals into frequencies.

Noise signals can be included in an aggregate analysis by taking into consideration their statistical distribution nature. All the methods above allow us to analyze signals within a boundary of different mathematical elements: constants,



Correlations

		GWLoad	DryBulb	DewPoint	DayOfWeek	WorkingDay	Year	Month	Day	Hour
GWLoad	Pearson Correlation	1	.204**	.084**	.019**	.213**	-.028**	-.039**	-.021**	.491**
	Sig. (2-tailed)		.000	.000	.000	.000	.000	.000	.000	.000
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
DryBulb	Pearson Correlation	.204**	1	.914**	-.009	.034**	-.164**	.323**	.010*	.130**
	Sig. (2-tailed)	.000		.000	.055	.000	.000	.000	.032	.000
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
DewPoint	Pearson Correlation	.084**	.914**	1	-.006	.044**	-.162**	.360**	.010*	.008
	Sig. (2-tailed)	.000	.000		.197	.000	.000	.000	.026	.099
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
DayOfWeek	Pearson Correlation	.019**	-.009	-.006	1	.014**	-.001	-.001	.003	.000
	Sig. (2-tailed)	.000	.055	.197		.002	.861	.897	.462	.997
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
WorkingDay	Pearson Correlation	.213**	.034**	.044**	.014**	1	-.005	-.012*	-.016**	.000
	Sig. (2-tailed)	.000	.000	.000	.002		.302	.011	.001	.996
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
Year	Pearson Correlation	-.028**	-.164**	-.162**	-.001	-.005	1	-.129**	-.003	.000
	Sig. (2-tailed)	.000	.000	.000	.861	.302		.000	.586	.989
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
Month	Pearson Correlation	-.039**	.323**	.360**	-.001	-.012*	-.129**	1	.012*	.000
	Sig. (2-tailed)	.000	.000	.000	.897	.011	.000		.013	.996
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
Day	Pearson Correlation	-.021**	.010*	.010*	.003	-.016**	-.003	.012*	1	.000
	Sig. (2-tailed)	.000	.032	.026	.462	.001	.586	.013		.990
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705
Hour	Pearson Correlation	.491**	.130**	.008	.000	.000	.000	.000	.000	1
	Sig. (2-tailed)	.000	.000	.099	.997	.996	.989	.996	.990	
	N	46705	46705	46705	46705	46705	46705	46705	46705	46705

**. Correlation is significant at the 0.01 level (2-tailed).

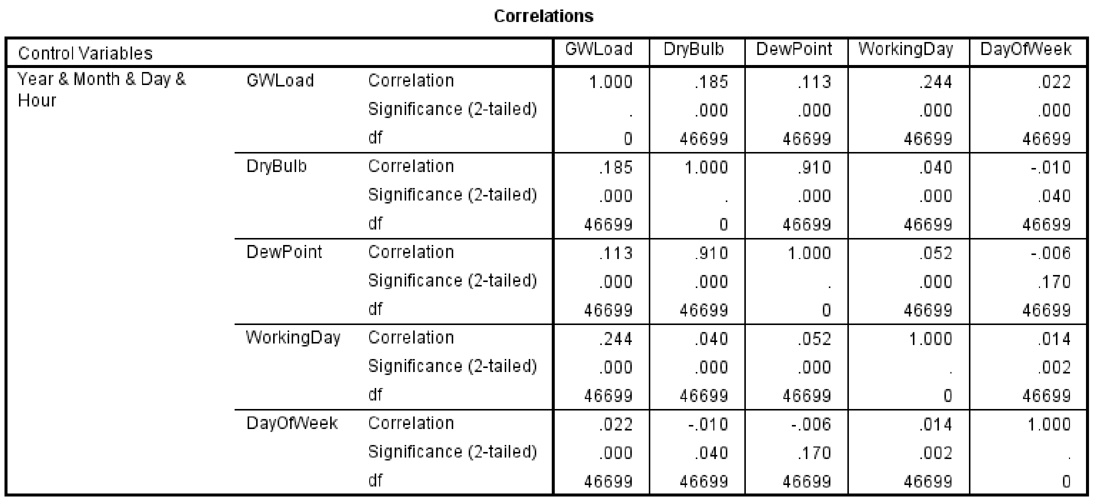
*. Correlation is significant at the 0.05 level (2-tailed).

Figure 3.7: Correlations Table - SPSS Output

straight lines, exponential trends, polynomial trends, sinusoidal frequencies, etc. But in all of them the component must be assigned *a priori* to one of the models. Here we will see how Artificial Neural Networks do not need an *a priori* knowledge of the underlying mathematical relationship between input and output.

3.1.2 Fourier Transform

A first approach at decomposing signal decomposition would be to model the time series as the linear combination of a constant value, a trend value (linear, quadratic, or exponential), and a series of sinusoidals with different frequencies and phase shifts. This attempt was carried out in order to find the prevailing frequencies of the model. Two additional steps were required: first, data was transformed in order to make the model stationary (the mean does not change



Correlations

Control Variables			GWLoad	DryBulb	DewPoint	WorkingDay	DayOfWeek
Year & Month & Day & Hour	GWLoad	Correlation	1.000	.185	.113	.244	.022
		Significance (2-tailed)	.	.000	.000	.000	.000
		df	0	46699	46699	46699	46699
	DryBulb	Correlation	.185	1.000	.910	.040	-.010
		Significance (2-tailed)	.000	.	.000	.000	.040
		df	46699	0	46699	46699	46699
	DewPoint	Correlation	.113	.910	1.000	.052	-.006
		Significance (2-tailed)	.000	.000	.	.000	.170
		df	46699	46699	0	46699	46699
	WorkingDay	Correlation	.244	.040	.052	1.000	.014
		Significance (2-tailed)	.000	.000	.000	.	.002
		df	46699	46699	46699	0	46699
	DayOfWeek	Correlation	.022	-.010	-.006	.014	1.000
		Significance (2-tailed)	.000	.040	.170	.002	.
		df	46699	46699	46699	46699	0

Figure 3.8: Correlations Table - SPSS Output - Controlling for Time Variables

with time): instead of processing the raw data, the model processes the differences between subsequent steps in data. This completely removes the variability in time, and therefore a stationary model is obtained. Second, in order to make the model stationary, the long term trend must be eliminated. A linear trend was found to be more fitting, and its value was subtracted from the data in order to make the time series stationary. Now the data series is *clean* and we can proceed with the frequency analysis.

In order to determine the dominant frequencies, we can compute the discrete Fourier Transform (DFT). The DFT transforms a finite series of equally-spaced values of a function into a same length series of equally-spaced values of the discrete time Fourier transform which is a function of frequency in the complex space. We do this calculation by using the Fast Fourier Transform (FFT) technique. If $x_0 \dots x_n$ are complex numbers, the DFT is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-i2\pi kn}{N}} \tag{3.3}$$

In order to compute this directly we would have an $O(N^2)$ problem, but the

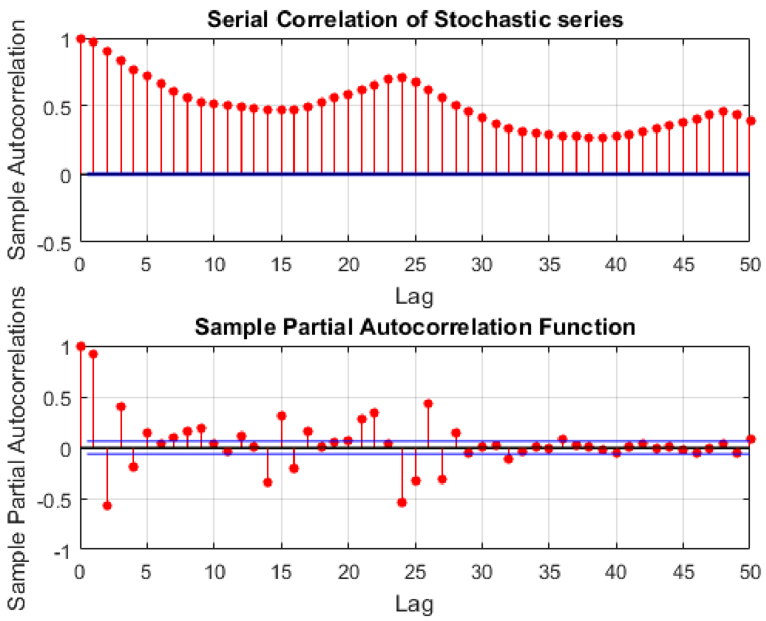


Figure 3.9: Serial Autocorrelation

FFT algorithm can compute this using a complexity of $O(NlogN)$ which is much lower. By using FFT we calculated the plot in Figure 3.10, and obtained the dominant frequencies: 365 days, 120 days, 7 days and 24 hours.

The total combined model gives us a series of residuals that are plotted in Figure 3.11. We still don't get a normal distribution of the residuals centered at zero. This shows the limitation of the technique. Even though all the steps of the statistical model have been followed:

- Apply a transformation (in our case differences) on the data in order to make the model stationary
- Remove the trend to ensure model stationarity
- Analyze autocorrelation graph in order to determine if moving average is an appropiate technique

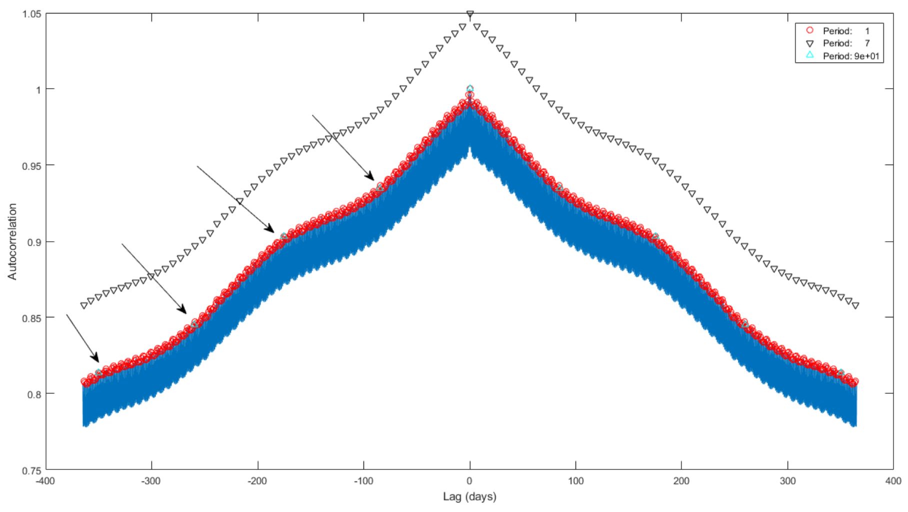


Figure 3.10: Signal Energy Spectrum

- Analyze partial autocorrelation graph in order to determine if autoregressive modeling is appropriate
- Look at extended autocorrelation chart of the data to see if an ARMA combination is needed
- Evaluate Akaike's Information Criterion (AIC) on a set of different ARMA models with different parameters in order to investigate the models with the lowest AIC value.
- Evaluate the Schwartz Bayesian Information Criterion (BIC) and study the model with the lowest BIC values.

After applying the above steps, the statistical model selected should yield residuals that meet the following conditions:

- Normally distributed
- Zero mean

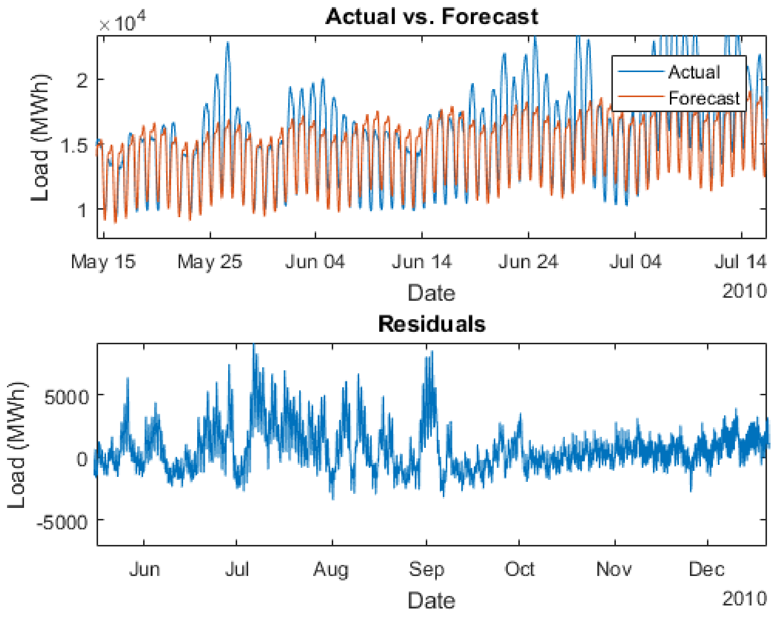


Figure 3.11: Residual Plot

- No autocorrelation

In the next section an ARIMA model will be fitted and the results analyzed.

3.2 Autoregression and Moving Average Considerations

Load forecasting methods can be classified into two main categories: statistical methods and computational intelligence (CI) methods. Statistical methods include all forms of autoregressive and parametric models. Regressive models are "basically linear devices that attempt to model distinctly nonlinear relationships" [5]. When nonlinear regression is used, the relationship between the demand and the explanatory inputs is very complex and it is difficult to validate empirically. CI methods include Artificial Neural Networks (ANN), fuzzy logic, and expert systems. Typical load patterns are non-linear and therefore non-linear models have proven more effective in generating short-term forecasts. ANNs provide an accurate approach to the problem and have the advantage of not requiring the user to have a clear understanding of the underlying mathematical relationship between input and output. ANNs have proven to be effective forecasting techniques with most studies showing an improvement in forecast accuracy over their statistical counterparts. The resulting performance of different ANN methods has been analyzed by many authors [24]. The results of this work falls into the expected forecast accuracy range with advantages both in performance and simplicity.

Statistical methods have been the preferred means of developing a fit and forecasting data regression model for most applications. Due to the nature of the data, as seen in Section 1.2, its seasonal and cyclical behavior, a linear regression model cannot accurately describe the data. An SPSS and Matlab data framework was established in order to analyze the statistical nonlinear regression model which will be used as a benchmark for studying the performance of the proposed ANN model later in Chapter 6.

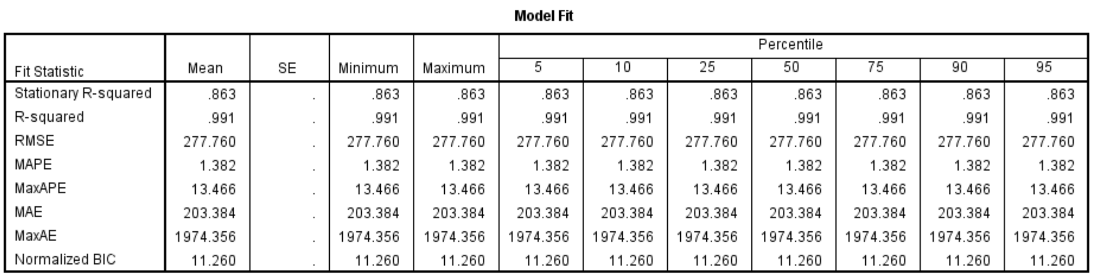
In this section an ARIMA model is developed and its performance analyzed. First, the prerequisites for ARIMA must be met and therefore some data transformations need to be performed *a priori*.

ARIMA models are the most general class of nonlinear models for time series forecasting. The main requirement for the data in order to apply an ARIMA model is that data be stationary, i.e. the characteristics of the data must not be dependent on which time frame is selected. This means that there must be no trends in the data, or predictable behavior on the residuals of the model. Different techniques for making the data stationary are used, the main ones of which are differencing, logging or deflating. The goal to accomplish with this transformation is to achieve a data set in which its autocorrelations remain constant over time. We must first identify the power spectrum of the data series in order to determine the natural frequencies present in the underlying data. We do this by running a fast Fourier transform on the data as explained in the previous section.

The results of the ARIMA methodology are presented in the next section.

3.3 ARIMA Results

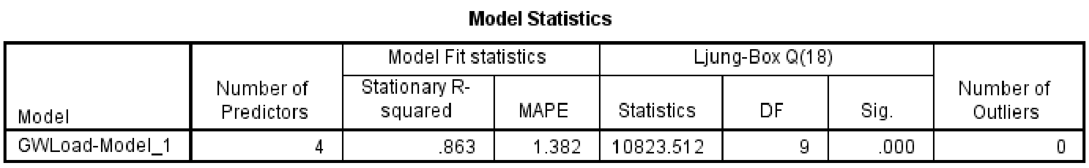
By using the model optimizer in SPSS, the ARIMA model was developed in SPSS and the results are shown in Figures 3.12 and 3.13. As shown in the result the fit is very good, and SPSS found the optimal parameters to be AR = 2, I = 1 and MA = 8. In Chapter 6 we will show that this fit does not necessarily translate into good forecasting performance.



Model Fit

					Percentile						
Fit Statistic	Mean	SE	Minimum	Maximum	5	10	25	50	75	90	95
Stationary R-squared	.863	.	.863	.863	.863	.863	.863	.863	.863	.863	.863
R-squared	.991	.	.991	.991	.991	.991	.991	.991	.991	.991	.991
RMSE	277.760	.	277.760	277.760	277.760	277.760	277.760	277.760	277.760	277.760	277.760
MAPE	1.382	.	1.382	1.382	1.382	1.382	1.382	1.382	1.382	1.382	1.382
MaxAPE	13.466	.	13.466	13.466	13.466	13.466	13.466	13.466	13.466	13.466	13.466
MAE	203.384	.	203.384	203.384	203.384	203.384	203.384	203.384	203.384	203.384	203.384
MaxAE	1974.356	.	1974.356	1974.356	1974.356	1974.356	1974.356	1974.356	1974.356	1974.356	1974.356
Normalized BIC	11.260	.	11.260	11.260	11.260	11.260	11.260	11.260	11.260	11.260	11.260

Figure 3.12: ARIMA(2,1,8) Model Fit Table - SPSS Output



Model Statistics

		Model Fit statistics		Ljung-Box Q(18)			
Model	Number of Predictors	Stationary R-squared	MAPE	Statistics	DF	Sig.	Number of Outliers
GWLoad-Model_1	4	.863	1.382	10823.512	9	.000	0

Figure 3.13: ARIMA(2,1,8) Model Statistics

Figures 3.14 and 3.15 show the autocorrelation plots and partial autocorrelation plots for the residuals. They show that there is still quite a lot of information left in the residuals, specially periodic data.

Figures 3.16 and 3.17 show the autocorrelation and partial autocorrelation tables where the peaks are shown in the expected lags.

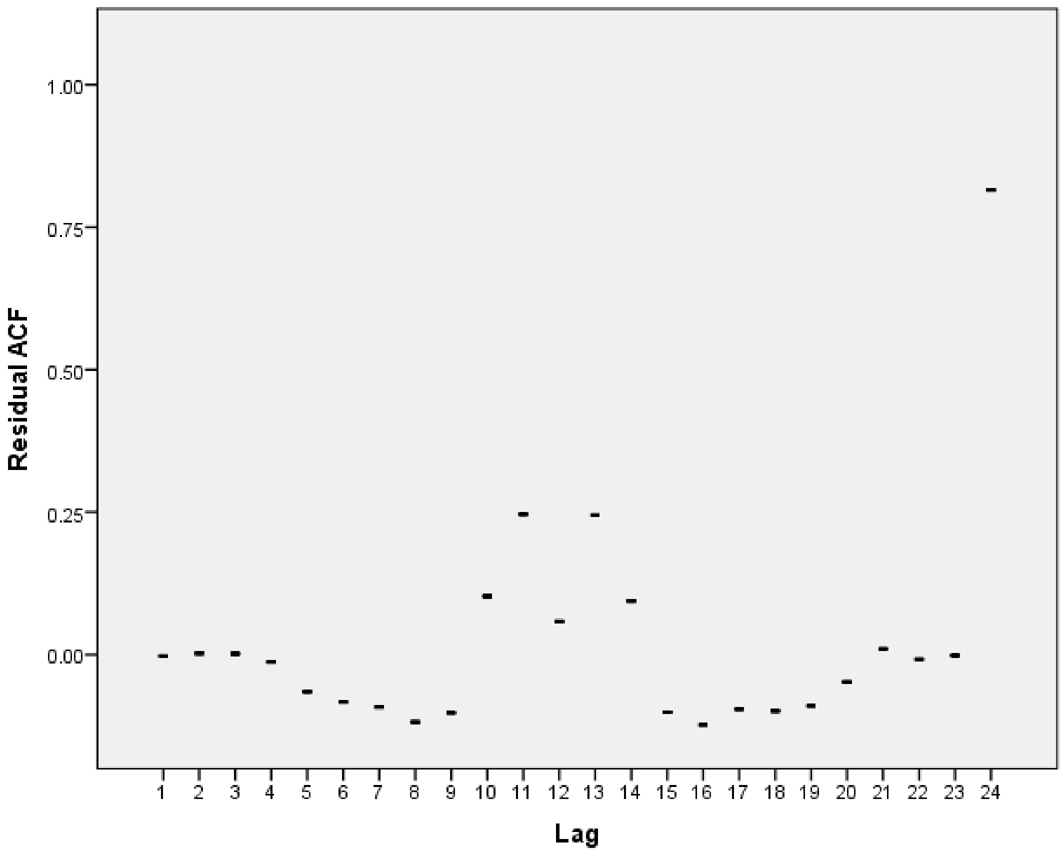


Figure 3.14: ARIMA(2,1,8) Residual Plot ACF Autocorrelation

Finally Figure 3.18 shows the lags at which the correlation and autocorrelation of the residuals show still data left.

The next chapter describes a proposed solution using neural networks to generate a 24-hour ahead forecast. As will be seen in Chapter 6, NARX ANNs address the shortcomings of the statistical approach.

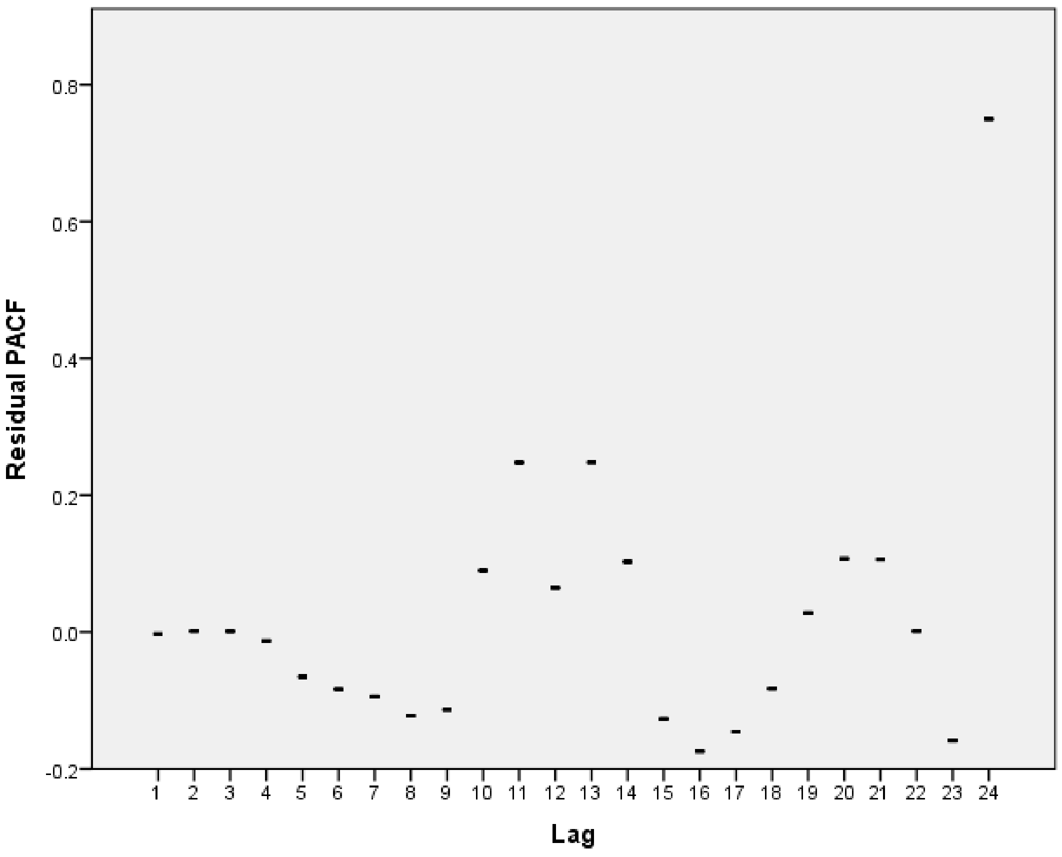
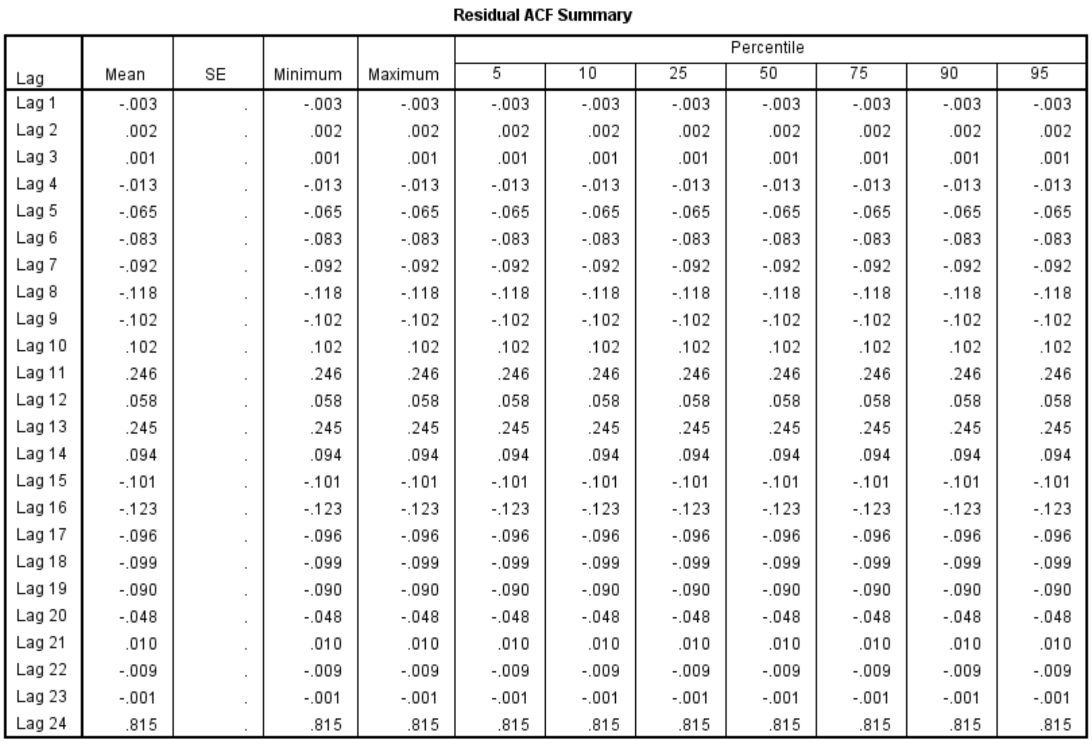
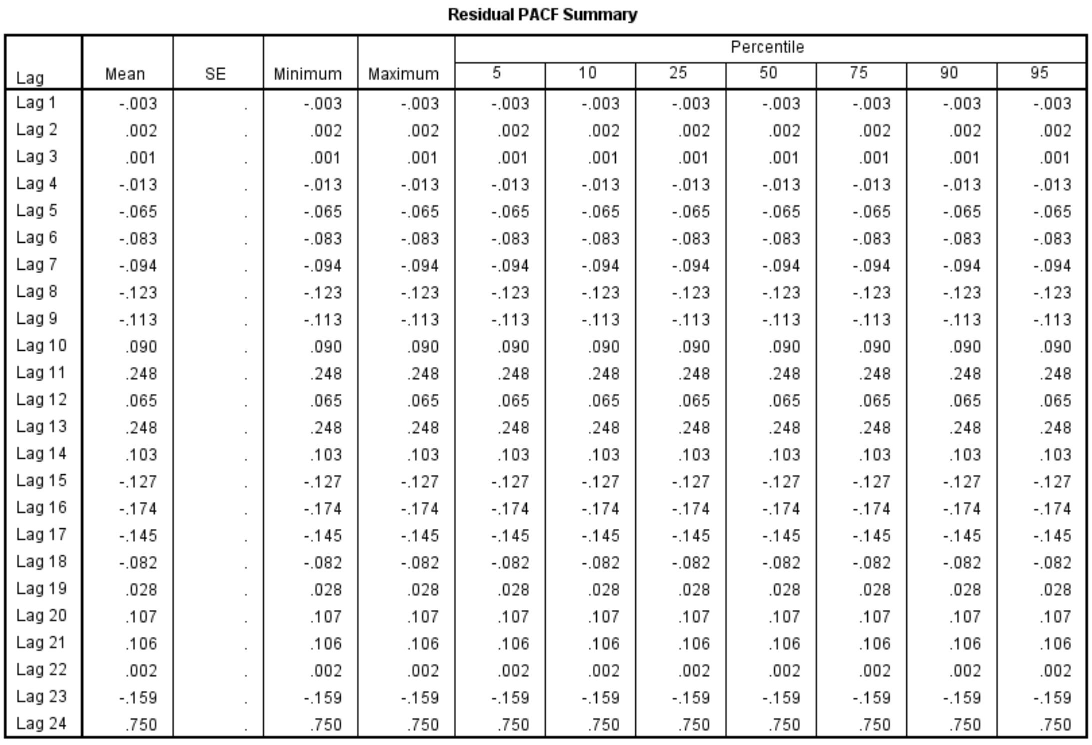


Figure 3.15: ARIMA(2,1,8) Residual Plot PACF Partial Autocorrelation


Residual ACF Summary

Lag	Mean	SE	Minimum	Maximum	Percentile						
					5	10	25	50	75	90	95
Lag 1	-.003	.	-.003	-.003	-.003	-.003	-.003	-.003	-.003	-.003	-.003
Lag 2	.002	.	.002	.002	.002	.002	.002	.002	.002	.002	.002
Lag 3	.001	.	.001	.001	.001	.001	.001	.001	.001	.001	.001
Lag 4	-.013	.	-.013	-.013	-.013	-.013	-.013	-.013	-.013	-.013	-.013
Lag 5	-.065	.	-.065	-.065	-.065	-.065	-.065	-.065	-.065	-.065	-.065
Lag 6	-.083	.	-.083	-.083	-.083	-.083	-.083	-.083	-.083	-.083	-.083
Lag 7	-.092	.	-.092	-.092	-.092	-.092	-.092	-.092	-.092	-.092	-.092
Lag 8	-.118	.	-.118	-.118	-.118	-.118	-.118	-.118	-.118	-.118	-.118
Lag 9	-.102	.	-.102	-.102	-.102	-.102	-.102	-.102	-.102	-.102	-.102
Lag 10	.102	.	.102	.102	.102	.102	.102	.102	.102	.102	.102
Lag 11	.246	.	.246	.246	.246	.246	.246	.246	.246	.246	.246
Lag 12	.058	.	.058	.058	.058	.058	.058	.058	.058	.058	.058
Lag 13	.245	.	.245	.245	.245	.245	.245	.245	.245	.245	.245
Lag 14	.094	.	.094	.094	.094	.094	.094	.094	.094	.094	.094
Lag 15	-.101	.	-.101	-.101	-.101	-.101	-.101	-.101	-.101	-.101	-.101
Lag 16	-.123	.	-.123	-.123	-.123	-.123	-.123	-.123	-.123	-.123	-.123
Lag 17	-.096	.	-.096	-.096	-.096	-.096	-.096	-.096	-.096	-.096	-.096
Lag 18	-.099	.	-.099	-.099	-.099	-.099	-.099	-.099	-.099	-.099	-.099
Lag 19	-.090	.	-.090	-.090	-.090	-.090	-.090	-.090	-.090	-.090	-.090
Lag 20	-.048	.	-.048	-.048	-.048	-.048	-.048	-.048	-.048	-.048	-.048
Lag 21	.010	.	.010	.010	.010	.010	.010	.010	.010	.010	.010
Lag 22	-.009	.	-.009	-.009	-.009	-.009	-.009	-.009	-.009	-.009	-.009
Lag 23	-.001	.	-.001	-.001	-.001	-.001	-.001	-.001	-.001	-.001	-.001
Lag 24	.815	.	.815	.815	.815	.815	.815	.815	.815	.815	.815

Figure 3.16: ARIMA(2,1,8) Model Residual Table ACF Autocorrelation


Residual PACF Summary

Lag	Mean	SE	Minimum	Maximum	Percentile						
					5	10	25	50	75	90	95
Lag 1	-.003	.	-.003	-.003	-.003	-.003	-.003	-.003	-.003	-.003	-.003
Lag 2	.002	.	.002	.002	.002	.002	.002	.002	.002	.002	.002
Lag 3	.001	.	.001	.001	.001	.001	.001	.001	.001	.001	.001
Lag 4	-.013	.	-.013	-.013	-.013	-.013	-.013	-.013	-.013	-.013	-.013
Lag 5	-.065	.	-.065	-.065	-.065	-.065	-.065	-.065	-.065	-.065	-.065
Lag 6	-.083	.	-.083	-.083	-.083	-.083	-.083	-.083	-.083	-.083	-.083
Lag 7	-.094	.	-.094	-.094	-.094	-.094	-.094	-.094	-.094	-.094	-.094
Lag 8	-.123	.	-.123	-.123	-.123	-.123	-.123	-.123	-.123	-.123	-.123
Lag 9	-.113	.	-.113	-.113	-.113	-.113	-.113	-.113	-.113	-.113	-.113
Lag 10	.090	.	.090	.090	.090	.090	.090	.090	.090	.090	.090
Lag 11	.248	.	.248	.248	.248	.248	.248	.248	.248	.248	.248
Lag 12	.065	.	.065	.065	.065	.065	.065	.065	.065	.065	.065
Lag 13	.248	.	.248	.248	.248	.248	.248	.248	.248	.248	.248
Lag 14	.103	.	.103	.103	.103	.103	.103	.103	.103	.103	.103
Lag 15	-.127	.	-.127	-.127	-.127	-.127	-.127	-.127	-.127	-.127	-.127
Lag 16	-.174	.	-.174	-.174	-.174	-.174	-.174	-.174	-.174	-.174	-.174
Lag 17	-.145	.	-.145	-.145	-.145	-.145	-.145	-.145	-.145	-.145	-.145
Lag 18	-.082	.	-.082	-.082	-.082	-.082	-.082	-.082	-.082	-.082	-.082
Lag 19	.028	.	.028	.028	.028	.028	.028	.028	.028	.028	.028
Lag 20	.107	.	.107	.107	.107	.107	.107	.107	.107	.107	.107
Lag 21	.106	.	.106	.106	.106	.106	.106	.106	.106	.106	.106
Lag 22	.002	.	.002	.002	.002	.002	.002	.002	.002	.002	.002
Lag 23	-.159	.	-.159	-.159	-.159	-.159	-.159	-.159	-.159	-.159	-.159
Lag 24	.750	.	.750	.750	.750	.750	.750	.750	.750	.750	.750

Figure 3.17: ARIMA(2,1,8) Model Residual Table PACF Partial Autocorrelation

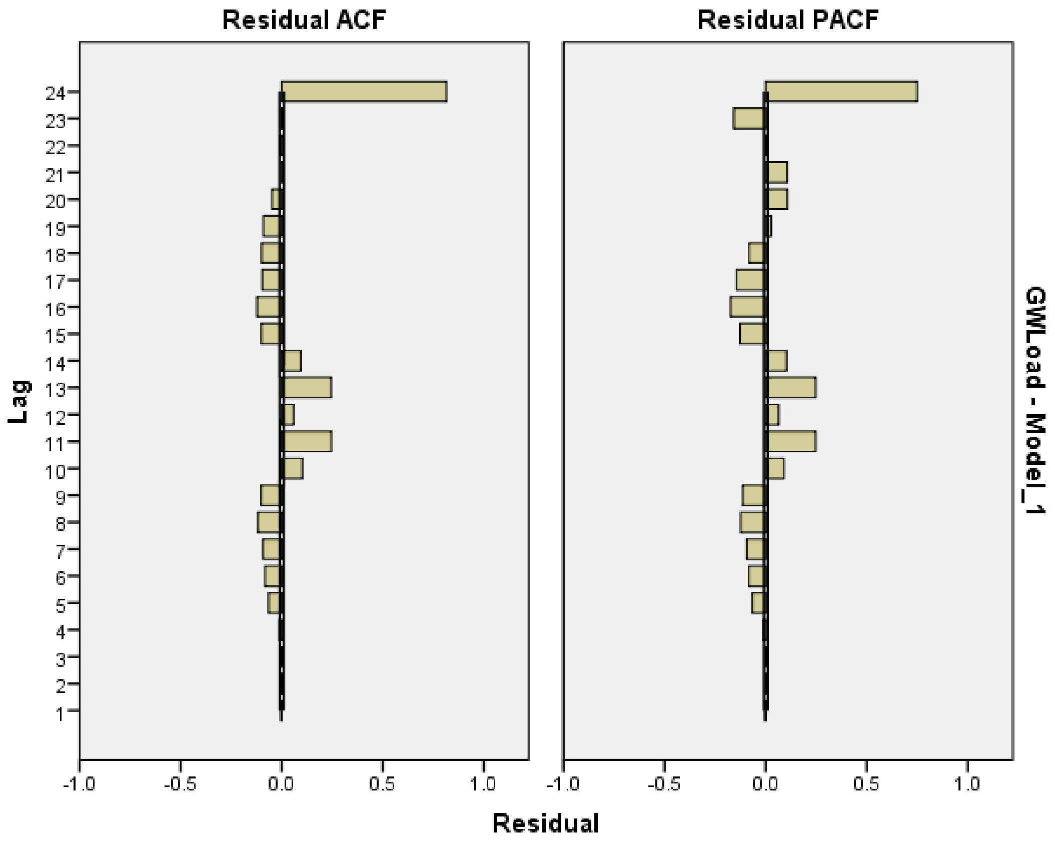


Figure 3.18: ARIMA(2,1,8) Residual Plot Lags

Chapter 4

Overview of Proposed Framework

The proposed method includes a design of a NARX neural network, its structure, inputs and outputs, and a method for simulating such network using a data set. The following sections will describe these points in detail.

4.1 Proposed Framework Description

This section describes the proposed solution: a NARX neural network design that takes exogenous inputs, such as weather and time variables, and endogenous input (electrical load), and produces a 24-hour forecast. We will first dedicate some time to explaining how to arrive to the proposed network, which will facilitate the implementation explained later on.

4.1.1 NARX Model

In order to improve the forecasting accuracy of the time-series model, we can incorporate predictor variables and time-lagged observations as inputs to the model to produce an output defined by

$$\vec{y_{t+1}} = f(\vec{x_1}, \vec{x_2}, \ldots, \vec{x_p}, \vec{y_t}, \vec{y_{t-1}}, \ldots, \vec{y_{t-q}}) \tag{4.1}$$

Notice that in equation 4.1, in addition to the lagged output vector $\vec{y_{t-q}}$, such as that shown in equation 2.2, we incorporate external vector predictor variables $\vec{x_p}$ to the input in order to improve its accuracy. The output vector $\vec{y_t}$ is a 24x1 vector containing the next 24 hours load forecast, all at once. Also, it is important to keep in mind that each of the inputs is in itself a vector of different variables. In the present work, the exogenous predictors $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_p}$ are the time and weather variables shown in Table 4.1. The time-lagged vector variables $\vec{y_t}, \vec{y_{t-1}}, \ldots, \vec{y_{t-q}}$ are the past values of the load. These variables are fed to the model in three vector streams as inputs: 5-year hourly load values, last week's hourly load values and last 24 hours hourly load values. The justification of this overlap is to increase the weight of more recent values of the output variables over the oldest values.

The dynamics of the ANN using NARX can be described by its input-output relationship [107]

$$y(t) = F[x(t), x(t-\Delta t), \ldots, x(t-n\Delta t), y(t), y(t-\Delta t), \ldots, y(t-m\Delta t)] \tag{4.2}$$

Where n is the number of time delay steps in the input and m is the number of time delays on the feedback (output).

The construction of the proposed neural network starts with the structure of a feedforward perceptron network in order to learn the behavior of the output (target) y at time t (y_t), by using inputs x_t, and modeled as a nonlinear functional form of a regressional model for y (output layer)

$$y_t = \Phi[\beta_0 + \sum_{i=1}^{q} \beta_i h_{it}] \tag{4.3}$$

Where (hidden layer)

$$h_{it} = \Psi[\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{jt}] \tag{4.4}$$

Φ is the activation function for the output, which is $\Phi(x) = x$, the linear function, Ψ is the activation function for the hidden neurons, in our case the logistic function of the form

$$\Psi(t) = \frac{1}{1+e^{-t}} \tag{4.5}$$

which is used to flatten or limit the neural weights, β_0 is the output bias, β_i are the output layer weights, γ_{i0} is the input bias and γ_{ij} are the weights of the input layer. i is the subindex of the q neurons, and j is the subindex of the n inputs. Combining equations 4.3 and 4.4 we have

$$y_t = \Phi\{\beta_0 + \sum_{i=1}^{q} \beta_i \Psi[\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{jt}]\} \tag{4.6}$$

Then we add the dynamic term, an autoregression on the output in order to describe a recurrent network where the hidden layers are described by

$$h_{it} = \Psi[\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{jt} + \sum_{r=1}^{q} \delta_{ir} h_{r,t-1}] \tag{4.7}$$

Where δ_{ir} is the weight of the delayed $h_{r,t-1}$ feedback term. By replacing 4.7 into 4.3 we obtain

$$y_t = \Phi\{\beta_0 + \sum_{i=1}^{q} \beta_i \Psi[\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{jt} + \sum_{r=1}^{q} \delta_{ir} h_{r,t-1}]\} \tag{4.8}$$

Equation 4.8 accounts for network dynamics: past values of the output, and multiple inputs. However, our model so far only accounts for one hidden neural layer. We must extend the description to N layers by adding index l and the multi-dimensional nature of the τ outputs by adding index k to yield

$$y_t^k = \Phi\{\beta_0^k + \sum_{l=1}^{N} \sum_{i=1}^{q} \beta_i^l \Psi[\gamma_{i0}^l + \sum_{j=1}^{n} \gamma_{ij}^l x_{jt} + \sum_{r=1}^{q} \delta_{ir} h_{r,t-1}]\} \tag{4.9}$$

Equation 4.9 describes the implemented NARX neural network. The open-loop and closed-loop networks are identical, except where the value of the delayed output is obtained. The open-loop network obtains the value of y from known past values of the output and therefore it is a regular input to the network, and the closed-loop network obtains the value from the predicted value of the output.

4.1.2 Neural Network Description

An exhaustive description of artificial neural networks features and how each one affects the performance of the forecast is beyond the scope of this paper. The most important features that need to be defined in order to construct a well-performing ANN are:

Feedforward or Recurrent The ANN used in the proposed framework is first trained in open-loop, i.e., the training set includes all the historical data for weather variables and also the historical data for the hourly loads. In this sense, it is a feedforward network. Once the node weights are found in open-loop, the network is used to calculate the output, which is then fed as input to the ANN, making it a recurrent network (feedback). Figure 4.1 shows the proposed structure. The reason why the closed-loop is used is that during training the actual output $y(t)$ is available and it is fed to the neural network in order to determine the network weights, but during the forecasting phase the actual output is not available and therefore the predicted delayed output is used to produce a forecast.

Input Size - In order to generate a parsimonious model, i.e., a model that can provide sufficient forecasting accuracy with the minimum set of inputs, a statistical analysis of correlation between input and output was carried out for each variable with respect to estimated load. Table 4.2 below shows the subset of parameters that yield the best performance with the smallest input set. The flowchart in Fig.

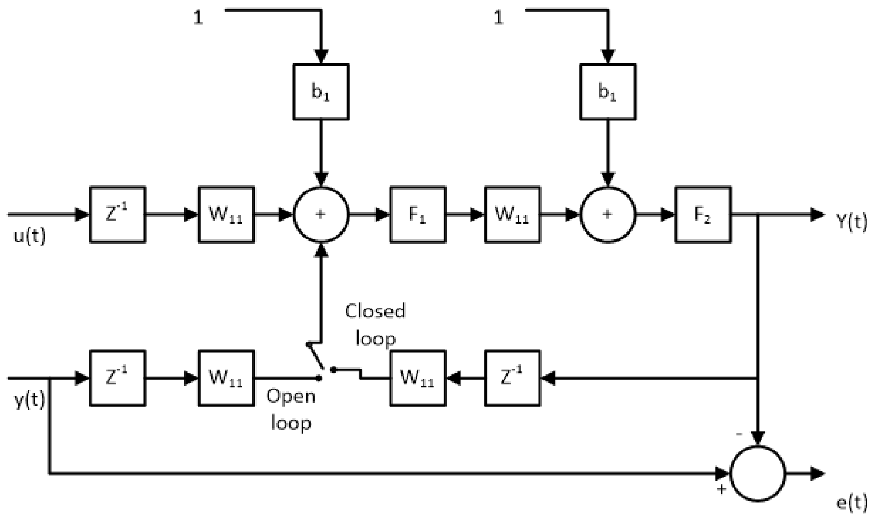


Figure 4.1: NARX Artificial Neural Network Structure

4.2 shows the procedure for preparing and processing the data through the ANN.

Data Normalization Data is normalized in pre-processing in the range [0,1] by using feature scaling, i.e. the equation

$$X^{'} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.10}$$

Table 4.1 summarizes the ANN Model Input.

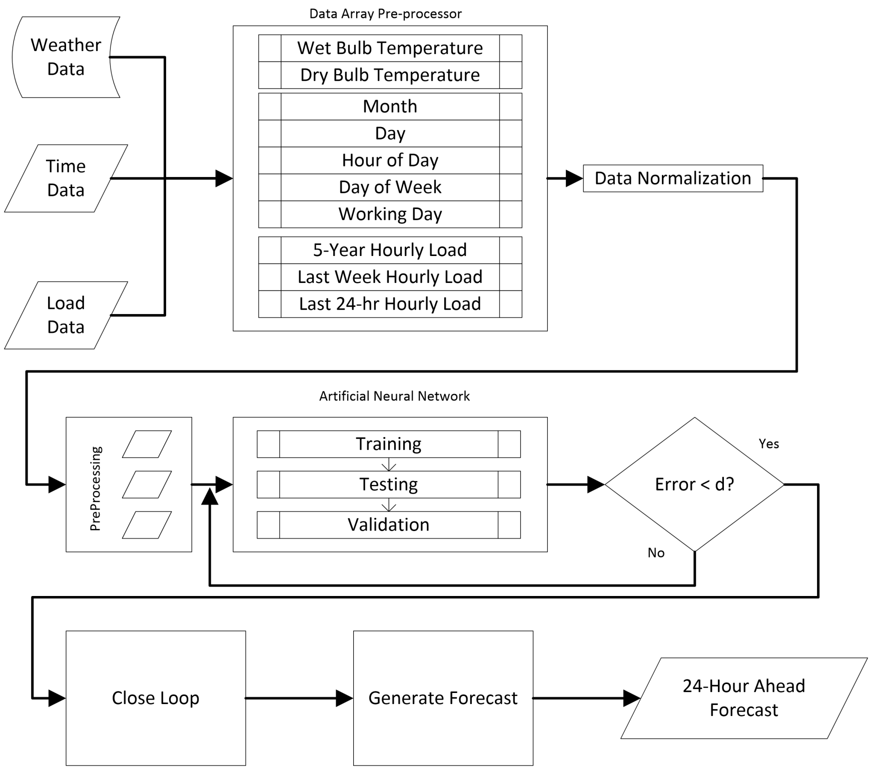


Figure 4.2: Proposed Flowchart

Table 4.1: ANN Model Input.

Exogenous	**Endogenous**	**Time Lag**
Month	Load	Last 24 h
Day		
Hour of Day		
Day of Week		
Working Day		
Wet Bulb Temperature		
Dew Point		

Table 4.2 summarizes the ANN Architecture.

Table 4.2: ANN Architecture.

ANN Architecture	
Input Nodes	One node for each input variable
Hidden Layers	1
Hidden Nodes	Equal to number of input nodes
Output Nodes	Equal to size of forecasting horizon (1 node)
Interconnection	Full
Activation Function	Sigmoid Function $$S(t) = \frac{1}{1+e^{-t}} \quad (4.11)$$
Learning Algorithm	Levenberg-Marquardt back-propagation

Training and testing sample size - The method utilizes 70% of the available data for training, 15% for testing, 15% for validation

Sample size - We used varied sample sizes from two years to five years. It is important to go over a period that covers more than one year in order to have the network learn the seasonal characteristics of the load.

Performance measurements The measure of performance used in this work is the Mean Absolute Percent Error defined as

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|E_i| \times 100\% \tag{4.12}$$

where the error E_i is given by

$$E_i = \frac{L_i^{actual} - L_i^{forecast}}{L_i^{actual}} \tag{4.13}$$

The error has been calculated in two instances: in open loop, the error corresponds to the fit error, i.e., the difference between the actual value and the output value of the open loop network for the validation set. This error is used to select the best network, which is the one that will be used for forecasting. Once the forecast is produced in closed-loop, the error is calculated between the actual value of the load, which is known to us because we are using past data for validation, and the closed-loop forecast output. This error is the error reported in the results as forecast error.

4.2 Running the Network

Once the network is defined and the performance measures are selected, we must develop the detail: what data is used, how inputs are defined, what simulation environment is selected.

4.2.1 Operating Procedure

Starting with the data set including year, month, date, day, hour, day of the week, whether day is a working day or a holiday, dew point temperature and dry bulb temperature, as well as hourly load, for 5 years, we first segment the data into 1-year groups. Each 1-year group is used for testing. The data is also aggregated with last week load data (as a redundant input), and last day (24 h) also as a redundant input. The data is then prepared so that the input size matches the number of input neurons. The initial weights are assigned in order to guarantee stability as discussed later. The resulting data set is split into training, validation and testing sets. The training set undergoes Levenberg-Marquadt backpropagation in order to obtain the open-loop weights. Testing and validation are done on the data set. This process is repeated 10 times each for 5 to 30 neurons in increments of 5. At the end, the best fitting open-loop network is selected, based on minimum MAPE criterion. The process does not guarantee an optimal solution, just the selection of the best solution available from within many different parameters. The best open-loop network is selected, the loop is closed, and the date and weather data forecast is fed to it in increments of one hour, in order to produce one-hour forecast at a time for the next 24 h. Now the model utilizes its own output load forecast as an input to generate the next hour forecast. This is how the 24-hour forecast is obtained.

4.2.2 Model Parameters

The parameters of the model are the inputs, the lag of the inputs, the number of hidden layers, the number of neurons in each layer, and the connectivity between neurons. The selection of the parameters was done as follows: For the inputs, a multiple correlation and partial correlation analysis was performed between a set of available variables, and the subset with the highest correlation was selected based on a stepwise selection. The lag of the inputs was tested from 1 h to 168 h, obtaining a good trade off between performance and error for 24 h. The number of hidden layers was tested from 1 to 5, with significantly better results for 1 hidden layer. The number of neurons in the hidden layers was tested from 5 to 30 neurons, and it was proven that a number between 10 and 15 neurons worked best. A fully connected network following a NARX definition was used as the selected model. Again, the parameters selected do not guarantee an optimal result, but the best result available from within the tested solutions.

4.2.3 Simulation Environment

The NARX and FitNet ANN models were simulated using Matlab 2016 Neural Networks Toolbox. The ARMAX and State Space models were simulated using

IBM's SPSS Statistical Package.

Two sets of simulations were run in MATLAB: one set for a feedforward neural network which is used for performance comparison, as it is the *de facto* standard for neural network data forecasting, and a second set for a NARX neural network as described in this paper. Both sets were run with the same dataset and the same number of iterations on the following variables:

- Number of data groups—10 different data periods were used
- Number of neural networks—5 different runs for each iteration in order to separate the outcome from the initial conditions
- Number of neurons in the hidden layer—simulations were run from 5 to 30 neurons in increments of 5

In addition, an ARMAX and a State Space Estimation model were developed for comparison using IBM's SPSS statistical package. The best ARIMA model produced in SPSS was a an ARIMA(2,1,8), which is using a second order autoregressive order, 1 degree of differencing, and an 8th degree moving average. The ARIMA model was selected from SPSS output utilizing Expert Modeler in order to minimize the fit error. The resulting ARIMA model was transformed into a state-space model to obtain the comparison. The best NARX network based on MAPE fit of the existing data in open loop was selected in order to provide a forecast. The NARX neural network was trained in open-loop using historical data, and then used for forecasting in closed-loop using the calculated load as the input for the next step. The MATLAB network configuration is shown in Figure 4.3.

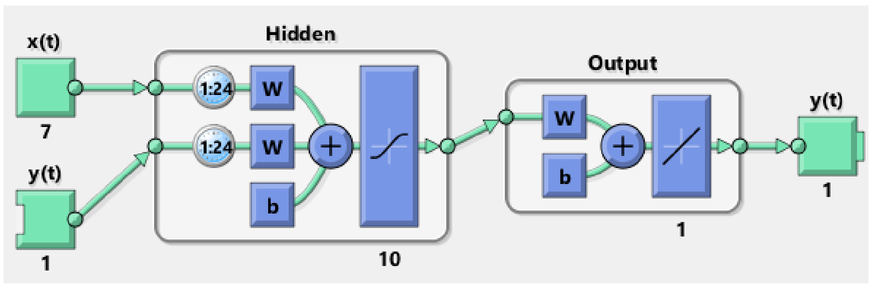


Figure 4.3: MATLAB NARX Network Configuration.

4.3 Data Set

Hourly real-time system demand load data was obtained from the ISO-NE grid operator for New England from 2005 to 2015 [105]. The statistical characteristics of the load data are shown in Figure 3.5. Notice the seasonal characteristic of the load, i.e., variability month to month; The particular reduction in the load on weekends, and the variation in the load according to the hour of the day. These features of the data are very appropriate for using ANNs since all the distinct features are identified by the parameters in the network. A correlation analysis was carried out from a set of candidate independent variables, and the result of this study is the input vector selected for the proposed framework shown in Table 4.2.

Stability An important issue to discuss about closed-loop systems is stability. Many researchers have studied the problem of closed-loop stability and have demonstrated that an adequate selection of initial weights is essential for convergence of the solution during the learning process. Furthermore, Irigoyen et al. [108] provide a framework for determining initial weights for NARX networks. Their results show that a sufficient condition for a NARX network in closed-loop with two layers and K sigmoid neurons in the output layer and J in the hidden one, to be stable is that the initial weights w_{kl}^{m} have a modulus that meets the requirement

$$|w_{kl}^{m}| \leq \frac{4}{\sqrt{JK}} \tag{4.14}$$

In the case of the proposed model $J = 10$ and $K = 1$, and therefore we must have initial weights that meet

$$|w_{kl}^{m}| \leq \frac{4}{\sqrt{10}} \approx 1.26 \tag{4.15}$$

The initial weights were started in the range $[-1.26, 1.26]$ to ensure stability. Also the convergence curves shown for training show no signs of divergence of the error.

Chapter 5

Effects of the Selection of Parameters

5.1 Motivation

Artificial Neural Networks (ANN) have been in use since 1943 [109]. One implementation of neural networks consisted on the perceptron, a mathematical device that mimics the way a neuron works by producing an output that is the linear combination of the input and synaptic weights. Training algorithms were proposed in order to adjust those weights so that by giving the networks several sets of input and output the network ”learns” the relationship between them and can respond to new inputs in the correct manner. One of the most effective neural network training algorithms is back propagation. This algorithm consists on a method that adjusts the weights in the network based on the error produced by the network when the current weights are used. The inputs are given to the network and the output is compared to the input to generate an error value. This error is then used to adjust the synaptic weights and propagated backwards through the layers of the network until all weights have been adjusted. The method uses a learning rate which is a value that determines how large the adjustment should be. The weight updates can be calculated by a number of methods proposed by different authors, most of them using some form of the gradient descent method, i.e. move in the direction of the steepest descent in error, thus ensuring faster convergence to the solution. Much research has been carried out in order to reduce the likelihood of the solution error being trapped in local minima giving as a result a sub-optimal solution.

Nonlinear Artificial Neural Networks with Exogenous Inputs (NARX) have been successfully used for time series forecasting [82]. NARX networks are a type of dynamic neural networks, more precisely a recurrent neural network, in which the input is lagged in order to create a ”memory” effect between input and output. In addition NARX networks utilize their own output as input by means of a feedback loop that makes it possible to train the network using all known inputs, and then splitting the inputs into known (exogenous) and unknown (endogenous) inputs and using the output of the network to feed the endogenous inputs back. This method is very useful in creating forecasting networks for time-series inputs, because the memory effect can be effectively created by using time-lagged values of the exogenous and endogenous inputs. This research focuses on improving the robustness and forecasting accuracy of NARX neural networks.

ANNs can be configured with a large number of parameters and architectures. First of all architectures can vary by changing the number of hidden layers and the way they are interconnected, adding feedback loops from output to input, adding interconnections between layers (consecutive or non-consecutive layers), adding feedback between layers, introducing parallel layers, etc. In addition to changing architectures, many parameters can be also modified in the scheme: the number of neurons in each layer, how many time lags are used for the input and the feedback, how much data to use to train the network, what training algorithm to use, the activation functions of the neurons, etc. This variety of network topologies and parameters provides a rich supply of arrangements that enables specific configurations to work best with different data sets.

In working with NARX networks for forecasting short-term electric loads in a power grid, the authors in [82] found that NARX networks, while giving very accurate forecasts, are very sensitive to the parameter selection. Their results are highly dependent on the number of neurons selected, the activation function in the hidden layers (the activation function of the output layer being $f(x) = x$), the training algorithm and the length of training data set. This part of the research proposes a technique which makes the NARX less sensitive to the selection of those parameters, i.e. more robust to parameter selection, and thus ensures higher accuracy of the forecast. The technique proposed is adding an exponential decaying mask to the error weight during the training phase of the ANN. By applying this technique, more weight is given to errors occurring later in the time series (more recently), and less weigh is given to errors occurring earlier on the time series (older data). An analysis of the results of applying this technique in a variety of scenarios by varying all the other parameters is necessary and the consequences of applying the technique on both, increasing the robustness of the ANN and improving the overall network forecasting accuracy, is necessary.

5.2 Proposed Improvement

In this section we will start by describing the NARX network being used for study; then we proceed to demonstrate the results in forecasting accuracy due to variation of several parameters; we proceed to describe the proposed modification to the training algorithm; and finally we show the results of the proposed modification to the model presented in Chapter 4.

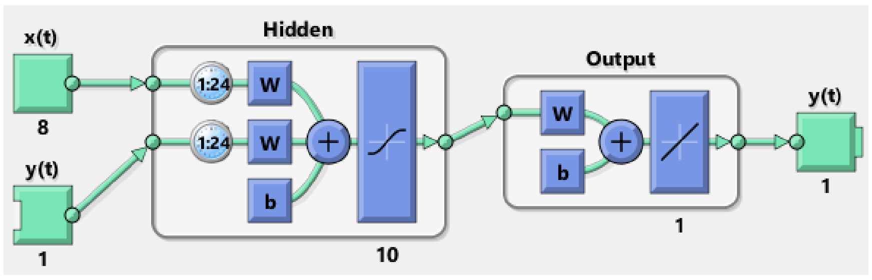


Figure 5.1: NARX Network Configuration in Open Loop - Training

5.2.1 NARX Network Description

Throughout this study the author used a NARX network in MATLAB (see Figure 5.1) that processes the following input streams: i) $x(t)$ are the exogenous inputs: Year, Month, Day, Day of the Week, Working Day, Wet Bulb Temperature (°F), Dew Point Temperature (°F). $y(t)$ are the known hourly power loads (GW). The network in Figure 5.1 is shown in training mode (open-loop) when the load quantities are known. Once training ends, the network is closed and the generated $y(t)$ is fed back as an endogenous input (closed-loop mode shown in Figure 5.2). Notice that in the networks shown, there is a 24-hour feedback lag (on $y(t)$), and also a 24-hour lag on the exogenous input $x(t)$. These lags do not need to be equal, this is a choice. Also the networks shown reflect 10 neurons on the hidden layer, a hyperbolic tangent activation function for the hidden layer and a linear activation function for the output layer. All of these parameters are varied and their results of the combinations will be shown in Section 6.2.1.

Based on the information shown by Figures 5.1 and 5.2, the number of inputs given to the network at a given time t are for this example:

$8\ exogenous\ inputs\ x\ 24\ delays\ +\ 1\ endogenous\ input\ x\ 24\ delays = 216$

There is only one output of the neural network which is $y(t)$ the load, which is a $24x1$ vector containing the forecast for the next 24 hours.

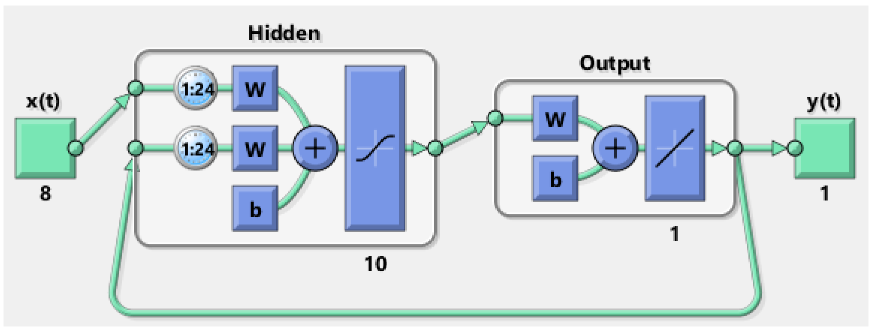


Figure 5.2: NARX Network Configuration in Closed Loop - Forecasting

Two different activation functions have been used in the hidden networks: Logistic Sigmoid and Hyperbolic Tangent. Logistic sigmoid and hyperbolic tangent. The objective of an activation function is to limit the range of the outputs to a finite number.

Logistic Sigmoid activation function The logistic sigmoid function is defined as

$$\Psi(x) = logsig(x) = \frac{1}{1+e^{-t}} \tag{5.1}$$

and it produces an all-positive output in the range $[0,1]$.

Hyperbolic Tangent activation function The hyperbolic tangent function is defined by

$$\Psi(x) = tansig(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5.2}$$

and it produces an output in the range $[-1,1]$.

The functions described in Equations 5.1 and 5.2 take the place of Ψ in the NARX equation used by Buitrago[82] and shown in Equation 4.9:

$$y_t^k = \Phi\{\beta_0^k + \sum_{l=1}^{N}\sum_{i=1}^{q}\beta_i^l\Psi[\gamma_{i0}^l + \sum_{j=1}^{n}\gamma_{ij}^l x_{jt} + \sum_{r=1}^{q}\delta_{ir}h_{r,t-1}]\} \tag{5.3}$$

where $k = 1 \ldots \tau$ are the outputs, Φ is the activation function for the output layer, which is $\Phi(x) = x$, the linear function; Ψ is the activation function for the hidden neurons; β_0 is the output bias; β_i are the output layer weights; γ_{i0} is the input bias; and γ_{ij} are the weights of the input layer. i is the subindex of the q neurons, and j is the subindex of the n inputs; N is the number of layers using index l.

Two different network training algorithms were used: Levenberg-Marquadt back propagation and Bayesian Regularization.

Levenberg-Marquadt Algorithm The Levenberg-Marquadt (LM) back propagation method [84] is widely used in training ANNs [110, 111]. It interpolates between the Gauss-Newton (GN) algorithm and the Gradient Descent (GD) method. The Gauss-Newton method is an improvement over the Newton method that doesn't impose restrictions on the existence of the second derivatives. The Newton method uses a second order Taylor expansion of the function and requires the use of first and second derivatives (Hessian) of the function to calculate the step. GD uses the Jacobian (first-order partial derivatives) of the errors in order to calculate the direction of motion in the error space. The GD method uses the steepest gradient of the error, which is a generalization of the Newton method. In general terms it can be said that the Newton (and thus GD method) moves faster along the error space towards a minimum than the LM method. The LM, although slower than the GN method, has proven a more robust method for finding error minima in ANNs, but due to the nature of the method, it can get stuck in local minima. The advantage of the LM algorithm is that it restricts (via a trust region) the magnitude of the step, making it slower but more reliable. The other advantage is that it does not have to compute the Hessian matrix, but instead at each step the algorithm makes a second-order approximation of the error and estimated the gradient linearly. This is much faster for computational purposes. The interpolation done by the LM algorithm can be explained by the equation

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e \tag{5.4}$$

where x_{k+1} is the value of the next step, J is the Jacobian, I is the identity matrix and e is the computed error. The value of μ varies in the range $[0, 1]$. When μ is 0, Equation 5.4 reduces to the Newton method; when μ approaches 1, it reflects the gradient descent method, which is slower. Since Newton's method is faster near a minimum, the algorithms attempts to make μ move smaller after each iteration, increasing μ only when that iteration step would increase the error objective function.

Bayesian Regularization Algorithm The Bayesian Regularization method (BR) utilizes LM to update the synaptic weights and biases [92, 96]. BR actually occurs within LM. At each step BR calculates a least squares regression on a linear combination of the weights and biases and discards the ones that are irrelevant, thus giving a set of adjustments that are a better generalization of the data. The validation process reduces the $O(N2)$ computation associated with normal regression methods (back propagation), and utilize a stopping criterion that avoids overfitting [93]. Other training methods are based in maximum likelihood techniques in which an objective error function is minimized and a single set of values for the weights is sought. The BR method, on the other hand, takes into consideration a probability distribution function spanning the weight space which represents the relative degrees of belief in the values of the synaptic weights and biases. At the beginning, the probability distribution function is assumed, and as the training progresses, the function can be converted to a posterior distribution by using Bayes' theorem [112]. The posterior distribution is used to evaluate the predictions of the trained network for new values of the inputs. BR tends to favor smaller weights because smoother solutions are preferred, and thus the overfitting risk is lower with BR than with LM.

5.2.2 Proposed Modification to NARX Network

Increasing the number of neurons in a neural network is akin to increasing the order of a polynomial regression. There is the possibility of improving the fit, but after a certain degree the polynomial tends to overfit the data. With a neural network it is very similar: increasing the number of neurons increases the degrees of freedom in the weight/bias solution and therefore there is a possibility of overfit. Avoiding overfitting is particularly relevant when the network (or polynomial regression) is to be used for forecasting. A perfect fit does not ensure a perfect forecast, in fact it may make the forecast more volatile. A smoother solution is almost always preferable. By reducing the impact of the error corrections that older training data can effect on the solution, and at the same time increasing the influence of newer data on the time series we can smooth the solution while maintaining or even improving its accuracy. But more relevantly, we can desensitize the network to variations in parameters such as number of neurons.

The proposed solution is to apply a mask on the error weights which gives a relatively higher weight to the errors in more recent data, and exponentially decreases the weight of the errors generated by lagged data.

Krogh [104] experimented with a linear weight decay with success. Hinton [86] demonstrated that this decay form leads to improvement of generalization. Moody [97] also mentions that this method of weight decay is effective. The error weight decay generalized function proposed by Krogh is

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2 \tag{5.5}$$

where E(w) is the error function (e.g. sum of squared errors), w, the weight vector, contains the weights and biases of the network, and λ is the decay constant. When gradient descent is used for training, the cost function will contain a new term $-\lambda w_i$ for the weight update:

$$\dot{w_i} \propto -\frac{\partial E_0}{\partial w_i} - \lambda w_i \tag{5.6}$$

if the gradient $\frac{\partial E_0}{\partial w_i}$ is not present, then the decay is exponential. But this exponential decay of the errors is not the one proposed by the current work. We are referring to exponentially decay the error weights based on time: assign a higher weight to those error weights due to recent data and reduce the weight (exponentially) to those errors associated with older data. The key variable is time t.

When the error weights $E(w)$ are constant, the error function for the *ith* synapse is

$$E(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5.7}$$

where N is the number of neurons, y is the target value of the output, and $\hat{y}$ is the calculated output by the network. By adding exponential decay, the weight function becomes

$$E(w) = \frac{1}{N}\sum_{i=1}^{N}(1-\epsilon)^{N-i}(y_i - \hat{y}_i)^2 \tag{5.8}$$

The exponential decay of error weights was applied to all the simulated neural networks in the preceding section, and the results are analyzed in the following section.

5.2.3 Finding the Optimal Time Lags for Input and Feedback Signals

In order to determine the optimal structure for the network it will be necessary to iterate on the number of neurons and save the best fitting network for forecasting purposes. However, for optimizing the number of lags this section describes a procedure. The input and feedback lags must be found following a procedure since their impact is very difficult to determine analytically due to the recursive nature of the output solution. One direct approach is to stop training when a selected minimum error is reached. This method would require to retrain as many times as necessary until the threshold is passed and an acceptable error is reached. This would require i) setting a pre-determined error threshold which is difficult to do *a priori* because little is known of the ability of the network to generalize the data up front. ii) many iterations which are time consuming in order to reach the minimum error threshold.

Another approach can be more effective to select a training stopping criterion. If the data set is separated into three sets, as described by Luk [113], into a training set, a validation set and a testing set, a much more reliable stopping method can be achieved. The training set is used, as is logical for training. But training is not stopped based on the results of the training set, but instead, a second set, the validation set is monitored. When the error reaches a minimum in the validation set, and it starts to climb again, the training stops. The third set, the testing set is used to evaluate the performance of the network. The rationale behind using the second set as the stopping criterion lies in the fact that the error on the training set is usually constantly decaying. On the other hand, the error in the validation set decreases until a minimum is reached and then it starts climbing up again.

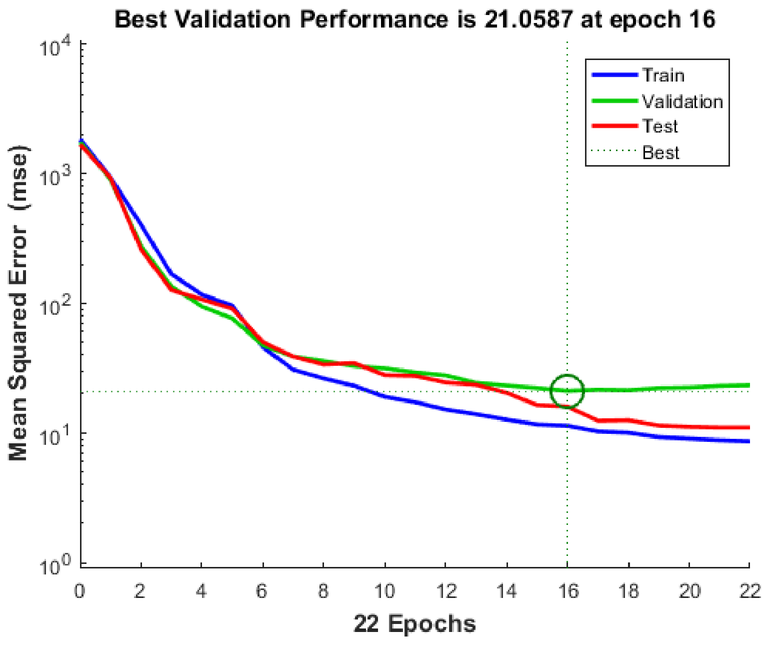


Figure 5.3: Training Set Error and Validation Set Error

The discrepancy between the behavior of the two sets is due to the fact that the training algorithm works constantly to reduce the fit error on the training set, and therefore the next step usually points to a lower error (in the fit), but this error does not necessarily means that beyond a certain point the network generalizes the behavior of the data in the optimum manner, but instead that the network is overtrained, and therefore, if used with data other than the training set, the fit is not so good. The validation set, on the other hand, corresponds to data that has not been used by the training algorithm, and thus whenever the error is minimum on this set, the network generalization capabilities are the best. Figure 5.3 shows this behavior: The training set, shown in blue, decreases its error at every step (epoch) of the training. Conversely, the validation error, shown in green, has a minimum at 16 epochs, which is the right moment to stop training.

Using this approach to training stopping, we can retrain the network for different input and feedback lags and stop early in order to determine the optimum

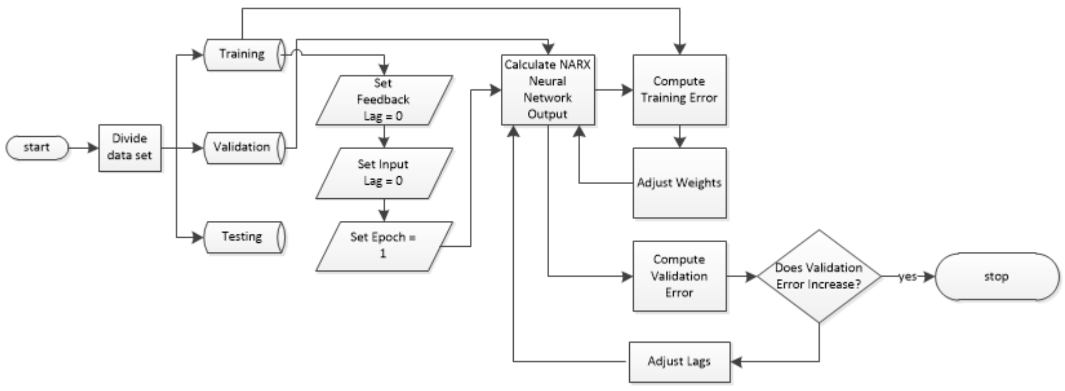


Figure 5.4: Flowchart for finding optimum values for input and feedback lags

input and feedback lags without having to completely retrain the network seeking a fixed threshold. Figure 5.4 shows a flowchart of the process of finding the optimum values for the input and feedback lags.

Chapter 6

Results and Analysis

6.1 Simulation Results for Proposed NARX Network

Two sets of simulations were run in MATLAB: one set for a feedforward neural network which is used for performance comparison, as it is the *de facto* standard for neural network data forecasting, and a second set for a NARX neural network as described in this paper. Both sets were run with the same dataset and the same number of iterations on the following variables:

- Number of data groups - 10 different data periods were used
- Number of neural networks - 5 different runs for each iteration in order to separate the outcome from the initial conditions
- Number of neurons in the hidden layer - simulations were run from 5 to 30 neurons in increments of 5

In addition, an ARMAX and a State Space Estimation model were developed for comparison. The best network based on MAPE fit of the existing data in open loop was selected in order to provide a forecast. The NARX neural network was trained in open-loop using historical data, and then used for forecasting in closed-loop using the calculated load as the input for the next step.

Table 6.1 shows the results of the simulations for a 24-hour ahead load forecast and the corresponding error performance comparison between the feedforward neural (FitNet) network, the ARMAX model, the State Space estimation, and the NARX neural network. The NARX forecast was generated in closed-loop, i.e., the network was trained in open-loop by using known values of the load; then, the first hourly load forecast value is calculated with the trained network, and that value is fed back to the input in order to obtain the second value recursively, and so on. The open-loop fit of the NARX model is shown in Figure 6.1.

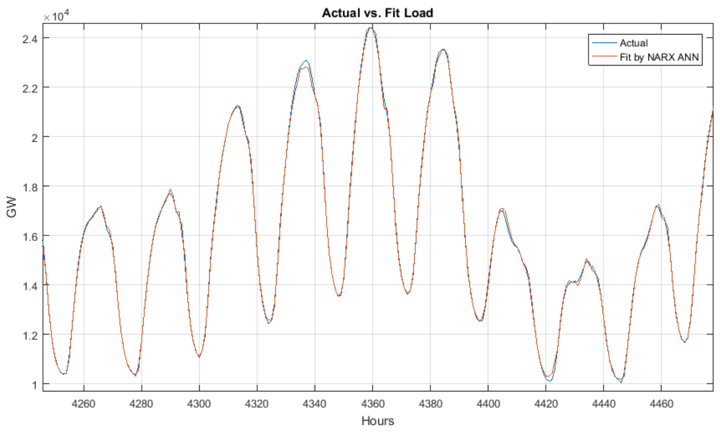


Figure 6.1: Open-loop fit of the NARX ANN model for one week.

As seen on the results Table 6.1, the Mean Absolute Percent Error (MAPE) for the NARX neural network is 0.85%, for the feedforward network is 1.55%, for the ARMAX model is 1.09%, for the state space estimation is 2.53%, which shows a great improvement accuracy offered by the NARX network. The maximum absolute percent errors are also shown in the table. Figure 6.4 shows the forecast tracking for each of the methods used.

The distribution of the hidden layer synaptic weights is shown in Figure 6.2. No silent synapses were found in the model, and the highest weight was given to the input associated with the day of the week, followed by the dry bulb temperature.

Table 6.1: Results

		FitNet		ARMAX		StateSpace		NARXNet	
Hour	Actual	Forecast	AE%	Forecast	AE%	Forecast	AE%	Forecast	AE%
1	12115	12,165	0.41	12248	1.1	12269	1.27	12,141	0.21
2	11576	11,713	1.19	11749	1.49	11553	0.2	11,657	0.7
3	11254	11,349	0.85	11278	0.21	10679	5.11	11,155	0.88
4	11073	11,074	0.01	10933	1.26	10090	8.87	11,229	1.41
5	11084	10,963	1.09	10886	1.79	10901	1.65	11,134	0.45
6	11326	11,123	1.79	11191	1.19	11246	0.7	11,449	1.08
7	11780	11,649	1.11	11624	1.32	12067	2.44	11,790	0.09
8	12338	12,456	0.96	12054	2.3	13132	6.44	12,270	0.55
9	13193	13,402	1.59	12932	1.98	13532	2.57	12,960	1.77
10	13752	13,757	0.04	13694	0.42	13452	2.18	13,790	0.28
11	13978	13,894	0.6	14023	0.32	13502	3.41	13,997	0.14
12	14018	13,867	1.07	14077	0.42	13825	1.38	14,022	0.03
13	13980	13,708	1.94	14076	0.69	14098	0.85	14,052	0.52
14	13900	13,660	1.73	14041	1.02	14141	1.74	13,842	0.42
15	13905	13,777	0.92	14051	1.05	14260	2.55	13,928	0.16
16	14298	14,053	1.71	14554	1.79	14782	3.38	14,153	1.01
17	15634	14,726	5.81	15916	1.8	15509	0.8	15,452	1.16
18	16275	15,280	6.11	16452	1.09	15931	2.11	16,725	2.77
19	16144	15,500	3.99	16257	0.7	15855	1.79	15,935	1.3
20	15633	15,412	1.41	15769	0.87	15506	0.81	15,872	1.53
21	15020	14,879	0.94	15227	1.38	15048	0.19	15,241	1.47
22	13927	13,974	0.34	14069	1.02	14319	2.81	14,141	1.54
23	12656	12,811	1.22	12679	0.18	13179	4.13	12,573	0.66
24	11509	11,553	0.38	11584	0.65	11885	3.27	11,546	0.32
Maximum			6.11		2.30		8.87		2.77
MAPE%			1.55		1.09		2.53		0.85

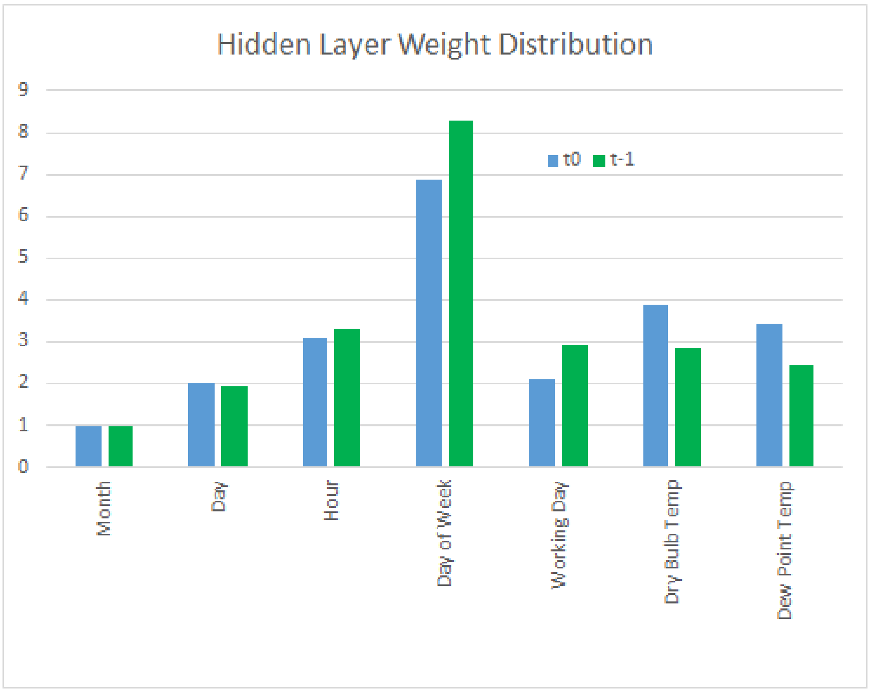


Figure 6.2: Synaptical Weight Distribution

The training, by means of a Levenberg-Marquadt backpropagation algorithm converged after fewer than 50 epochs and it showed stability (no increase after converging) and no overshoot (no increase before converging), as shown in Figure 6.3.

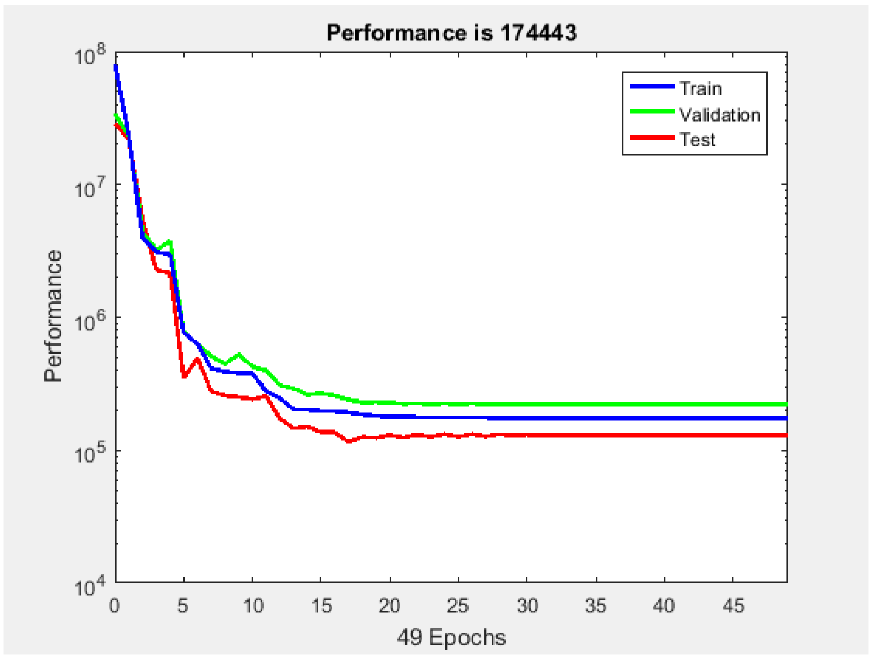


Figure 6.3: Training Performance Graph.

The algorithm was run under many different conditions and configurations. The following items have been taken into consideration:

- Different input periods (both in time and length) —this is to include and exclude the effects of holidays, working and non-working days, seasons, etc. ANN NARX work well when sufficient cyclical data are available for training.
- The number of neurons in the hidden layer in the model —simulations were run from 1 to 30 neurons in the hidden layer. Typically the forecast accuracy peaks at around 20 neurons.
- The number of delayed inputs and feedback —simulations ranging from 1

hour to 48 hour delays were performed. A 24-hour time delay for inputs worked best.

- The redundancy on the input data —reinforcement of training data with the most recent available data. It was found that feeding redundant data for last week and last 24 hours worked best.

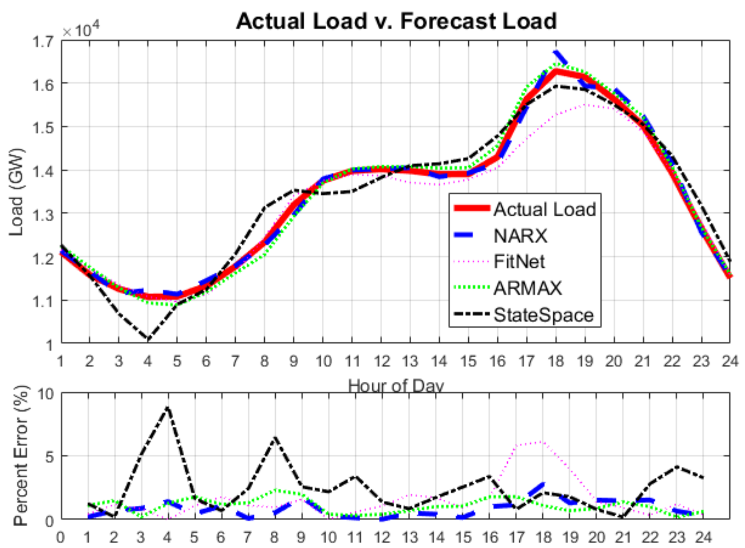


Figure 6.4: 24-hour Actual vs. Forecast Comparison

Figure 6.4 shows the forecast tracking for each of the methods used. Figure 6.5 shows the 95% confidence level range of the 24-hour forecast.

In addition, the error histogram for the fit set is shown in Figure 6.6. The regression on the error between forecast and actual is a measure of performance, and it is plotted in Figure 6.7.

A comparison of the results for regular days (Tuesday to Friday) versus special days (Mondays, weekends and holidays) is shown in Figure 6.8. The results are very similar for both groups of days.

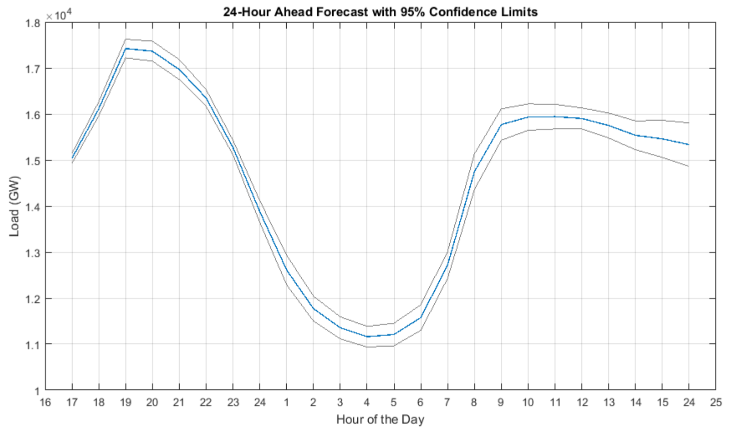


Figure 6.5: 24-Hour Ahead Forecast with 95% Confidence Level Range

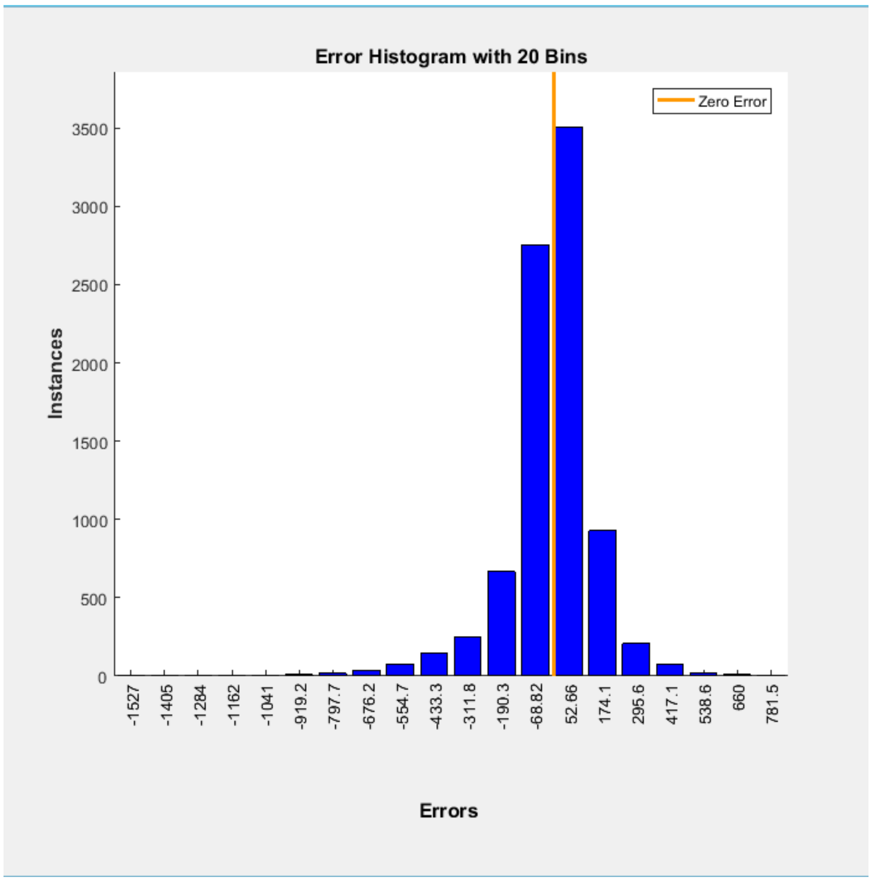


Figure 6.6: Error Histogram for Fit Set.

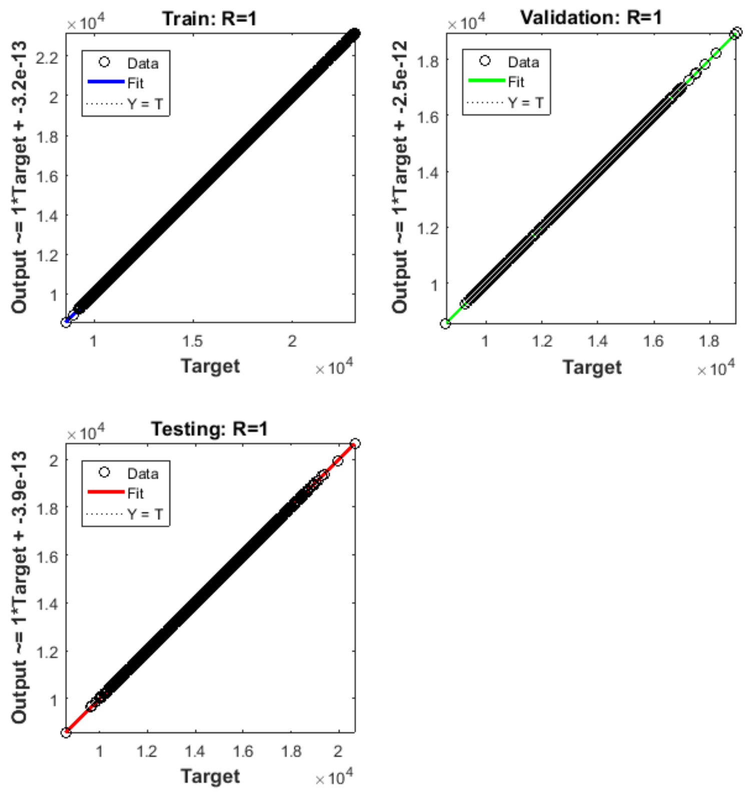


Figure 6.7: Error Regression on Forecast vs. Actual.

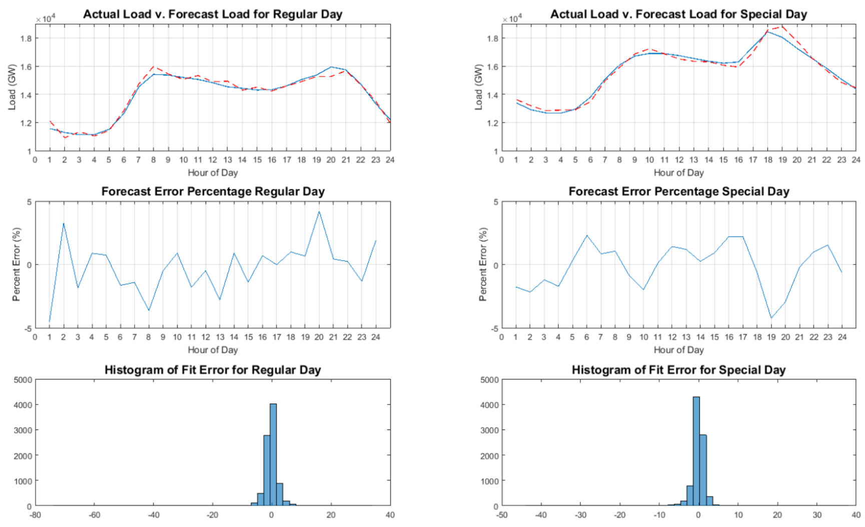


Figure 6.8: Comparison between results for regular vs. special days.

6.2 Simulation Results for Proposed Exponential Error Weight Decay Improvement

6.2.1 Experimental Results with Existing NARX Network before Improvement

Using the network and parameters described in Section 5.2.1, simulations were run in Matlab [114] by varying the parameters in a Monte Carlo simulation scheme as shown in Table 6.2 using data from ISO New England, the operator of the electrical power grid for the New England states [105].

The Mean Absolute Percent Error (MAPE) was used as the performance measure for this research. The MAPE is defined as

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}|\frac{A_t - F_t}{A_t}| \tag{6.1}$$

where A_t is the actual value and F_t is the forecast value.

Table 6.3 shows the results of all the simulations. In total 6400 (10 of each x 20 neuron numbers x 2 training algorithms x 2 feedback delays x 2 training intervals x 2 activation functions x 2 error weight functions) neural networks were designed, trained and used for forecasting the next 24 hour load, and their results recorded in terms of forecasting MAPE and its standard deviation. The results recorded in Table 6.4 are the average of the 10 runs and the 1-20 neurons.

Table 6.2: Simulation Parameters

Parameter	**Values**
Number of Runs	10
Number of Neurons	1-20
Training Algorithm	LM/BR
Hidden Layer Activation Function	Logistic Sigmoid/Hyperbolic Tangent
Training Data Length (hours)	7x24 / 14x24
Input/Feedback Time Lag (hours)	24 / 48

Table 6.3: Simulation Results with Existing Network

Training Algorithm	Feedback Delay	Training Interval	Activation Function	MAPE	Std. Dev. of MAPE
Bayesian Regularization	24	7 x 24	LogSig	4.05	2.52
			TanSig	1.97	0.68
		14 x 24	LogSig	10.3	1.06
			TanSig	10.58	0.89
	48	7 x 24	LogSig	9.22	4.04
			TanSig	9.23	4.39
		14 x 24	LogSig	8.92	0.75
			TanSig	9.6	1.27
Levenberg-Marquadt	24	7 x 24	LogSig	6.9	1.81
			TanSig	12.16	5.53
		14 x 24	LogSig	11.48	1.93
			TanSig	13.98	2.73
	48	7 x 24	LogSig	8.76	1.74
			TanSig	17.45	7.38
		14 x 24	LogSig	12.07	2.08
			TanSig	15.74	2.75

Each data point is the result of 10 simulations (10 different training and forecasting exercises) for one network. The graph shown in Figure 6.9 shows the resulting MAPE for both the Logistic Sigmoid (LogSig) and the Hyperbolic Tangent (TanSig) activation functions in the hidden layer using the Levenberg-Marquadt training algorithm. As seen in this graph, as the number of neurons in the hidden layer is increased, the error gets increasingly large for the TanSig activation function, which does not occur with the LogSig activation function. The best MAPE obtained by either activation function is about 5%.

Figure 6.10 shows the same simulation but using the Bayesian Regularization technique. In this case, both activation functions improve in performance with the BR algorithm, resulting in an improved error. The best MAPE obtained (using TanSig activation function) is about 2.5%. We will analyze these and other observations in the context of the overall results in the following sections.

We must first focus on the results that have the largest impact, and as with many dynamic neural networks, it has to do with the selection of feedback and

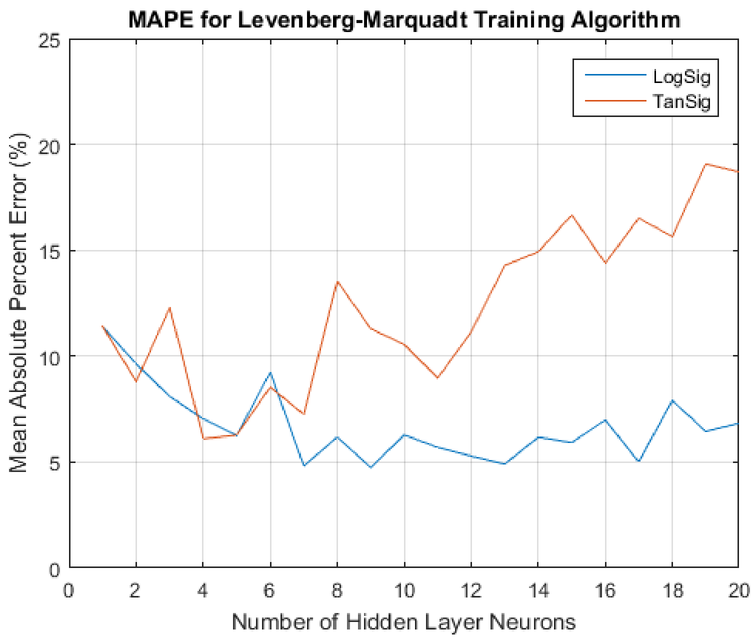


Figure 6.9: MAPE error for Levenberg-Marquadt, constant error weights and logtan activation function

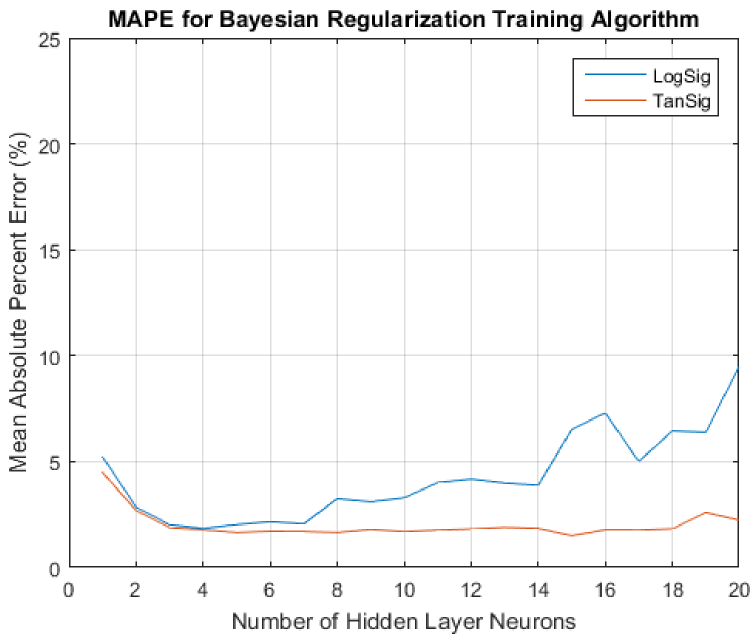


Figure 6.10: MAPE error for Bayesian regularization, constant error weights and logtan activation function

input delays and the size of the training set. The comparisons will be made based on the simulation results.

Feedback and Input Delays Figures 6.11 and 6.12 show the error results for 24 and 48-hour feedback and input delays. It is clear that it is better to use a 24-hour feedback and input delay than a 48-hour because both the average MAPE and its standard deviation are lower.

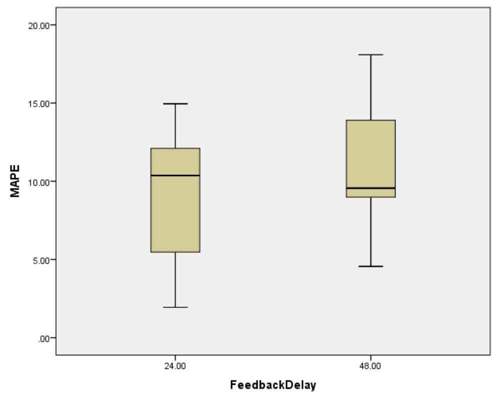


Figure 6.11: MAPE error for 24 and 48-hour feedback and input delays

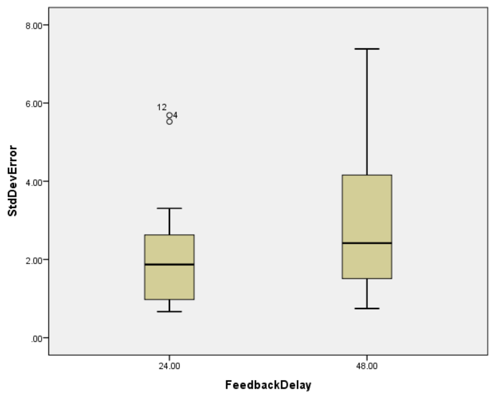


Figure 6.12: Standard Deviation of MAPE for 24 and 48-hour delays

Training Data Set Size Figures 6.13 and 6.14 show the error results for 168 (7 x 24) hour training and 336 97 x 24) hour training data sets. Although the standard deviation of the error is smaller with the largest data set, as expected, errors are smaller with the 168-hour training set.

Several observations are important at this point:

1. The Levenberg-Marquadt algorithm is far more sensitive to the number of neurons than the Bayesian regularization algorithm, and consistently gives smaller error (MAPE) across all scenarios as shown in Figures 6.15 and 6.16.

2. Training algorithm and activation function combinations work best: Levenberg Marquadt works best with Logistic Sigmoid (logsig), and Bayesian

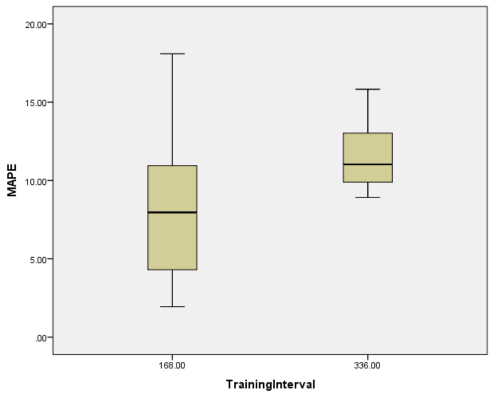


Figure 6.13: MAPE error for 168 and 336 hours training data sets

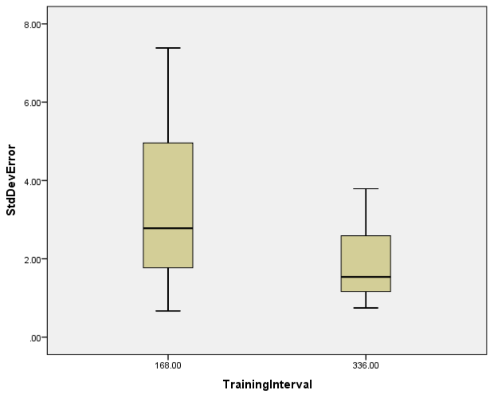


Figure 6.14: Standard Deviation of MAPE 168 and 336 hours training

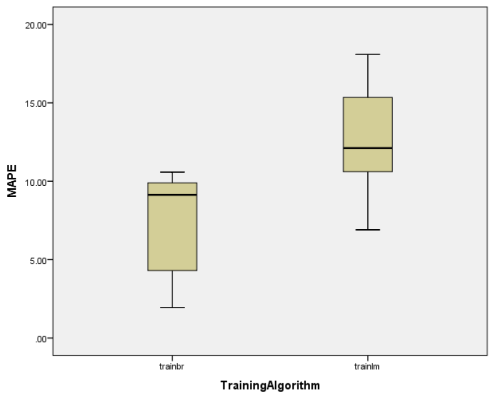


Figure 6.15: MAPE for Levenberg Marquadt and Bayesian Regularization

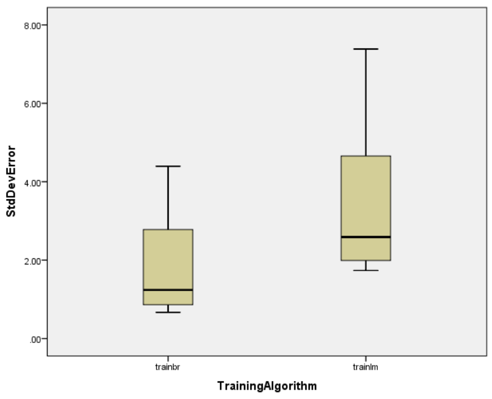


Figure 6.16: Standard Deviation of Error for Levenberg Marquadt and Bayesian Regularization Methods

regularization works best with hyperbolic tangent. See Figures 6.9 and 6.10. It is clear that this conclusion is valid only if all other parameters are kept equal.

3. Overall logistic sigmoid activation function yields lower error than hyperbolic tangent as shown in Figures 6.17 and 6.18.

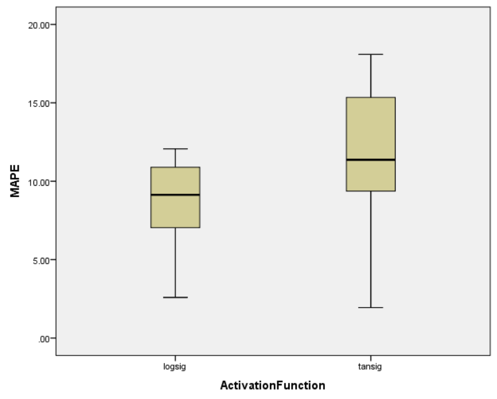


Figure 6.17: MAPE error for logarithmic sigmoid and hyperbolic tangent activation functions

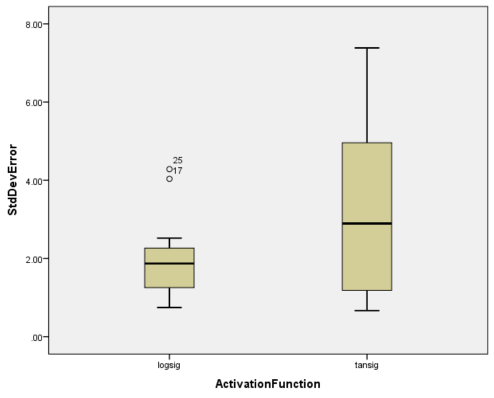


Figure 6.18: Standard Deviation of MAPE for logarithmic sigmoid and hyperbolic tangent activation functions

So the question that motivates this part of the research is: Is there a way in which I can improve the accuracy of the forecast which reduces the sensitivity of the forecast result to the number of neurons, or even the selection of activation function?

6.2.2 Experimental Results after Proposed Improvement

After applying the exponential error weight decay function, the networks were simulated again, and the results are shown in Table 6.4.

The first observation we can make, by looking at the results on Table 6.4 and Figure 6.19 is that, by using the exponential decay of the error weight function, we improve the forecasting accuracy of the NARX neural network, as long as the

Table 6.4: Simulation Results with Proposed Modification

Training Algorithm	Feedback Delay	Training Interval	Activation Function	Error Weight Function	MAPE	Std. Dev. of MAPE
Bayesian Regularization	24	7 x 24	Logistic Sigmoid	Constant	4.05	2.52
				Exponential	2.59	0.83
			Hyperbolic Tangent	Constant	1.97	0.68
				Exponential	1.94	0.67
		14 x 24	Logistic Sigmoid	Constant	10.3	1.06
				Exponential	10.2	1.22
			Hyperbolic Tangent	Constant	10.58	0.89
				Exponential	10.44	1.11
	48	7 x 24	Logistic Sigmoid	Constant	9.22	4.04
				Exponential	6.93	4.28
			Hyperbolic Tangent	Constant	9.23	4.39
				Exponential	4.56	3.04
		14 x 24	Logistic Sigmoid	Constant	8.92	0.75
				Exponential	9.05	1.29
			Hyperbolic Tangent	Constant	9.6	1.27
				Exponential	9.52	1.26
Levenberg-Marquadt	24	7 x 24	Logistic Sigmoid	Constant	6.9	1.81
				Exponential	7.16	2.06
			Hyperbolic Tangent	Constant	12.16	5.53
				Exponential	13.43	5.69
		14 x 24	Logistic Sigmoid	Constant	11.48	1.93
				Exponential	12.05	2.45
			Hyperbolic Tangent	Constant	13.98	2.73
				Exponential	14.96	3.31
	48	7 x 24	Logistic Sigmoid	Constant	8.76	1.74
				Exponential	9.74	2.04
			Hyperbolic Tangent	Constant	17.45	7.38
				Exponential	18.09	7.24
		14 x 24	Logistic Sigmoid	Constant	12.07	2.08
				Exponential	11.54	1.78
			Hyperbolic Tangent	Constant	15.74	2.75
				Exponential	15.83	3.79

training algorithm is Bayesian regularization. If the training algorithm used is Levenberg-Marquadt, there is no perceivable gain in accuracy. Also true, is that the standard deviation of the MAPE error, as shown in Figure 6.20, is also smaller if using exponential decay and Bayesian regularization.

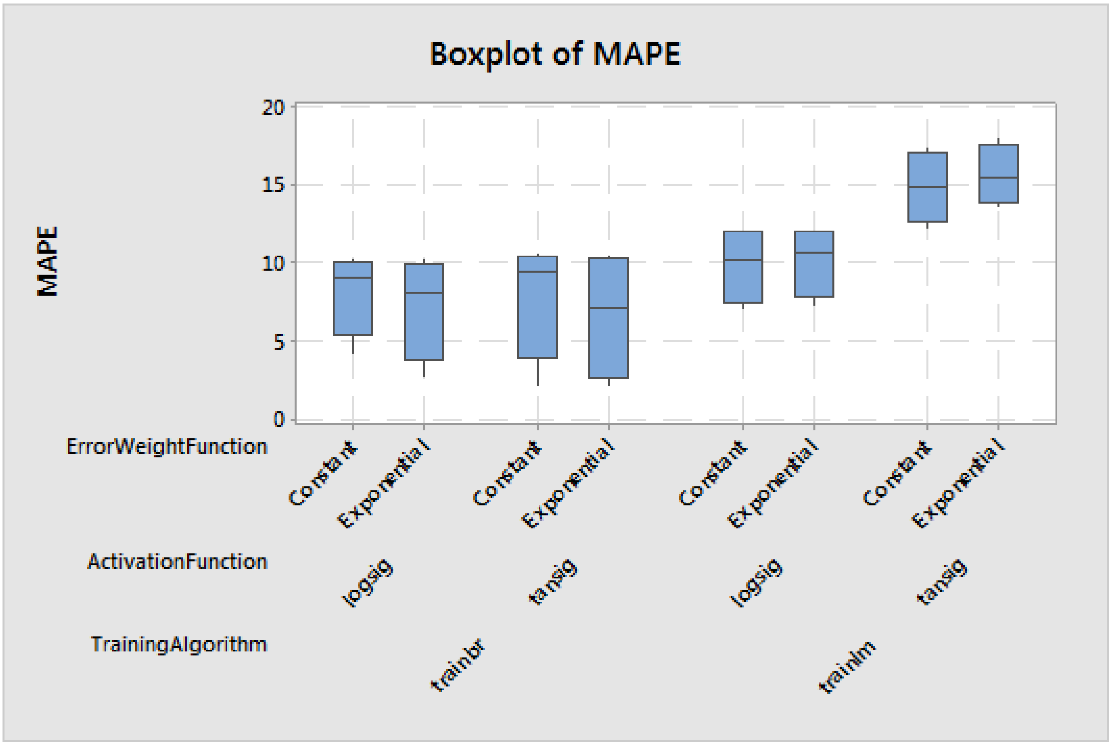


Figure 6.19: Box Plot of MAPE for Bayesian Regularization and Levenberg-Marquadt training algorithms

This behavior can also be visualized for $N = 1 \ldots 20$ neurons, as shown in Figures 6.21 and 6.22. When the Levenberg-Marquadt training algorithm is used we can see that there is very little effect on the error by using the exponential decay mask. Also notice that logarithmic sigmoid activation function yields a better error performance than hyperbolic tangent using Levenberg-Marquadt.

Figure 6.23 shows the error behavior for the combination of Bayesian regularization and logarithmic sigmoid for both constant and exponentially decaying error weights. Here we see great improvement in the error. Also, there is less ef-

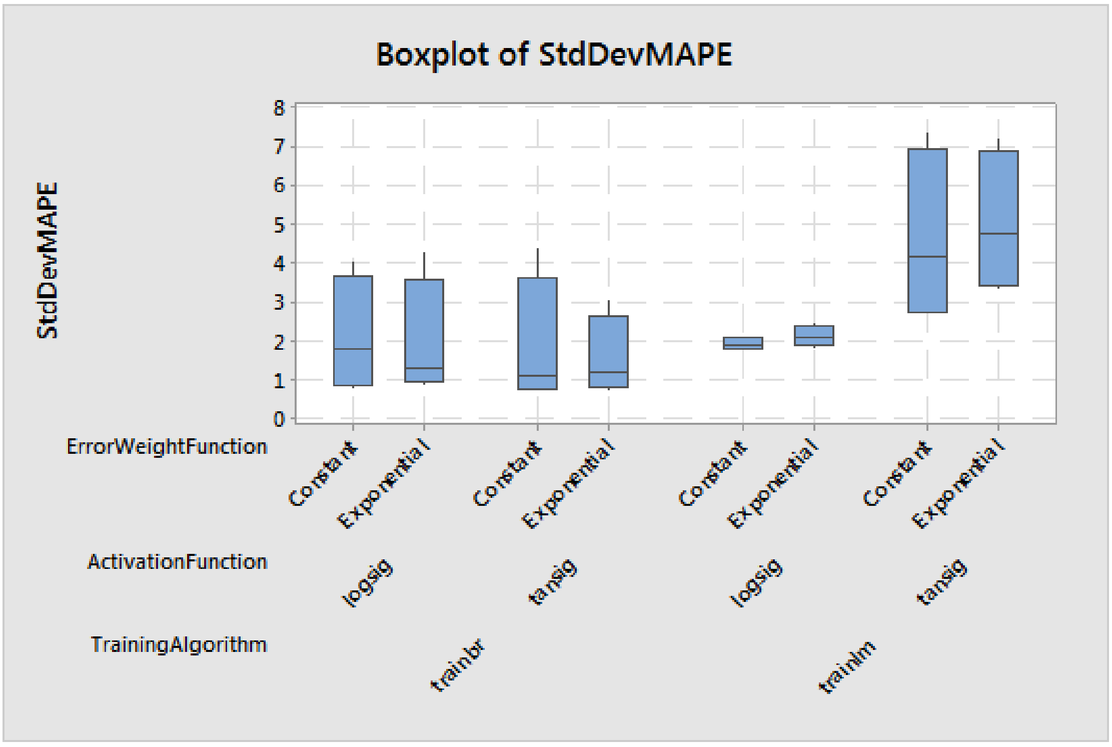


Figure 6.20: Box Plot of Standard Deviation of MAPE for Bayesian Regularization and Levenberg-Marquadt training algorithms

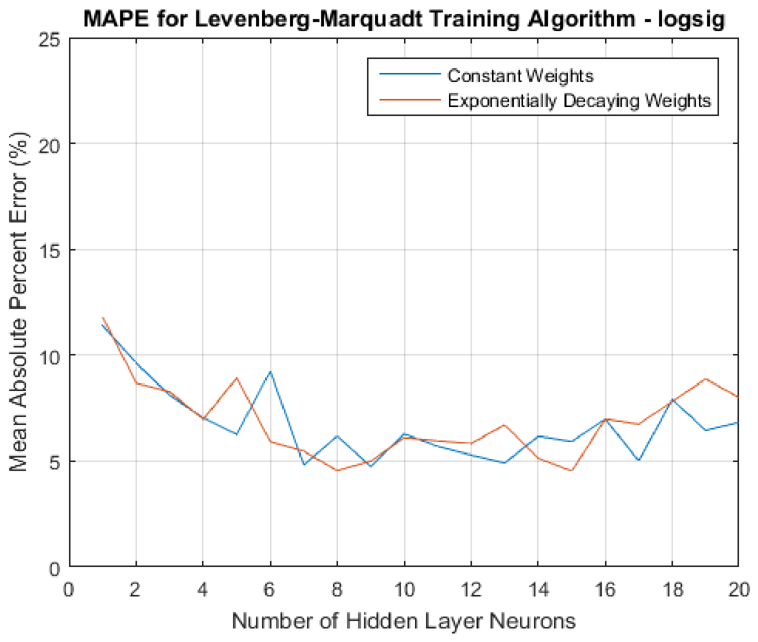


Figure 6.21: MAPE for Levenberg-Marquadt training algorithm using both constant weights and exponential weights and logarithmic sigmoid activation function

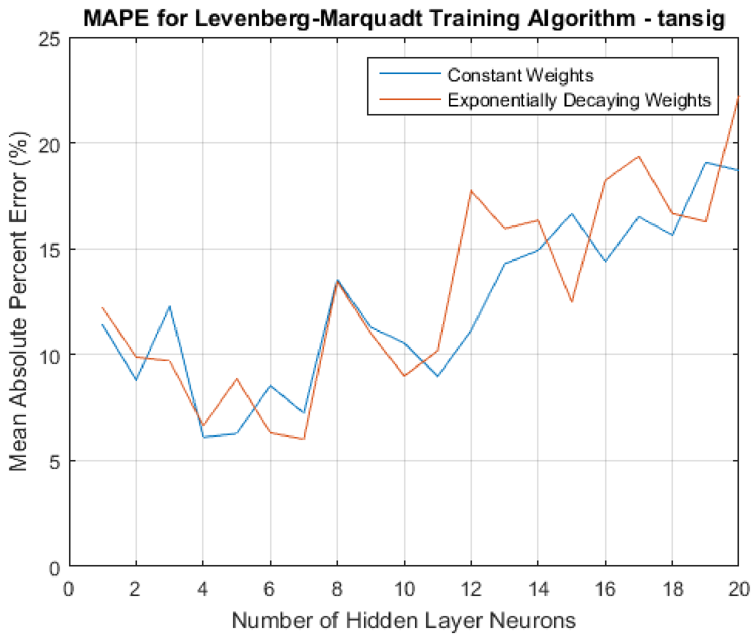


Figure 6.22: MAPE for Levenberg-Marquadt training algorithm using both constant weights and exponential weights and hyperbolic tangent activation function

fect of the number of neurons on the error throughout the range of neurons $[0, 20]$. This means that the network is more robust and accurate.

Figure 6.24 shows the error behavior for the combination of Bayesian regularization and hyperbolic tangent activation function for both constant weights and exponentially decaying error weights. The error behavior is also improved, and the resulting error is less sensitive to variation in number of neurons. Robustness is therefore improved.

If we look at all the scenarios (6400 neural networks) shown in Figure 6.25 we can clearly see what the results mean: we obtain the best results (indicated by a red arrow) when we use 24 hour feedback and input delay, 7 x 24-hour training set, hyperbolic tangent activation function, Bayesian regularization training algorithm. If we furthermore add exponentially decaying weight functions, we see an improvement in both error and standard deviation of error.

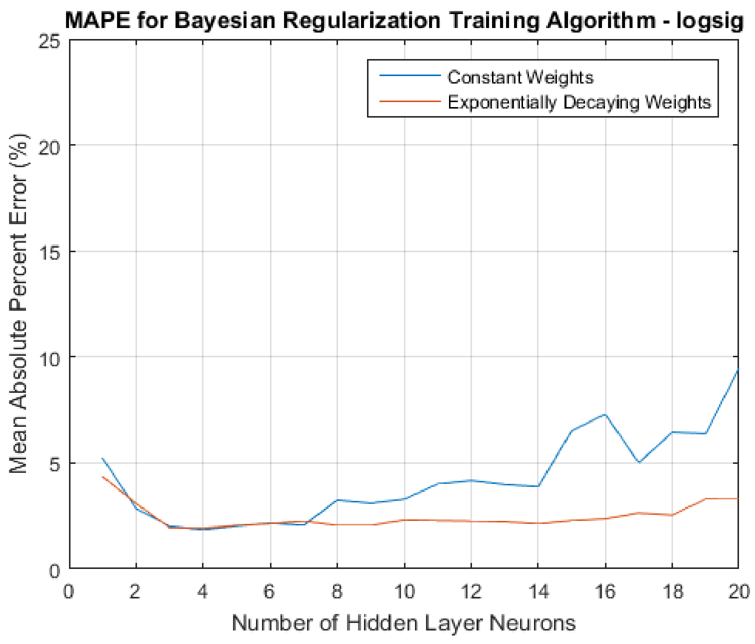


Figure 6.23: MAPE for Bayesian Regularization training algorithm using both constant weights and exponential weights and logarithmic sigmoid activation function

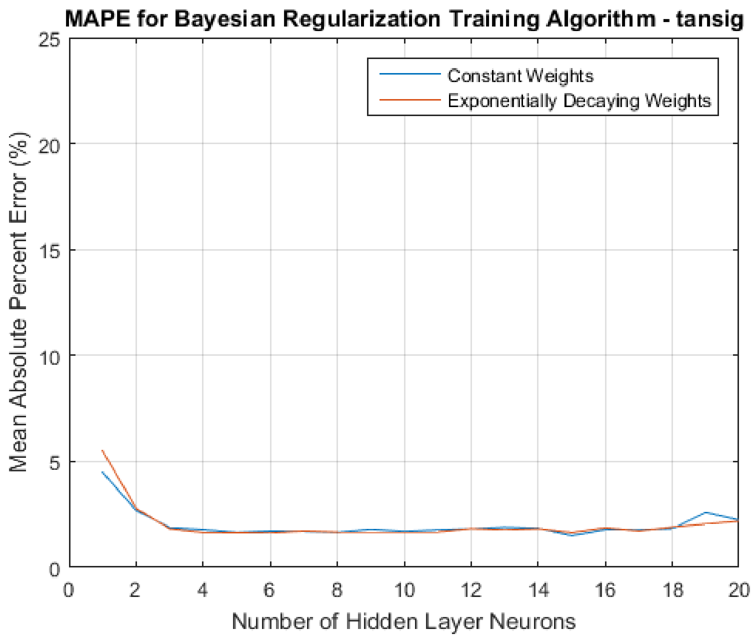


Figure 6.24: MAPE for Bayesian Regularization training algorithm using both constant weights and exponential weights and hyperbolic tangent activation function

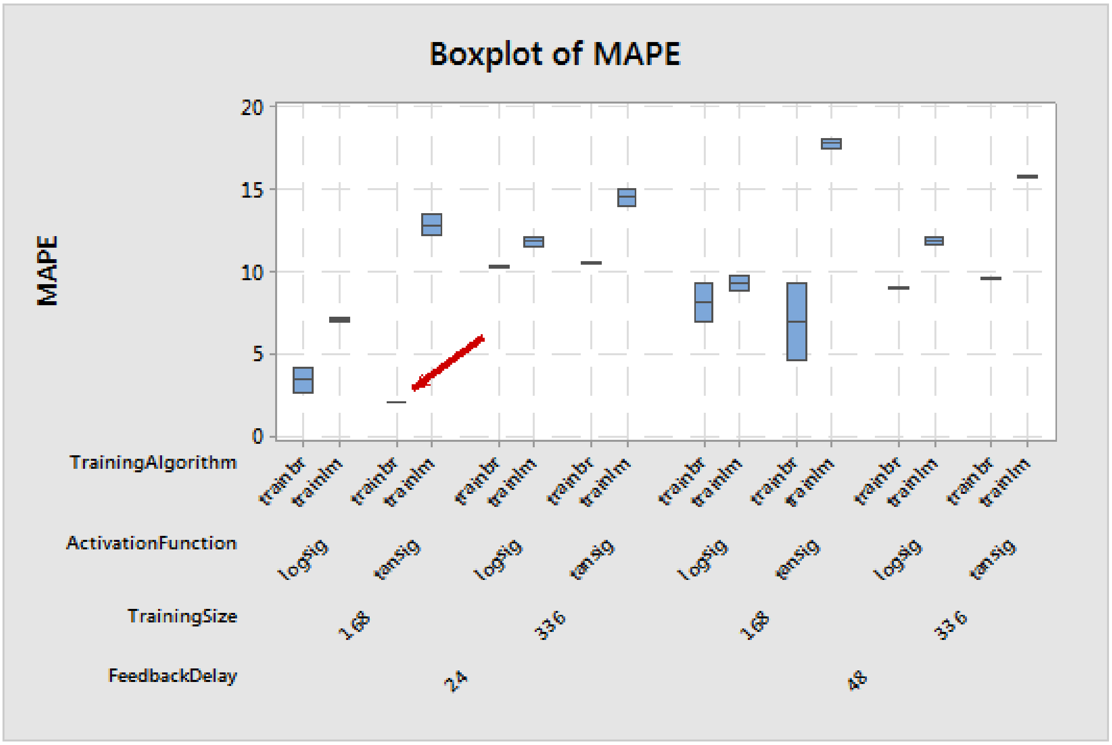


Figure 6.25: Box Plot of MAPE for all simulation results

As stated before, what we are after is an improvement in both accuracy and robustness, and it has been shown that by using the proposed method, this is accomplished. Figure 6.26 shows the combination of the error and its standard deviation. What we are looking for are points in the lowest possible values for both. It should be noted that the three points in the lowest left part of the graph were accomplished by using the exponential decay mask.

Finally, if we filter data by using only the scenarios where the feedback and input delays are 24 hours, and the training set is 7 x 24 hours, we can see the results in a different light. Figures 6.27 and 6.28 show the improvement achieved by the exponential decay mask both in error and standard deviation of error. Notice that the improvement is limited to Bayesian regularization, and does not apply to Levenberg-Marquadt training algorithm. The average error achieved is in

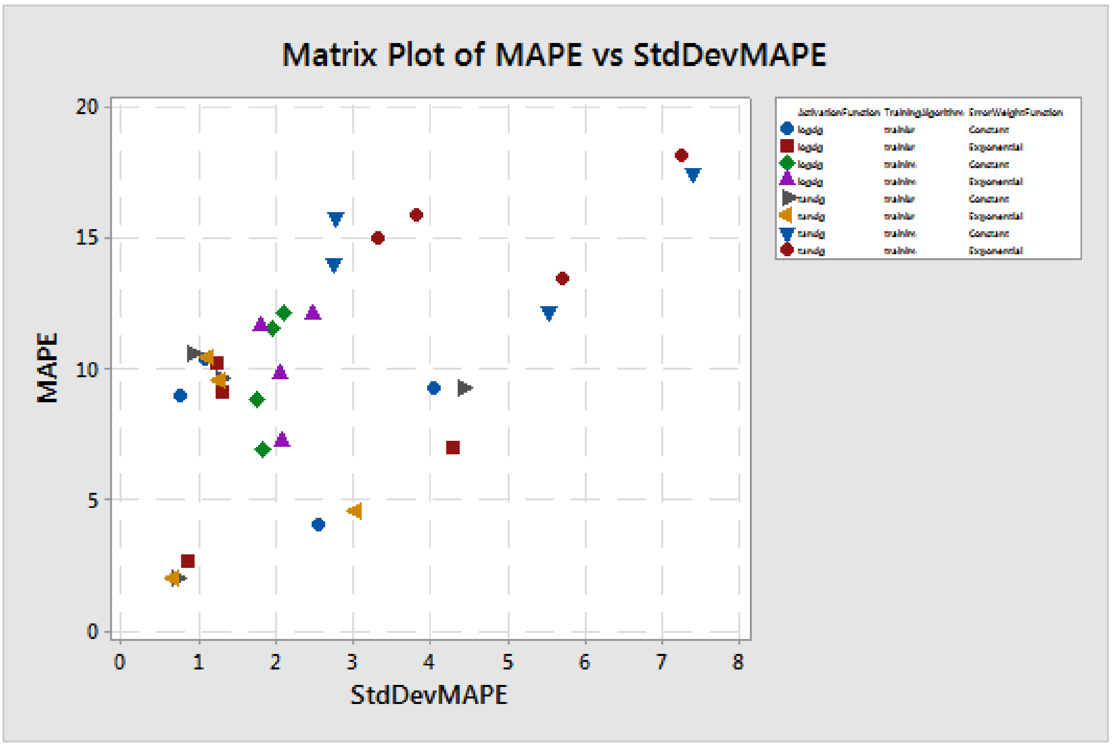


Figure 6.26: MAPE v. Standard Deviation of MAPE

Table 6.5: MAPE Error Improvements

	MAPE Error Improvements			
Error Weights	**Bayesian 24h - 168h**	**Bayesian 24h**	**Bayesian**	**BR and LM**
Constant	3.01	6.73	7.98	10.15
Exponential	2.27	6.29	6.90	9.88
Difference	0.75	0.43	1.08	0.27
Percent Diff.	24.8%	6.4%	13.5%	2.7%

the order of 2% for all the cases studied. This is an important conclusion because traditionally forecast has been done training the network with the Levenberg-Marquadt algorithm as opposed to Bayesian Regularization.

Using the correct specifications in terms of time lags for the feedback and inputs, and the correct size of the training set, the use of the proposed technique can result in an improvement of almost 25% over the use of constant error weights, as shown in Table 6.5. 13.5% improvement is due to the use of Bayesian regularization and the rest is due to the exponential decay mask.

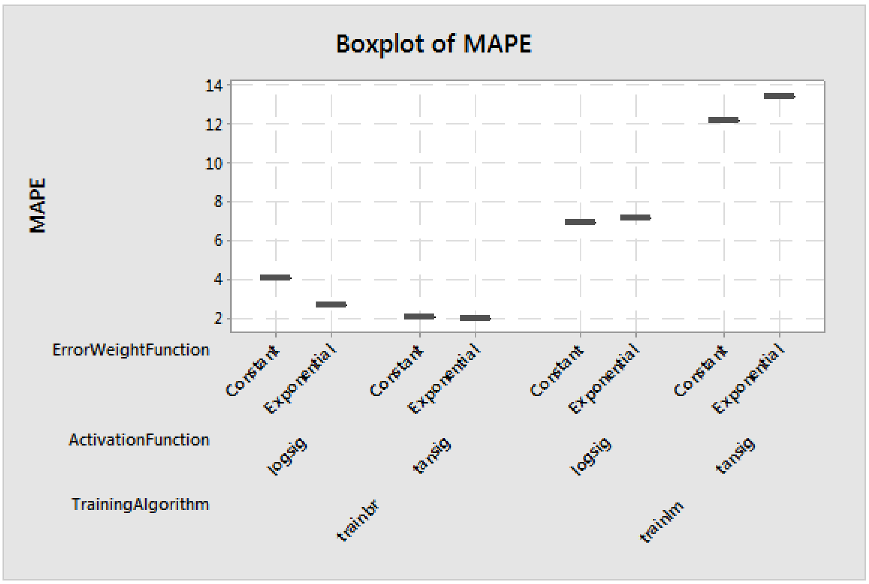


Figure 6.27: Box Plot of MAPE for 24-hour feedback and input delays and 168-hour training

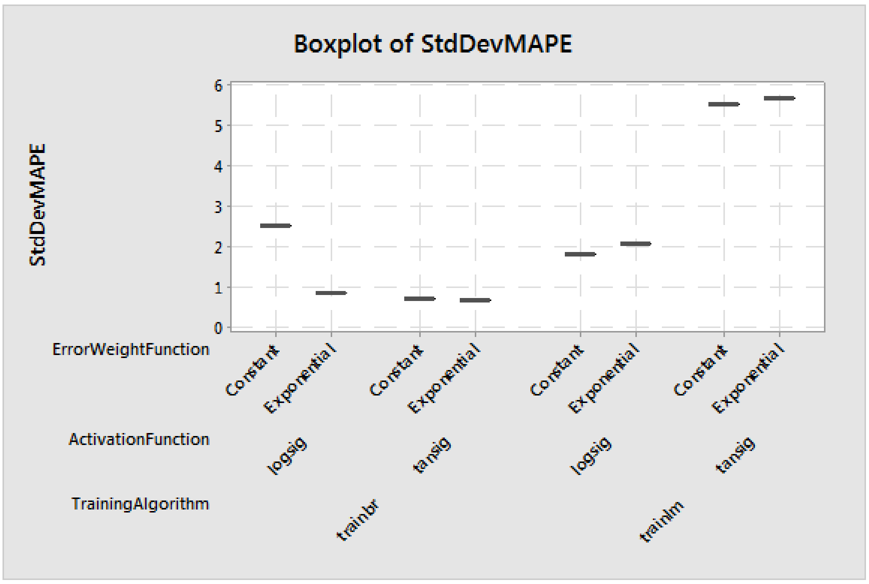


Figure 6.28: Box Plot of Standard Deviation of MAPE for 24-hour feedback and input delays and 168-hour training

6.3 Computational Efficiency

Traditional methods in ANN are computationally intensive. The number of neurons in a neural network tends to linearly increase the computational time. This section attempts to clarify the impact of running a NARX network as opposed to a more traditional neural network such as a Fitnet or a feedforward network. The algorithm presented in this work was executed with standardized data across all models in order to develop a comparison. For the NARX network two time lags in the input and two time lags in the feedback were used, and of course the other models do not contain time lags. The number of neurons were varied from 1 to 50 neurons, and the execution times for training are shown below in Figure 6.29. It is clear that adding complexity to the neural network also increases the execution time for training of said network. We see that in general terms, the execution times were double for the NARX network when compared with both the Fitnet network and the Feedforward network. However, the total execution time for 50 neurons is under 20 seconds which makes it a very acceptable performing network.

These results were run on an Intel i7 processor with 4 cores running Windows 10, and 16 Gigabytes of RAM memory.

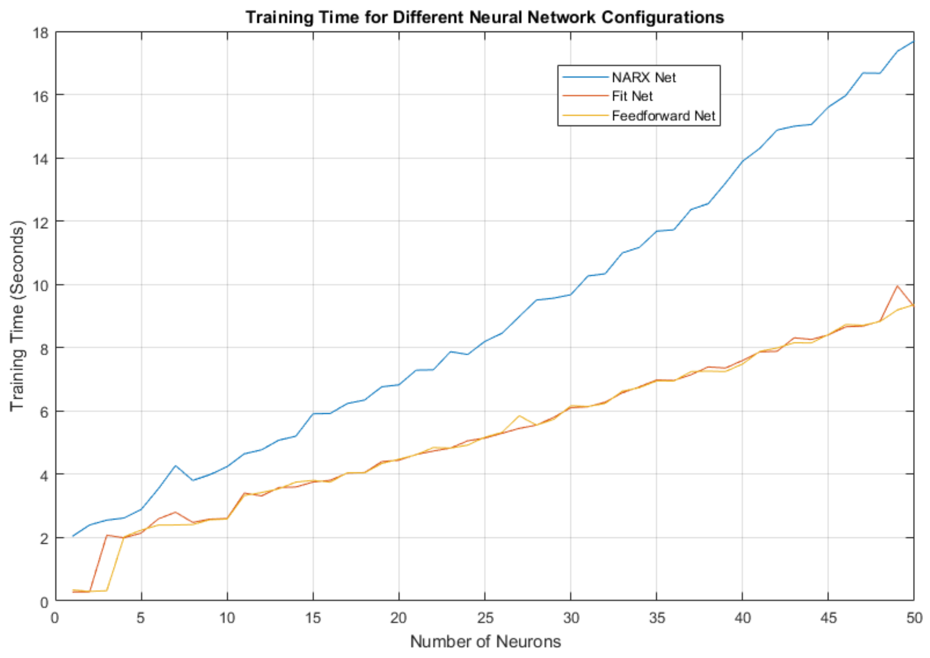


Figure 6.29: Training Time Comparison

6.4 Discussion

The algorithm was run under many different conditions and configurations. The following items have been taken into consideration:

- Different input periods (both in time and length)—this is to include and exclude the effects of holidays, working and non-working days, seasons, etc. ANN NARX work well when sufficient cyclical data are available for training.
- The number of neurons in the hidden layer in the model—simulations were run from 1 to 30 neurons in the hidden layer. Typically the forecast accuracy peaks at around 20 neurons.
- The number of delayed inputs and feedback—simulations ranging from 1 h to 48 h delays were perfored. A 24-hour time delay for inputs worked best.

- The redundancy on the input data—reinforcement of training data with the most recent available data. It was found that feeding redundant data for last week and last 24 h worked best.

An important aspect of the simulation results, which can be seen on Figure 6.4 is that there is a slight overshoot of the forecast calculated by the NARX model. The overshoot (on hour 18) corresponds to a 2.77% error in the forecast with respect with the actual value. The maximum point error by the ARMAX model was 2.3%. Typically forecast overshoots of this nature are the result of overfitting, in the case of statistical models, or having too many hidden neurons in the case of ANNs. However, as the results show, the mean of the error is smaller in the NARX model than it is in the ARMAX model. Overall, the NARX model gives a much closer forecast across the time series.

When different datasets were used for testing the model it was encountered that the NARX model, in absence of multiple inputs and without using time lags for the inputs and output feedback behaves similarly to a Fitnet or Feedforward neural network. This is an expected result since the training of a NARX without multiple exogenous inputs corresponds to a one-to-one input-target relationship which is well generalized by the use of feedforward neural networks.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

A novel implementation of a nonlinear autoregressive neural network with exogenous input (NARX) has been presented. The approach shows that an ANN can be trained in open-loop by using all of the available endogenous and exogenous inputs. Electric load has been used as the endogenous variable, and time and weather (dry bulb and dew point temperatures) are used as exogenous inputs. The network is a recursive ANN with connections between the output, hidden and input layers. The network was trained using a Levenberg–Marquardt back-propagation algorithm. Once the neural weights are calculated, the load input is disconnected, and the predicted (forecast) value of the output is fed back to the input. This isolates the network and removes the requirement for retraining for generating each instance of the output (predicted load). The accuracy of the traditional ANN network is improved to a forecast MAPE error of less than 1%. The authors have compared the proposed method with traditional statistical models ARMAX and state space, as well as the forecast calculated by a feedforward ANN with a multilayer perceptron architecture. In all three cases, the proposed

NARX architecture provides a lower MAPE error. In practical terms, this translates into savings because the forecast load is used to commit power plants for power availability, and the more accurate the forecast is, the better the operations performance achieved, thus saving energy and cost. The accuracy in the forecast is very important, especially for borderline cases in which a plant start up may be delayed for an hour or even avoided altogether given a reliable forecast. Impact on a network, such as ISO New England, where the data for this research was obtained, a network with a 19.3-GW available capacity and typical demand varying from 10 GW–18 GW, could represent savings of 0.5% of the installed capacity, i.e., 100 MW for periods of time ranging between 1 and 4 h per day. These are the typical overshoot times for forecasted loads, and their predictability offers an opportunity for real cost savings.

Forecasting error behavior using nonlinear autoregressive neural networks with exogenous inputs can be improved by implementing exponentially decaying error weights. The technique is particularly effective if the training algorithm includes Bayesian regularization, in which case the proposed solution is effective in reducing error and its standard deviation for both logarithmic sigmoid and hyperbolic tangent activation functions. Also the proposed method is effective in increasing the robustness of the NARX network to the change in number of neurons in the hidden layers. The technique gives mixed results if the Levenberg-Marquadt back propagation training algorithm is used. The proposed framework has been tested under different circumstances of training data lengths, feedback and input lags, activation functions, and training algorithms. Up to 25% improvement in error has been achieved by using the proposed framework, half of it due to the Bayesian regularization technique for training, and the other half due to the exponential error weight decay strategy. The combination of both is what gives the proposed solution the robustness to consistently generate accurate forecasts.

7.2 Model Limitations

This section analyzes the limitations of the NARX neural networks for forecasting loads.

7.2.1 Forecast Horizon

This section studies the forecast horizon: how well does the forecast perform in the 24-hour ahead forecasting period, the 48-hour ahead, and the 7-Day ahead. All the parameters in the neural network have been selected to provide an accurate 24-hour forecast.

Simulations were run for 48 hours and 7 days. Figure 7.1 shows, as expected, a degradation of the quality of the forecast after 24 hours. Figure 7.2 shows the degradation of the forecast during a 7-day period. It is interesting to notice that after 7 days, the forecast reduces it error again because of the similarities between 7-day periods.

It is clear from this analysis that the proposed NARX ANN model is accurate for the first 24-hours, but its performance degrades as time increases, as expected. Table 7.1 shows the performance degradation as the forecasting window becomes longer as demonstrated by the increasing error and decreasing Pearson R correlation coefficient between the actual value and the predicted value of the load. The slope (greater than 1) indicates a slight inclination of the forecast to overestimate the load value, and that inclination increases as the forecasting window expands.

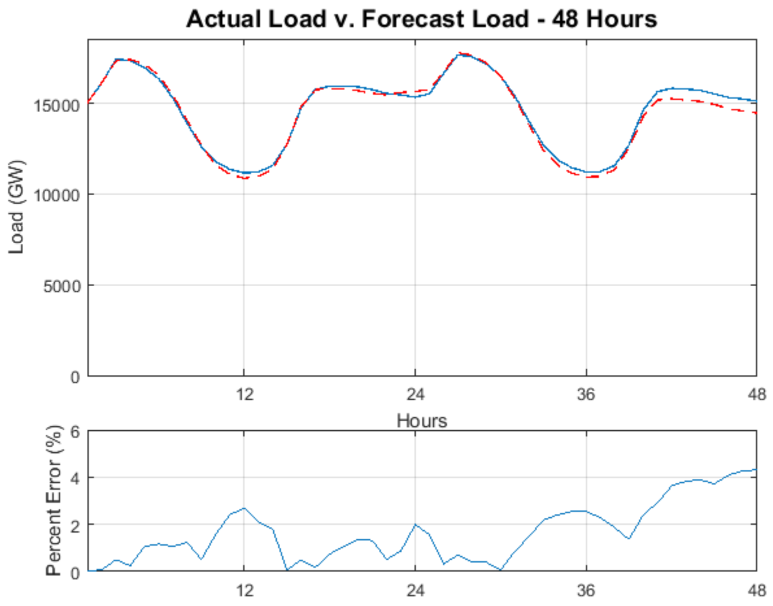


Figure 7.1: 48-Hour Ahead Forecast

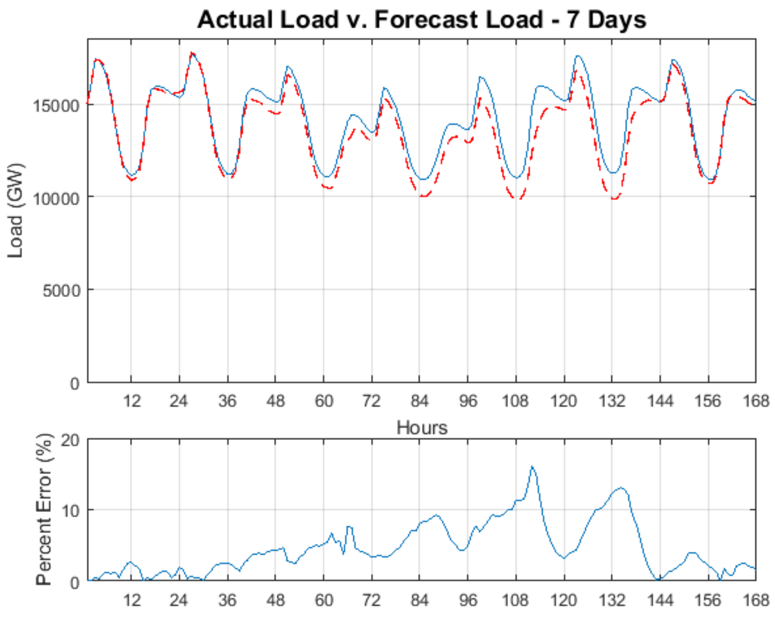


Figure 7.2: 7-Day Ahead Hourly Forecast

Table 7.1: Forecasting Error for Longer Contiguous Forecast Periods

Forecasting Period	MAPE Error (%)	Pearson R	Slope
24-hour Ahead	1.04%	0.99749	1.03858
48-hour Ahead	1.65%	0.99295	1.03035
7-Day Hourly Ahead	4.61%	0.97097	1.04578

7.2.2 Broken Forecasting Period

Now, we are going to explore the ability to forecast with a NARX ANN model a period not immediately following the end of the training data set. If we break the time series, at least we must ensure that the training data set ends on a similar date (of the week) so that there is continuity on the type of data expected to be forecast.

The ANN was trained with data from 2011 and used to forecast exactly 52 weeks later starting on the same day of the week and time as the end of the training data. The results of the simulation are shown in Figures 7.3 and 7.4.

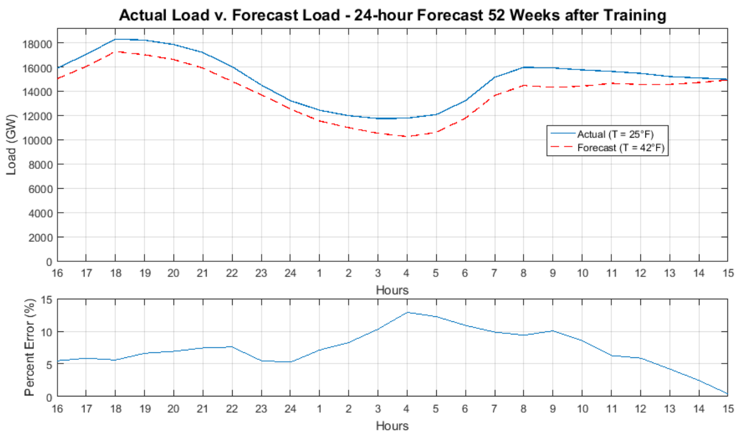


Figure 7.3: 1-Year Ahead 24-Hour Forecast

Table 7.2 shows the forecasting error results for estimating values long after the training period. The 1-Year Ahead 24-hour forecast has a MAPE of 7.30% which shows a clear degradation of the generalization ability of the NARX network for long periods of time after training has ceased. The 1-Year Ahead 7-Day hourly

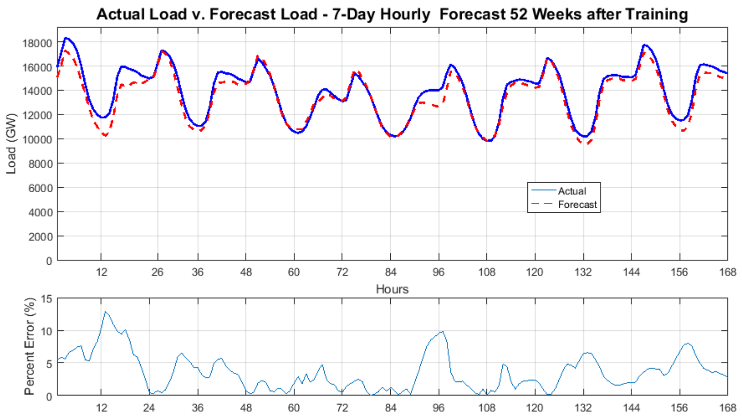


Figure 7.4: 1-Year Ahead 7-Day Hourly Forecast

forecast has a 3.61% MAPE error. This is an indication that year-on-year load is relatively similar on longer periods of time (7-days) than on shorter periods (1-day). Also, on longer periods of time (7-days), the network exhibits a clear inclination to underestimate the load value, as indicated by the slope (0.94644) of the predicted vs. actual linear regression line. This is the result of the increasing trend in the load which cannot be captured by the model because of the time gap without training data.

Table 7.2: Forecasting Error for Longer Non-Contiguous Forecast Periods

Forecasting Period	**MAPE Error (%)**	**Pearson R**	**Slope**
24-hour Ahead	1.04%	0.99749	1.03858
1-Year Ahead 24-hour	7.30%	0.98278	1.00294
1-Year Ahead 7-Day	3.61%	0.97587	0.94644

7.2.3 Loss of Endogenous Input Signals

Non-linear Autoregressive Neural Networks with Exogenous Inputs have an advantage over traditional forecasting methods, in which the loss of endogenous input signals do not carry a loss of forecasting ability. In the NARX model for short-term load forecasting, the actual past load is the endogenous variable. If for some reason that signal's value is not known for a period of time, the network can obtain the estimated value from its own output until the actual values are available without having to retrain the neural network.

A simulation has been carried out where the network is switched from open-loop (used for training) to closed loop (used for forecasting), a forecast for the next 12 hours is created and suddenly the actual values of the load for the past 12 hours are available. There is no need to retrain the network, instead, the loop can be open to take the 12-hour data points (actual) and continue forecasting until those values are no longer available. At that time, the network can be used in closed-loop to provide the next step prediction, and it continues to produce predicted values until the actual historical values of the load are recovered. Figure 7.5 shows the behavior of the first 12 hours in closed-loop mode, and then the forecast produced by the open-loop network for the remaining 24 hours. Again, no retraining of the network is necessary.

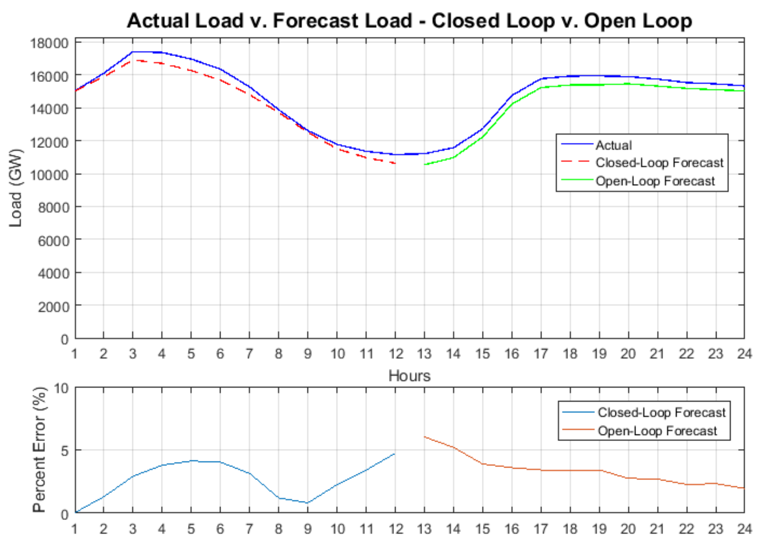


Figure 7.5: 24 Hour Forecast with Data Interruption at 12 Hours

This type of switching is useful when using sensors for forecasting. The loss of a sensor signal does not mean that the system shuts down, but instead, by switching to the open-loop state, the model continues providing accurate forecast values.

7.3 Future Work

Artificial Neural Networks have proven to be robust methods for providing forecasts for time series data. Together with statistical methods they establish a framework for time series data forecasting that comprises an arsenal of tools and techniques that must be adapted for every situation. The fact that ANNs are model-free data-driven methods are a welcome addition to statistical methods when models are highly non-linear or time variant. However, one of the drawbacks of ANNs is that the fact that they are data-driven reduces the ability of the researcher to use analytical tools to study the solutions. The final result of a neural network is a set of weights and biases that are not associated with particular features of a model (as opposed to parameters of a standard model), and therefore inferring into the significance of a result is pointless. Furthermore, it is yet unknown how to determine which ANN architecture is best for a particular data set. What this work, and the work of many other researchers, addresses is merely empirical and therefore limited in its application. But each step in the right direction makes us get closer to understanding under which circumstances certain models behave in a more favorable manner.

There is a lot of room for improvement in the general topic of ANN forecasting for time series. We can list some of the general topics by going over the different components of an ANN model:

Network Topology More models on how to determine the optimal network architecture are needed. The number of input nodes, hidden layers, hidden nodes and output nodes and the manner in which they are all interconnected are still determined by trial and error, and in spite of many efforts to find a method to determine an optimal configuration, none have proven all-encompassing. It would be necessary to develop an analytical model for network architecture so that it would be easy to identify the type of network and configuration to use for each type of data input.

Activation Functions We have seen in the present research that activation function has a strong influence on the outcome of the ANN. But it also depends on the training method and the way in which the input is defined (number of time lags, etc.). A more analytical study of this topic is necessary. Certain activation functions work better with certain training methods and certain time lags. It would be highly desirable to have, not a heuristic, but analytic method of determining which activation function to use.

Training Algorithms As shown by this study, the training algorithm used has huge implications on the result of the forecast. Not all training methods work well under all circumstances. More research needs to be done on the impact of the training methods. Some empirical results point to methods that favor smaller weights, but their performance may be lower. Is there an optimal balance between convergence and accuracy? The objective is to capture the attributes of a time series without over-reacting to its particular features. Given data features, is there a way to determine which training method is going to perform better than the others?

Data Normalization Current data preparation methods squash the inputs so that none of the variables has preference in the values of their weights. Of the normalization methods (external, along-channel, across-channel and mixed-channel) which one provides the best result? Is that result best overall or it just works for a type of data? Typically external normalization is used for time series data, but more study into the different methods is necessary. Should input data normalization and output data normalization be somehow correlated? Most research discuss data normalization as independent, but it would be interesting to know if by tying input and output data normalization forecasting data generalization results could improve.

Size of Training Data As discussed in this study, the size of the training data set has a great impact on the forecasting accuracy. It also mainly impacts the performance of the forecasting model: larger data sets are harder to train due to the large amount of computation required. What is the smallest sample needed to properly train a network? Research in this topic is limited to performance, but data generalization needs to be addressed.

Performance Indicators Not all errors have the same nature. The selection of performance indicators has been studied, but the combination of these indicators with network topologies, training algorithms and activation functions needs to be further studied.

Bibliography

[1] B. Satish, K. S. Swarup, S. Srinivas, and A. H. Rao, "Effect of temperature on short term load forecasting using an integrated ann," *Electric Power Systems Research*, vol. 72, no. 1, pp. 95–101, 2004.

[2] T. Nengling, J. Stenzel, and W. Hongxiao, "Techniques of applying wavelet transform into combined model for short-term load forecasting," *Electric Power Systems Research*, vol. 76, no. 6âĂŞ7, pp. 525–533, 2006.

[3] X. Liu, B. Ang, and T. Goh, "Forecasting of electricity consumption: a comparison between an econometric model and a neural network model," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on.* IEEE, 1991, Conference Proceedings, pp. 1254–1259.

[4] R. E. Abdel-Aal, "Univariate modeling and forecasting of monthly energy demand time series using abductive and neural networks," *Computers and Industrial Engineering*, vol. 54, no. 4, pp. 903–917, 2008.

[5] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: a review and evaluation," *Power Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 44–55, 2001.

[6] S. Tzafestas and E. Tzafestas, "Computational intelligence techniques for short-term electric load forecasting," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 31, no. 1-3, pp. 7–68, 2001.

[7] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *European Journal of Operational Research*, vol. 199, no. 3, pp. 902–907, 2009.

[8] S. Rahman and I. Drezga, "Identification of a standard for comparing short-term load forecasting techniques," *Electric Power Systems Research*, vol. 25, no. 3, pp. 149–158, 1992.

[9] M. E. El-Hawary and G. A. N. Mbamalu, "Short-term power system load forecasting using the iteratively reweighted least squares algorithm," *Electric Power Systems Research*, vol. 19, no. 1, pp. 11–22, 1990.

[10] B. Wang, N.-l. Tai, H.-q. Zhai, J. Ye, J.-d. Zhu, and L.-b. Qi, "A new armax model based on evolutionary algorithm and particle swarm optimization for short-term load forecasting," *Electric Power Systems Research*, vol. 78, no. 10, pp. 1679–1685, 2008.

[11] S. S. Pappas, L. Ekonomou, P. Karampelas, D. C. Karamousantas, S. K. Katsikas, G. E. Chatzarakis, and P. D. Skafidas, "Electricity demand load forecasting of the hellenic power system using an arma model," *Electric Power Systems Research*, vol. 80, no. 3, pp. 256–264, 2010.

[12] C. Kang, X. Cheng, Q. Xia, Y. Huang, and F. Gao, "Novel approach considering load-relative factors in short-term load forecasting," *Electric Power Systems Research*, vol. 70, no. 2, pp. 99–107, 2004.

[13] T. N. Goh, S. S. Choi, C. H. Tan, and K. C. Tan, "A comparative study of short-term forecasting of energy and peak power demand," *Electric Power Systems Research*, vol. 5, no. 1, pp. 63–71, 1982.

[14] J.-F. Chen, W.-M. Wang, and C.-M. Huang, "Analysis of an adaptive time-series autoregressive moving-average (arma) model for short-term load forecasting," *Electric Power Systems Research*, vol. 34, no. 3, pp. 187–196, 1995.

[15] E. Almeshaiei and H. Soltan, "A methodology for electric power load forecasting," *Alexandria Engineering Journal*, vol. 50, no. 2, pp. 137–144, 2011.

[16] K. Metaxiotis, A. Kagiannas, D. Askounis, and J. Psarras, "Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher," *Energy Conversion and Management*, vol. 44, no. 9, pp. 1525–1534, 2003.

[17] G. Zhang, E. B. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[18] S. A. Kalogirou, "Applications of artificial neural-networks for energy systems," *Applied Energy*, vol. 67, no. 1âĂŞ2, pp. 17–35, 2000.

[19] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, 1990.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[21] H. White, "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks*, vol. 3, no. 5, pp. 535–549, 1990.

[22] T. TerÃďsvirta, “Forecasting economic variables with nonlinear models,” *Handbook of economic forecasting*, vol. 1, pp. 413–457, 2006.

[23] H. White, “Approximate nonlinear forecasting methods,” *Handbook of economic forecasting*, vol. 1, pp. 459–512, 2006.

[24] M. Ghayekhloo, M. B. Menhaj, and M. Ghofrani, “A hybrid short-term load forecasting with a new data preprocessing framework,” *Electric Power Systems Research*, vol. 119, pp. 138–148, 2015.

[25] J. Buitrago, A. Abdulaal, and S. Asfour, “Electric load pattern classification using parameter estimation, clustering and artificial neural networks,” *International Journal of Power and Energy Systems*, vol. 35, no. 4, pp. 167–174, 2015.

[26] A. Abdulaal, J. Buitrago, and S. Asfour, “Electric load pattern classification for demand-side management planning: A hybrid approach,” in *Software Engineering and Applications: Advances in Power and Energy Systems*, A. Press, Ed., vol. 831-012. ACTA Press, 2015, Conference Proceedings.

[27] M. Lopez, S. Valero, C. Senabre, J. Aparicio, and A. Gabaldon, “Application of som neural networks to short-term load forecasting: The spanish electricity market case study,” *Electric Power Systems Research*, vol. 91, pp. 18–27, 2012.

[28] R.-H. Liang and Y.-Y. Hsu, “A hybrid artificial neural networkâĂŤdifferential dynamic programming approach for short-term hydro scheduling,” *Electric Power Systems Research*, vol. 33, no. 2, pp. 77–86, 1995.

[29] A. Khotanzad, R. Afkhami-Rohani, T.-L. Lu, A. Abaye, M. Davis, and D. J. Maratukulam, “Annstlf - a neural-network-based electric load forecasting system. (artificial neural-network short-term load forecaster)(special issue on everyday applications of neural networks),” *IEEE Transactions on Neural Networks*, vol. 8, no. 4, p. 835, 1997.

[30] Y. Shimakura, Y. Fujisawa, Y. Maeda, R. Makino, Y. Kishi, M. Ono, J.-Y. Fann, and N. Fukusima, “Short-term load forecasting using an artificial neural network,” in *Neural Networks to Power Systems, 1993. ANNPS’93., Proceedings of the Second International Forum on Applications of.* IEEE, 1993, Conference Proceedings, pp. 233–238.

[31] A. Andalib and F. Atry, “Multi-step ahead forecasts for electricity prices using narx: A new approach, a critical analysis of one-step ahead forecasts,” *Energy Conversion and Management*, vol. 50, no. 3, pp. 739–747, 2009.

[32] M. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, 2015.

[33] A. S. Alfuhaid, M. A. El-Sayed, and M. S. Mahmoud, "Cascaded artificial neural networks for short-term load forecasting," *Power Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1524–1529, 1997.

[34] C. Bennett, R. A. Stewart, and J. Lu, "Autoregressive with exogenous variables and neural network short-term load forecast models for residential low voltage distribution networks," *Energies*, vol. 7, no. 5, pp. 2938–2960, 2014.

[35] M. Bilgic, C. Girep, S. Aslanoglu, and M. Aydinalp-Koksal, "Forecasting turkey's short term hourly load with artificial neural networks," in *Developments in Power System Protection (DPSP 2010). Managing the Change, 10th IET International Conference on.* IET, 2010, Conference Proceedings, pp. 1–5.

[36] N. Kandil, R. Wamkeue, M. Saad, and S. Georges, "An efficient approach for short term load forecasting using artificial neural networks," *International Journal of Electrical Power and Energy Systems*, vol. 28, no. 8, pp. 525–530, 2006.

[37] T. Bugwan and R. T. A. King, "Short term electrical load forecasting for mauritius using artificial neural networks," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on.* IEEE, 2008, Conference Proceedings, pp. 3668–3673.

[38] W. Charytoniuk and M. S. Chen, "Very short-term load forecasting using artificial neural networks," *Power Systems, IEEE Transactions on*, vol. 15, no. 1, pp. 263–268, 2000.

[39] H. Gooi, C. Teo, L. Chin, S. Ang, and E. Khor, "Adaptive short-term load forecasting using artificial neural networks," *The 1993 IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering (TENCON '93). Part 2 (of 5), Beijing, China, 10/19-21/93*, 1993.

[40] C. Y. Tee, J. B. Cardell, and G. W. Ellis, "Short-term load forecasting using artificial neural networks," in *North American Power Symposium (NAPS), 2009.* IEEE, 2009, Conference Proceedings, pp. 1–6.

[41] M. H. H. Harun, M. M. Othman, and I. Musirin, "Short term load forecasting (stlf) using artificial neural network based multiple lags and stationary time series," in *Power Engineering and Optimization Conference (PEOCO), 2010 4th International.* IEEE, 2010, Conference Proceedings, pp. 363–370.

[42] L. Hernandez, C. Baladron Zorita, J. M. Aguiar Perez, B. Carro Martinez, A. Sanchez Esguevillas, and J. Lloret, “Short- term load forecasting for microgrids based on artificial neural networks,” *Energies*, 2013.

[43] L. Hernandez, C. Baladron, J. M. Aguiar, L. Calavia, B. Carro, A. Sanchez-Esguevillas, J. Sanjuan, l. Gonzalez, and J. Lloret, “Improved short-term load forecasting based on two-stage predictions with artificial neural networks in a microgrid environment,” *Energies*, vol. 6, no. 9, pp. 4489–4507, 2013.

[44] K. Kalaitzakis, G. S. Stavrakakis, and E. M. Anagnostakis, “Short-term load forecasting based on artificial neural networks parallel implementation,” *Electric Power Systems Research*, vol. 63, no. 3, pp. 185–196, 2002.

[45] S. Kiartzis, C. Zoumas, A. Bakirtzis, and V. Petridis, “Data pre-processing for short-term load forecasting in an autonomous power system using artificial neural networks,” in *Electronics, Circuits, and Systems, 1996. ICECS’96., Proceedings of the Third IEEE International Conference on*, vol. 2. IEEE, 1996, Conference Proceedings, pp. 1021–1024.

[46] S. J. Kiartzis, A. G. Bakirtzis, and V. Petridis, “Short-term load forecasting using neural networks,” *Electric Power Systems Research*, vol. 33, no. 1, pp. 1–6, 1995.

[47] T. Matsumoto, S. Kitamura, Y. Ueki, and T. Matsui, “Short-term load forecasting by artificial neural networks using individual and collective data of preceding years,” in *Neural Networks to Power Systems, 1993. ANNPS’93., Proceedings of the Second International Forum on Applications of.* IEEE, 1993, Conference Proceedings, pp. 245–250.

[48] N. S. Moharari and A. S. Debs, “An artificial neural network based short term load forecasting with special tuning for weekends and seasonal changes,” in *Neural Networks to Power Systems, 1993. ANNPS’93., Proceedings of the Second International Forum on Applications of.* IEEE, 1993, Conference Proceedings, pp. 279–283.

[49] A. Papalexopoulos, S. Hao, and T.-M. Peng, “Short-term system load forecasting using an artificial neural network,” in *Neural Networks to Power Systems, 1993. ANNPS’93., Proceedings of the Second International Forum on Applications of.* IEEE, 1993, Conference Proceedings, pp. 239–244.

[50] K. Reinschmidt and B. Ling, “Artificial neural networks in short term load forecasting,” in *Control Applications, 1995., Proceedings of the 4th IEEE Conference on.* IEEE, 1995, Conference Proceedings, pp. 209–214.

[51] J. P. Santos, M. A. Gomes, and A. J. Pires, "Next hour load forecast in medium voltage electricity distribution," *International Journal of Energy Sector Management*, vol. 2, no. 3, pp. 439–448, 2008.

[52] P. J. Santos, A. G. Martins, and A. J. Pires, "On the use of reactive power as an endogenous variable in short-term load forecasting," *International Journal of Energy Research*, vol. 27, no. 5, pp. 513–29, 2003.

[53] S. Zhang, J. Lian, Z. Zhao, H. Xu, and J. Liu, "Grouping model application on artificial neural networks for short-term load forecasting," in *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on.* IEEE, 2008, Conference Proceedings, pp. 6203–6206.

[54] A. Sinha, "Short term load forecasting using artificial neural networks," in *Industrial Technology 2000. Proceedings of IEEE International Conference on*, vol. 1. IEEE, 2000, Conference Proceedings, pp. 548–553.

[55] A. P. A. da Silva, V. H. Ferreira, and R. M. G. Velasquez, "Input space to neural network based load forecasters," *International Journal of Forecasting*, vol. 24, no. 4, pp. 616–29, 2008.

[56] V. H. Ferreira and A. P. Alves da Silva, "Toward estimating autonomous neural network-based electric load forecasters," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1554–1562, 2007.

[57] P. J. Santos, A. G. Martins, and A. J. Pires, "Designing the input vector to ann-based models for short-term load forecast in electricity distribution systems," *International Journal of Electrical Power and Energy Systems*, vol. 29, no. 4, pp. 338–347, 2007.

[58] L. C. M. de Andrade, M. Oleskovicz, A. Q. Santos, D. V. Coury, and R. A. S. Fernandes, "Very short-term load forecasting based on narx recurrent neural networks," in *2014 IEEE PES General Meeting Conference and Exposition.* IEEE, 2014, Conference Proceedings, pp. 1–5.

[59] C. Hanjie, D. Yijun, and J. N. Jiang, "Weather sensitive short-term load forecasting using knowledge-based arx models," in *IEEE Power Engineering Society General Meeting, 2005*, 2005, Conference Proceedings, pp. 190–196 Vol. 1.

[60] S. Fattaheian, A. Fereidunian, H. Gholami, and H. Lesani, "Hour-ahead demand forecasting in smart grid using support vector regression (svr)," *International Transactions on Electrical Energy Systems*, vol. 24, no. 12, pp. 1650–1663, 2014.

[61] M. Hayati, “Short term load forecasting using artificial neural networks for the west of iran,” *Journal of Applied Sciences*, vol. 7, no. 12, pp. 1582–1588, 2007.

[62] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Artificial neural networks for short-term load forecasting in microgrids environment,” *Energy*, vol. 75, pp. 252–264, 2014.

[63] Y.-Y. Hsu and C.-C. Yang, “Design of artificial neural networks for short-term load forecasting. part ii. multilayer feedforward networks for peak load and valley load forecasting,” *IEE Proceedings C: Generation Transmission and Distribution*, vol. 138, no. 5, pp. 414–418, 1991.

[64] ——, “Design of artificial neural networks for short-term load forecasting. part i. self-organising feature maps for day type identification,” *IEE Proceedings C: Generation Transmission and Distribution*, vol. 138, no. 5, pp. 407–413, 1991.

[65] F. J. Marin, F. Garcia-Lagos, G. Joya, and F. Sandoval, “Global model for short-term load forecasting using artificial neural networks,” *IEE Proceedings: Generation, Transmission and Distribution*, vol. 149, no. 2, pp. 121–125, 2002.

[66] A. Badri, Z. Ameli, and A. M. Birjandi, “Application of artificial neural networks and fuzzy logic methods for short term load forecasting,” *Energy Procedia*, vol. 14, pp. 1883–1888, 2012.

[67] A. Khosravi, S. Nahavandi, D. Creighton, and D. Srinivasan, “Interval type-2 fuzzy logic systems for load forecasting: A comparative study,” *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1274–1282, 2012.

[68] K. H. Kim, “Implementation of hybrid short-term load forecasting system using artificial neural networks and fuzzy expert systems,” *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1534–1539, 1995.

[69] T. S. Mahmoud, D. Habibi, M. Y. Hassan, and O. Bass, “Modelling self-optimised short term load forecasting for medium voltage loads using tunning fuzzy systems and artificial neural networks,” *Energy Conversion and Management*, vol. 106, pp. 1396–1408, 2015.

[70] P. Santos, S. Rafael, P. Lobato, and A. Pires, “A stlf in distribution systems-a short comparative study between anfis neuro-fuzzy and ann approaches-part i,” in *2009 International Conference on Power Engineering, Energy and Electrical Drives.* IEEE, 2009, Conference Proceedings, pp. 661–665.

[71] S. Rafael, P. Santos, P. Lobato, and A. Pires, “A stlf in distribution systems-a short comparative study between anfis neuro-fuzzy and ann approaches-part ii,” in *2009 International Conference on Power Engineering, Energy and Electrical Drives*, 2009, Conference Proceedings.

[72] D. Srinivasan, “Evolving artificial neural networks for short term load forecasting,” *Neurocomputing*, vol. 23, no. 1, pp. 265–276, 1998.

[73] P. Subbaraj and V. Rajasekaran, “Short term hourly load forecasting using combined artificial neural networks,” in *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, vol. 1. IEEE, 2007, Conference Proceedings, pp. 155–163.

[74] C. Sun, J. Song, L. Li, and P. Ju, “Implementation of hybrid short-term load forecasting system with analysis of temperature sensitivities,” *Soft Computing*, vol. 12, no. 7, pp. 633–638, 2008.

[75] H.-T. Yang and C.-M. Huang, “New short-term load forecasting approach using self-organizing fuzzy armax models,” *IEEE Transactions on Power Systems*, vol. 13, no. 1, pp. 217–225, 1998.

[76] O. A. S. Carpinteiro, A. J. R. Reis, and A. P. A. da Silva, “A hierarchical neural model in short-term load forecasting,” *Applied Soft Computing*, vol. 4, no. 4, pp. 405–412, 2004.

[77] S. F. Crone, J. Guajardo, and R. Weber, “A study on the ability of support vector regression and neural networks to forecast basic time series patterns,” in *IFIP International Conference on Artificial Intelligence in Theory and Practice.* Springer, 2006, Conference Proceedings, pp. 149–158.

[78] H. T. Siegelmann and E. D. Sontag, “On the computational power of neural nets,” in *Proceedings of the fifth annual workshop on Computational learning theory.* ACM, 1992, Conference Proceedings, pp. 440–449.

[79] D. R. Seidl and R. D. Lorenz, “A structure by which a recurrent neural network can approximate a nonlinear dynamic system,” in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 2. IEEE, 1991, Conference Proceedings, pp. 709–714.

[80] H. T. Siegelmann, B. G. Horne, and C. L. Giles, “Computational capabilities of recurrent narx neural networks,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 2, pp. 208–215, 1997.

[81] T. Lin, B. G. Horne, and C. L. Giles, “How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies,” *Neural Networks*, vol. 11, no. 5, pp. 861–868, 1998.

[82] J. Buitrago and S. Asfour, "Short-term forecasting of electric loads using nonlinear autoregressive artificial neural networks with exogenous vector inputs," *Energies*, vol. 10, no. 1, p. 40, 2017.

[83] M. Chui, "Artificial intelligence the next digital frontier?." *McKinsey and Company Global Institute*, p. 47, 2017.

[84] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[85] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[86] G. E. Hinton, "Learning translation invariant recognition in a massively parallel networks," in *International Conference on Parallel Architectures and Languages Europe.* Springer, 1987, Conference Proceedings, pp. 1–13.

[87] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[88] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239–242, 1990.

[89] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *J. Artif. Intell. Res.(JAIR)*, vol. 11, pp. 241–276, 1999.

[90] K. L. Ho, Y. Y. Hsu, and C. C. Yang, "Short term load forecasting using a multilayer neural network with an adaptive learning algorithm," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 141–149, 1992.

[91] N. V. Chawla and G. I. Karakoulas, "Learning from labeled and unlabeled data: An empirical study across techniques and domains," *J. Artif. Intell. Res.(JAIR)*, vol. 23, pp. 331–366, 2005.

[92] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[93] F. Burden and D. Winkler, "Bayesian regularization of neural networks," *Artificial Neural Networks: Methods and Applications*, vol. Chapter 3, no. 1, pp. 23–42, 2009.

[94] J. Y. Halpern, "Conditional plausibility measures and bayesian networks," *Journal of Artificial Intelligence Research*, vol. 14, pp. 369–399, 2001.

[95] S. Sanghai, P. Domingos, and D. Weld, "Relational dynamic bayesian networks," *Journal of Artificial Intelligence Research*, vol. 24, pp. 759–797, 2005.

[96] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[97] J. E. Moody, "Note on generalization, regularization and architecture selection in nonlinear learning systems," in *Neural Networks for Signal Processing [1991]., Proceedings of the 1991 IEEE Workshop.* IEEE, 1991, Conference Proceedings, pp. 1–10.

[98] ——, "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems," in *NIPS*, vol. 4, 1991, Conference Proceedings, pp. 847–854.

[99] G. B. Orr and K.-R. MÃijller, *Neural networks: tricks of the trade.* Springer, 2003.

[100] T. S. Rognvaldsson, *A simple trick for estimating the weight decay parameter.* Springer, 1998, book section 3, pp. 71–92.

[101] T. S. Dillon, S. Sestito, and S. Leung, "Short term load forecasting using an adaptive neural network," *International Journal of Electrical Power and Energy Systems*, vol. 13, no. 4, pp. 186–192, 1991.

[102] S. J. Hanson and L. Y. Pratt, *Comparing biases for minimal network construction with back-propagation.* Morgan Kaufmann Pub, 1989, vol. 1.

[103] N. K. Treadgold and T. D. Gedeon, "Simulated annealing and weight decay in adaptive learning: The sarprop algorithm," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 662–668, 1998.

[104] A. Krogh, J. Moody, S. Hanson, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems*, vol. 4, pp. 950–957, 1995.

[105] I. N. E. ISO, "Energy, load and demand reports," ISO New England, Report, 12/02/2015 2015.

[106] B. Osgood, "The fourier transform and its applications," *Electrical Engineering Department, Stanford University*, 2009.

[107] M. Venturini, "Simulation of compressor transient behavior through recurrent neural network models," *Journal of Turbomachinery-Transactions of the Asme*, vol. 128, no. 3, pp. 444–454, 2006.

[108] E. Irigoyen and M. Pinzolas, “Numerical bounds to assure initial local stability of narx multilayer perceptrons and radial basis functions,” *Neurocomputing*, vol. 72, no. 1âĂŞ3, pp. 539–547, 2008.

[109] S. Haykin, *Neural Networks: A comprehensive foundation*, ser. Neural Networks. McMillan College Publishing Company, 2004, vol. 2.

[110] M. T. Hagan and H. B. Demuth, “Neural networks for control,” in *American Control Conference, 1999. Proceedings of the 1999*, vol. 3. IEEE, 1999, Conference Proceedings, pp. 1642–1656.

[111] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De JesÃžs, *Neural network design.* PWS publishing company Boston, 1996, vol. 20.

[112] C. M. Bishop, *Neural networks for pattern recognition.* Springer Science + Business Media, 2006.

[113] K. Luk, J. E. Ball, and A. Sharma, “A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting,” *Journal of Hydrology*, vol. 227, no. 1, pp. 56–65, 2000.

[114] H. Demuth and M. Beale, “Neural network toolbox for use with matlab,” *Matlab Toolbox Manuals*, vol. R2016b, 2016.